

Ричард С. Саттон, Эндрю Дж. Барто

Обучение с подкреплением

Введение

Reinforcement Learning

An Introduction

Second Edition

Richard S. Sutton and Andrew G. Barto



The MIT Press

Обучение с подкреплением

Введение

Второе издание

Ричард С. Саттон, Эндрю Дж. Барто



Москва, 2020

УДК 004.85
ББК 32.971.3
С21

Саттон Р. С., Барто Э. Дж.

С21 Обучение с подкреплением: Введение. 2-е изд. / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2020. – 552 с.: ил.

ISBN 978-5-97060-097-9

Идея обучения с подкреплением возникла десятки лет назад, но этой дисциплине предстояло пройти долгий путь, прежде чем она стала одним из самых активных направлений исследований в области машинного обучения и нейронных сетей. Сегодня это предмет интереса ученых, занимающихся психологией, теорией управления, искусственным интеллектом и многими другими отраслями знаний.

Подход, принятый авторами книги, ставит акцент на практическое использование обучения с подкреплением. В первой части читатель знакомится с базовыми его аспектами. Во второй части представлены приближенные методы решения в условиях ограниченных вычислительных ресурсов. В третьей части книги обсуждается важность обучения с подкреплением для психологии и нейронаук.

Издание предназначено для студентов технических вузов, разработчиков, специализирующихся на машинном обучении и искусственном интеллекте, а также представителей нетехнических профессий, которые могут использовать описанные методики в своей работе.

УДК 004.85
ББК 32.971.3

Original English language edition published by The MIT Press Cambridge, MA. Copyright © 2018 Richard S. Sutton and Andrew G. Barto. Russian-language edition copyright © 2020 by DMK Press. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

ISBN 978-0-262-03924-6 (англ.)
ISBN 978-5-97060-097-9 (рус.)

Copyright © 2018 Richard S. Sutton and Andrew G. Barto
© Оформление, издание, перевод, ДМК Пресс, 2020

Посвящается памяти А. Гарри Клопфа

Содержание

Вступительное слово	11
Предисловие ко второму изданию	12
Предисловие к первому изданию	17
Обозначения	20
От издательства	25
Глава 1. Введение	26
1.1. Обучение с подкреплением.....	26
1.2. Примеры	30
1.3. Элементы обучения с подкреплением	31
1.4. Ограничения и круг вопросов	33
1.5. Развернутый пример: игра в крестики-нолики.....	34
1.6. Резюме	39
1.7. История ранних этапов обучения с подкреплением	39
Библиографические замечания.....	49
Часть I. ТАБЛИЧНЫЕ МЕТОДЫ РЕШЕНИЯ	50
Глава 2. Многорукие бандиты	51
2.1. Задача о k -руком бандите.....	51
2.2. Методы ценности действий	53
2.3. 10-рукий испытательный стенд.....	54
2.4. Инкрементная реализация.....	57
2.5. Нестационарная задача	59
2.6. Оптимистические начальные значения.....	60
2.7. Выбор действия, дающего верхнюю доверительную границу.....	62
2.8. Градиентные алгоритмы бандита.....	64
2.9. Ассоциативный поиск (контекстуальные бандиты).....	68
2.10. Резюме	69
Библиографические и исторические замечания.....	71
Глава 3. Конечные марковские процессы принятия решений	74
3.1. Интерфейс между агентом и окружающей средой.....	74
3.2. Цели и вознаграждения	80
3.3. Доход и эпизоды	82
3.4. Унифицированная нотация для эпизодических и непрерывных задач	84
3.5. Стратегии и функции ценности	86
3.6. Оптимальные стратегии и оптимальные функции ценности	91
3.7. Оптимальность и аппроксимация	96
3.8. Резюме	97
Библиографические и исторические замечания.....	99

Глава 4. Динамическое программирование	102
4.1. Оценивание стратегии (предсказание)	103
4.2. Улучшение стратегии	107
4.3. Итерация по стратегиям	109
4.4. Итерация по ценности	112
4.5. Асинхронное динамическое программирование	115
4.6. Обобщенная итерация по стратегиям	116
4.7. Эффективность динамического программирования	117
4.8. Резюме	118
Библиографические и исторические замечания	119
Глава 5. Методы Монте-Карло	122
5.1. Предсказание методами Монте-Карло	123
5.2. Оценивание ценности действий методом Монте-Карло	127
5.3. Управление методом Монте-Карло	129
5.4. Управление методом Монте-Карло без исследовательских стартов	132
5.5. Предсказание с разделенной стратегией посредством выборки по значимости	135
5.6. Инкрементная реализация	142
5.7. Управление методом Монте-Карло с разделенной стратегией	143
5.8. *Выборка по значимости с учетом обесценивания	146
5.9. *Приведенная выборка по значимости	147
5.10. Резюме	149
Библиографические и исторические замечания	150
Глава 6. Обучение на основе временных различий	152
6.1. Предсказание TD-методами	152
6.2. Преимущества TD-методов предсказания	157
6.3. Оптимальность TD(0)	159
6.4. Sarsa: TD-управление с единой стратегией	162
6.5. Q-обучение: TD-управление с разделенной стратегией	165
6.6. Expected Sarsa	167
6.7. Смещение максимизации и двойное обучение	169
6.8. Игры, послесостояния и другие специальные случаи	171
6.9. Резюме	173
Библиографические и исторические замечания	174
Глава 7. n-шаговый бутстрэппинг	176
7.1. n -шаговое TD-предсказание	176
7.2. n -шаговый алгоритм Sarsa	181
7.3. n -шаговое обучение с разделенной стратегией	184
7.4. *Приведенные методы с переменным управлением	186
7.5. Обучение с разделенной стратегией без выборки по значимости: n -шаговый алгоритм обновления по дереву	188
7.6. *Унифицированный алгоритм: n -шаговый Q(σ)	190
7.7. Резюме	193
Библиографические и исторические замечания	194
Глава 8. Планирование и обучение табличными методами	195
8.1. Модели и планирование	195
8.2. Дупа: объединение планирования, исполнения и обучения	198

8.3. Когда модель неверна	203
8.4. Приоритетный проход	206
8.5. Сравнение выборочного и полного обновлений	210
8.6. Траекторная выборка	213
8.7. Динамическое программирование в реальном времени	216
8.8. Планирование в момент принятия решений	220
8.9. Эвристический поиск	221
8.10. Разыгрывающие алгоритмы	224
8.11. Поиск по дереву методом Монте-Карло	226
8.12. Резюме главы	229
8.13. Резюме части I: оси	230
Библиографические и исторические замечания	233

Часть II. ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ 236

Глава 9. Предсказание с единой стратегией и аппроксимацией 238

9.1. Аппроксимация функции ценности	239
9.2. Целевая функция предсказания ($\bar{V}_E$)	240
9.3. Стохастические градиентные и полуградиентные методы	242
9.4. Линейные методы	246
9.5. Конструирование признаков для линейных методов	252
9.5.1. Полиномы	252
9.5.2. Базис Фурье	254
9.5.3. Грубое кодирование	257
9.5.4. Плиточное кодирование	260
9.5.5. Радиально-базисные функции	265
9.6. Выбор размера шага вручную	266
9.7. Нелинейная аппроксимация функций: искусственные нейронные сети	267
9.8. Алгоритм TD наименьших квадратов	272
9.9. Аппроксимация функций с запоминанием	274
9.10. Аппроксимация с помощью ядерных функций	276
9.11. Более глубокий взгляд на обучение с единой стратегией: заинтересованность и значимость	278
9.12. Резюме	280
Библиографические и исторические замечания	281

Глава 10. Управление с единой стратегией и аппроксимацией 288

10.1. Эпизодическое полуградиентное управление	288
10.2. Полуградиентный n -шаговый Sarsa	292
10.3. Среднее вознаграждение: новая постановка непрерывных задач	294
10.4. Возражения против постановки с обесцениванием	299
10.5. Дифференциальный полуградиентный n -шаговый Sarsa	301
10.6. Резюме	302
Библиографические и исторические замечания	303

Глава 11. *Методы с разделенной стратегией и аппроксимацией 304

11.1. Полуградиентные методы	305
11.2. Примеры расходимости в случае с разделенной стратегией	307
11.3. Смертельная триада	312

11.4. Геометрия линейной аппроксимации функций ценности	314
11.5. Градиентный спуск по беллмановской ошибке	318
11.6. Беллмановская ошибка необучаема	322
11.7. Градиентные TD-методы	327
11.8. Эмфатические TD-методы	330
11.9. Уменьшение дисперсии	332
11.10. Резюме	334
Библиографические и исторические замечания	335

Глава 12. Следы приемлемости

12.1. λ -доход	338
12.2. TD(λ)	342
12.3. n -шаговые усеченные λ -доходные методы	346
12.4. Пересчет обновлений: онлайнный λ -доходный алгоритм	348
12.5. Истинно онлайнный TD(λ)	350
12.6. *Голландские следы в обучении методами Монте-Карло	352
12.7. Sarsa(λ)	354
12.8. Переменные λ и γ	359
12.9. Следы с разделенной стратегией и переменным управлением	361
12.10. От Q(λ) Уоткинса к Tree-Backup(λ)	364
12.11. Устойчивые методы с разделенной стратегией со следами приемлемости	367
12.12. Вопросы реализации	368
12.13. Выводы	369
Библиографические и исторические замечания	371

Глава 13. Методы градиента стратегии

13.1. Аппроксимация стратегии и ее преимущества	374
13.2. Теорема о градиенте стратегии	376
13.3. REINFORCE: метод Монте-Карло на основе градиента стратегии	378
13.4. REINFORCE с базой	381
13.5. Методы исполнитель-критик	383
13.6. Метод градиента стратегии для непрерывных задач	385
13.7. Параметризация стратегии для непрерывных действий	388
13.8. Резюме	389
Библиографические и исторические замечания	390

Часть III. ЗАГЛЯНЕМ ПОГЛУБЖЕ

Глава 14. Психология

14.1. Предсказание и управление	394
14.2. Классическое обусловливание	395
14.2.1. Блокирующее обусловливание и обусловливание высшего порядка	397
14.2.2. Модель Рескорлы-Вагнера	399
14.2.3. TD-модель	401
14.2.4. Имитирование TD-модели	403
14.3. Инструментальное обусловливание	410
14.4. Отложенное подкрепление	415
14.5. Когнитивные карты	416
14.6. Привычное и целеустремленное поведение	418
14.7. Резюме	423
Библиографические и исторические замечания	425

Глава 15. Нейронауки	432
15.1. Основы нейронаук	433
15.2. Сигналы вознаграждения, сигналы подкрепления, ценности и ошибки предсказания	435
15.3. Гипотеза об ошибке предсказания вознаграждения	437
15.4. Дофамин	439
15.5. Экспериментальное подтверждение гипотезы об ошибке предсказания вознаграждения	443
15.6. Параллель между TD-ошибкой и дофамином	447
15.7. Нейронный исполнитель – критик	452
15.8. Правила обучения критика и исполнителя	456
15.9. Гедонистические нейроны	460
15.10. Коллективное обучение с подкреплением	462
15.11. Основанные на модели методы в мозге	466
15.12. Наркотическая зависимость	468
15.13. Резюме	469
Библиографические и исторические замечания	472
Глава 16. Примеры и приложения	481
16.1. TD-Gammon	481
16.2. Программы игры в шашки Сэмюэла	486
16.3. Стратегия выбора ставки в программе Watson	489
16.4. Оптимизация управления памятью	492
16.5. Игра в видеоигры на уровне человека	497
16.6. Мастерство игры в го	503
16.6.1. AlphaGo	506
16.6.2. AlphaGo Zero	509
16.7. Персонализированные веб-службы	513
16.8. Парение в восходящих потоках воздуха	516
Глава 17. Передовые рубежи	521
17.1. Общие функции ценности и вспомогательные задачи	521
17.2. Абстрагирование времени посредством опций	523
17.3. Наблюдения и состояния	526
17.4. Проектирование сигналов вознаграждения	532
17.5. Остающиеся вопросы	535
17.6. Экспериментальное подтверждение гипотезы об ошибке предсказания вознаграждения	539
Библиографические и исторические замечания	543
Предметный указатель	587

Вступительное слово от ГК «Цифра»

Прошло уже несколько лет с тех пор, как наша команда ступила на путь применения искусственного интеллекта для совершенствования процессов в промышленности и логистике. В самом начале мы и представить не могли, насколько тернистой, но в то же время невероятно интересной окажется эта дорога. За это время мы успели поработать с различными производствами и решить множество задач – от оптимизации производства битумных материалов до улучшения системы распределения нефтепродуктов и внедрения систем машинного зрения на карьерные экскаваторы. Методы машинного обучения, которые используются для решения подобных задач, постоянно совершенствуются, и мы внимательно следим за развитием подходов в области искусственного интеллекта, в том числе за исследованиями в обучении с подкреплением.

Обучение с подкреплением – это один из разделов машинного обучения, исследующий вычислительный подход к обучению агента, который пытается максимизировать свою совокупную накопленную награду путем взаимодействия со сложной, зачастую стохастической средой. Последние несколько лет исследования этого подхода переживают настоящий ренессанс – ни одна научная конференция по искусственному интеллекту не обходится без секции на эту тему. Каждый год публикуются сотни научных статей, и все больше компаний в России и за рубежом начинают применять последние достижения этой области в своем бизнесе для улучшения различных внутренних процессов – от рекомендательных систем до оптимизации цепей поставок.

Мы видим огромный потенциал практического применения методов обучения с подкреплением для совершенствования процессов в промышленности и логистике, а также верим в решающее значение данных теоретических концепций и алгоритмов для прогресса искусственного интеллекта как области человеческого знания. Несмотря на огромный интерес к этой области в последнее время, по указанной теме издано не так много литературы. Именно поэтому мы решили поучаствовать в публикации этой замечательной книги на русском языке.

Данная книга представляет собой исчерпывающее введение в такую интересную и быстро развивающуюся область искусственного интеллекта, как обучение с подкреплением. Ее авторы, Ричард Саттон и Эндрю Барто, проделали невероятную работу, описав простым и понятным языком не только ключевые концепции и алгоритмы обучения с подкреплением, но и современные достижения этой области. В книге продемонстрирована связь дисциплины с психологией и нейронауками. Авторами подробно рассматриваются детали работы системы AlphaGo, обыгравшей чемпиона мира в японскую настольную игру го, а также алгоритма, играющего в игры Atari на уровне человека, и многие другие приложения.

Мы желаем читателю удачи на пути изучения такой сложной, но невероятно полезной и увлекательной дисциплины.

Сергей Свиридов,
директор по исследованиям и разработкам, группа компаний «Цифра»



Группа компаний «Цифра» разрабатывает технологии цифровизации промышленности, инвестирует в продукты и развивает среду промышленного интернета вещей и искусственного интеллекта. Компания создала самую крупную в России лабораторию промышленного AI. Сегодня решения «Цифры» повышают эффективность промышленных предприятий в 22 странах мира. Ключевые отрасли для группы – это горная добыча и металлургия, машиностроение, нефтегазовый сектор и химическая промышленность. «Цифра входит» в Industrial Internet Consortium и ряд других российских и международных отраслевых ассоциаций.

Предисловие ко второму изданию

За двадцать лет, прошедших после выхода первого издания этой книги, мы стали свидетелями колоссального прогресса в области искусственного интеллекта, в немалой степени обусловленного достижениями машинного обучения, в т. ч. обучения с подкреплением. И этот прогресс был достигнут не только за счет впечатляющего роста вычислительных мощностей, но и благодаря развитию теории и алгоритмов. Поэтому необходимость во втором издании книги, вышедшей в 1998 году, давно назрела и перезрела, и наконец-то в 2012 году мы решили приняться за нее. Во втором издании мы ставили себе ту же цель, что и в первом: дать простое и понятное изложение основных идей и алгоритмов обучения с подкреплением, которое было бы доступно специалистам из смежных дисциплин. Книга по-прежнему осталась введением, основное внимание уделяется базовым алгоритмам онлайн-обучения. Мы включили ряд новых вопросов, возникших и приобретших важность за прошедшие годы, а также расширили описание тем, которые теперь понимаем лучше. Но мы даже не пытались дать исчерпывающее изложение всего предмета, который стремительно развивался во многих направлениях. Приносим извинения за то, что были вынуждены оставить все эти достижения (за исключением небольшого числа) без внимания.

Как и в первом издании, мы решили отказаться от строго формального изложения теории обучения с подкреплением и от постановки задачи в самом общем виде. Но по мере углубления нашего понимания некоторых вопросов потребовалось включить больше математики; части, для которых необходимо более уверенное владение математическим аппаратом, оформлены в виде врезок; читатели, не склонные к математике, могут их пропустить. Мы также используем не совсем такую же нотацию, как в первом издании. В процессе преподавания мы поняли, что новая нотация помогает устранить ряд распространенных недоразумений. Она подчеркивает различие между случайными величинами, которые обозначаются заглавными буквами, и их экземплярами, обозначаемыми строчными буквами. Например, состояние, действие и вознаграждение на временном шаге t обозначаются S_t , A_t и R_t , а их возможные значения – s , a и r . Кроме того, строчными буквами записываются функции ценности (например, v_π), а заглавными – их табличные представления (например, $Q_t(s, a)$). Приближенные функции ценности являются детерминированными функциями случайных параметров, поэтому также записываются строчными буквами (например, $\hat{v}(s, \mathbf{w}_t) \approx v_\pi(s)$). Векторы, например вектор весов $\mathbf{w}_t$ (ранее обозначался θ_t) и вектор признаков $\mathbf{x}_t$ (ранее ϕ_t), записываются строчными полужирными буквами, даже если являются случайными величинами. Заглавные полужирные буквы оставлены для матриц. В первом издании мы употребляли специальные обозначения $\mathcal{P}_{ss'}^a$ и $\mathcal{R}_{ss'}^a$ для вероятности перехода и ожидаемого вознаграждения. Один из недостатков этой нотации заключается в том, что она не полностью характеризует динамику вознаграждения,

а дает только математические ожидания – этого достаточно для динамического программирования, но не для обучения с подкреплением. Другой недостаток – чрезмерное количество верхних и нижних индексов. В этом издании мы ввели явное обозначение $p(s', r | s, a)$ для совместной вероятности следующего состояния и вознаграждения при условии текущего состояния и действия. Все изменения нотации сведены в таблице на стр. 20.

Второе издание значительно дополнено, и организация материала претерпела изменения. После первой вводной главы появились три новые части. В первой части (главы 2–8) обучение с подкреплением рассматривается настолько полно, насколько возможно без выхода за пределы табличного случая, для которого можно найти точные решения. Мы включили методы обучения и планирования для табличного случая, а также их унификацию в n -шаговых методах и в архитектуре Дупа. Многих алгоритмов, представленных в этой части, в первом издании не было, например: UCB, Expected Sarsa, двойное обучение, обновление по дереву, $Q(\sigma)$, RTDP и MCTS. Подробное рассмотрение табличного случая в начале книги позволяет изложить основные идеи в простейшей постановке. Вторая часть книги (главы 9–13) посвящена обобщению этих идей на аппроксимации функций. В ней появились новые разделы об искусственных нейронных сетях, о базисе Фурье, LSTD, ядерных методах, методах Gradient-TD и Emphatic-TD, методах среднего вознаграждения, истинно онлайн-методе TD(λ) и методах градиента стратегии. Во втором издании намного подробнее рассмотрено обучение с разделенной стратегией, сначала в табличном случае (главы 5–7), а затем для аппроксимации функций в главах 11 и 12. Еще одно отличие второго издания заключается в отделении идеи прямого представления, связанной с n -шаговым бутстрэппингом (теперь она более полно рассмотрена в главе 7), от идеи обратного представления, связанной со следами приемлемости (она теперь независимо описана в главе 12). В третью часть книги включены новые большие главы о связях обучения с подкреплением с психологией (глава 14) и нейронауками (глава 15), а также переработанная глава с примерами, включающая игры Atari, стратегию ставок в программе Watson, а также две программы игры в го: AlphaGo и AlphaGo Zero (глава 16). Но по необходимости мы смогли включить лишь малую часть сделанного в этой области. Выбор отражает наш давний интерес к недорогим безмодельным методам, которые хорошо масштабируются на крупные приложения. Последняя глава посвящена обсуждению будущего влияния обучения с подкреплением на общество. Хорошо это или плохо, но второе издание получилось почти в два раза больше первого.

Эта книга задумывалась как основной учебник для одно- или двухсеместрового курса по обучению с подкреплением. В односеместровый курс следует включить первые десять глав и излагать их по порядку. Это составит хорошую основу, к которой можно добавить материал из других глав, а также из других книг, например Bertsekas and Tsitsiklis (1996), Wiering and van Otterlo (2012), Szepesvári (2010), или из литературы – сообразуясь со вкусами лектора. В зависимости от подготовки студентов может оказаться полезным дополнительный материал по онлайн-обучению с учителем. Естественным дополнением будут идеи опций и моделей опций (Sutton, Precup and Singh, 1999). В двухсеместровый курс можно включить все главы и дополнительные материалы. Эту книгу можно также включить как часть более широких курсов машинного обучения, искусственного интеллекта или ней-

ронных сетей. В таком случае имеет смысл рассматривать только некоторое подмножество глав. Мы рекомендуем главу 1 в качестве краткого обзора, главу 2 до раздела 2.4, главу 3, а затем избранные разделы остальных глав в зависимости от располагаемого времени и интересов лектора и аудитории. Глава 6 наиболее важна для предмета и всей книги. В курс, ориентированный на машинное обучение или нейронные сети, следует включить главы 9 и 10, а в курс, ориентированный на искусственный интеллект или планирование, – главу 8. Разделы и главы, которые мы считаем более трудными и не существенными для книги в целом, помечены звездочкой. Их можно опустить при первом чтении без ущерба для понимания последующего текста. Упражнения повышенной сложности также помечены звездочкой, они не существенны для усвоения основного материала главы.

Большинство глав заканчиваются разделом «Библиографические и исторические замечания», в которых мы перечисляем источники идей, изложенных в главе, приводим ссылки на литературу для дальнейшего чтения и на текущие исследовательские работы, а также даем историческую справку. Несмотря на все усилия сделать эти разделы полными и авторитетными, мы наверняка упустили какие-то важные работы предшественников. Приносим свои извинения и открыты для исправлений и дополнений, которые будут внесены в электронную версию книги.

Это издание, как и первое, посвящено памяти А. Гарри Клопфа. Именно Гарри познакомил нас друг с другом, и именно его идеи о мозге и искусственном интеллекте побудили нас отправиться в долгое путешествие по миру обучения с подкреплением. Гарри получил образование в области нейрофизиологии и очень интересовался машинным интеллектом, он работал старшим научным сотрудником в отделе авионики Управления научно-исследовательских работ ВВС США (AFOSR) при базе ВВС Райт-Паттерсон в штате Огайо. Он был недоволен тем, что процессам поиска равновесия, в т. ч. гомеостазу и методам классификации на основе исправления ошибок, придают чрезмерно большую важность при объяснении естественного интеллекта и закладывания фундамента машинного интеллекта. Он отмечал, что системы, пытающиеся что-то максимизировать (не важно, что именно), качественно отличаются от систем поиска равновесия, и доказывал, что именно в максимизирующих системах ключ к пониманию важных аспектов естественного интеллекта и построения искусственного. Гарри сыграл решающую роль в получении от AFOSR финансирования для проекта оценки научной ценности этих и родственных им идей. Этот проект был запущен в конце 1970-х годов в Массачусетском университете в Амхерсте (UMass Amherst), сначала под руководством Майкла Эрбиба (Michael Arbib), Уильяма Килмера (William Kilmer) и Нико Спинелли (Nico Spinelli), профессоров факультета компьютерных и информационных наук и членов-основателей университетского кибернетического центра нейронаучных систем, созданного с перспективой работы на стыке нейронаук и искусственного интеллекта. Барто, недавно получивший докторскую степень в Мичиганском университете, был принят в проект на должность младшего научного сотрудника. Тем временем Саттон, студент старшего курса, изучавший информатику и психологию в Стэнфорде, переписывался с Гарри на тему их общего интереса к роли временных характеристик возбудителя в классической теории обусловливания. Гарри убедил группу в UMass в том, что Саттон станет отличным приобретением для проекта. Так Саттон оказался аспирантом

в UMass и начал писать докторскую диссертацию под руководством Барто, который к тому времени занял должность доцента. Исследования обучения с подкреплением, описанные в этой книге, – закономерный итог проекта, начатого Гарри и питавшегося его идеями. Таким образом, Гарри свел нас, авторов книги, положив начало долгой и плодотворной совместной работе. Посвящая эту книгу Гарри, мы отдаем должное его существенному вкладу не только в дисциплину обучения с подкреплением, но и в наше сотрудничество. Мы также выражаем благодарность профессорам Эрбибу, Килмеру и Спинелли за предоставленную нам возможность начать разработку этих идей. Наконец, мы благодарны AFOSR за щедрую поддержку, которую управление оказывало на ранней стадии наших исследований, и Национальному научному фонду (NSF) за щедрое финансирование в течение ряда последующих лет.

Есть много людей, которым мы благодарны за их идеи и помощь в подготовке второго издания. Все, кого мы благодарили за помощь в первом издании, заслуживают нашей глубочайшей благодарности и за это издание тоже – оно бы просто не состоялось без их вклада в первое издание. К этому длинному перечню мы обязаны добавить многих, кто помогал готовить только второе издание. Студенты, которым мы много лет преподавали эту дисциплину, отметились самыми разными способами: находили ошибки, предлагали исправления и – не в последнюю очередь – испытывали затруднения, заставляя нас думать, как объяснить материал лучше. Мы выражаем особую благодарность Марте Стинструп (Martha Steenstrup), которая прочитала весь текст и поделилась подробными комментариями. Главы по психологии и нейронаукам не были бы написаны без помощи многочисленных специалистов в этих областях. Мы признательны Джону Муру (John Moore) за его многолетние терпеливые разъяснения теории и экспериментов по обучению животных и основ нейронауки, а также за внимательное прочтение нескольких черновых вариантов глав 14 и 15. Мы также благодарны Мэтту Ботвинику (Matt Botvinick), Натаниэлю Дой (Nathaniel Daw), Питеру Дайяну (Peter Dayan) и Йелю Ниву (Yael Niv) за проницательные замечания к черновикам этих глав, помощь в освоении огромного массива литературы и указание на наши многочисленные ошибки в ранних вариантах рукописи. Разумеется, все оставшиеся ошибки в этих главах (а их не может не быть) – целиком наша вина. Мы выражаем благодарность Филу Томасу (Phil Thomas), который помог сделать эти главы доступными неспециалистам в области психологии и нейронаук, и Питеру Стерлингу (Peter Sterling), помогавшему сделать объяснения более понятными. Спасибо также Джиму Хоуку (Jim Houk) за знакомство с вопросами обработки информации в подкорковых ядрах головного мозга и за привлечение нашего внимания к смежным разделам нейронауки. Хосе Мартинес (José Martinez), Терри Сейновски (Terry Sejnowski), Дэвид Силвер (David Silver), Джерри Тезауро (Gerry Tesauro), Георгиос Теочарус (Georgios Theodoros) и Фил Томас (Phil Thomas) любезно помогли нам разобраться в деталях их приложений обучения с подкреплением, чтобы мы могли включить их в главу с примерами. Они же поделились ценными комментариями к черновым вариантам соответствующих разделов. Отдельное спасибо Дэвиду Силверу, который помог нам лучше понять дерево поиска Монте-Карло и программу DeepMind для игры в го. Мы также благодарны Джорджу Конидарису (George Konidaris) за помощь при написании раздела о базисе Фурье. Эмилио Картони (Emilio Cartoni), Томас Седерборг (Thomas Cederborg), Стефан Дерббах

(Stefan Dernbach), Клеменс Розенбаум (Clemens Rosenbaum), Патрик Тэйлор (Patrick Taylor), Томас Колин (Thomas Colin) и Пьер-Люк Бэкон (Pierre-Luc Bacon) помогли нам различными способами, за что мы им очень благодарны.

Саттон также выражает благодарность сотрудникам лаборатории обучения с подкреплением и искусственного интеллекта в университете Альберты за вклад во второе издание. Отдельное спасибо Рупаму Махмуду (Rupam Mahmood) за ценный вклад в обсуждение методов Монте-Карло обучения с разделенной стратегией в главе 5, Хамиду Мэю (Hamid Maei) за помощь в становлении взгляда на обучение с разделенной стратегией, представленного в главе 11, Эрику Грейвсу (Eric Graves) за постановку экспериментов в главе 13, Шан-тон Чжану (Shangdong Zhang) за воспроизведение и, как следствие, проверку почти всех экспериментальных результатов, Крису де Асису (Kris De Asis) за улучшение нового технического наполнения глав 7–12 и Харму ван Сейну (Harm van Seijen) за идеи, которые привели к отделению n -шаговых методов от следов приемлемости и (совместно с Хадом ван Хасселтом [Hado van Hasselt]) – за идеи, касающиеся точной эквивалентности прямого и обратного представления следов приемлемости (глава 12). Саттон также выражает признательность за финансовую поддержку и свободу исследований, которые обеспечивали правительство провинции Альберты и Национальный совет научных и инженерных исследований Канады на протяжении всей работы над вторым изданием книги. В частности, он благодарен Рэнди Гебелю (Randy Goebel) за создание благоприятной среды для исследований в Альберте с прицелом на перспективу. Также он благодарен компании DeepMind за поддержку на протяжении последних шести месяцев работы над книгой.

Наконец, мы признательны многочисленным придирчивым читателям черновых вариантов второго издания, которые мы выкладывали в интернет. Они нашли немало пропущенных нами ошибок и указали места, где может возникнуть недопонимание.

Предисловие к первому изданию

То, что теперь называется обучением с подкреплением, впервые привлекло наше внимание в конце 1979 года. Мы оба работали в Массачусетском университете над одним из ранних проектов воскрешения идеи сетей с нейроноподобными адаптивными элементами, которая могла оказаться многообещающим подходом к искусственному адаптивному интеллекту. Проект был посвящен исследованию «гестеростатической теории адаптивных систем» и разрабатывался под руководством А. Гарри Клопфа. Работа Гарри была богатейшим источником идей, а нам было позволено критически изучить их и сравнить с долгой историей предшествующих исследований в области адаптивных систем. Нашей задачей стало расчленение этих идей на составные части в попытке понять их взаимосвязи и сравнительную важность. Это продолжается и по сей день, но в 1979 году мы впервые осознали, что самая простая идея, которую долго считали чем-то само собой разумеющимся, удостоилась на удивление скромного внимания с вычислительной точки зрения. Это была идея обучающейся системы, которая хочет чего-то достичь и для этого адаптирует свое поведение так, чтобы максимизировать специальный сигнал со стороны окружающей среды. Иначе говоря, идея «гедонистической» обучающейся системы, или, как мы сказали бы теперь, идея обучения с подкреплением.

Как и многие другие, мы полагали, что обучение с подкреплением было всесторонне исследовано еще на заре развития кибернетики и искусственного интеллекта. Но при ближайшем рассмотрении оказалось, что его изучали очень поверхностно. Хотя обучение с подкреплением, безусловно, стало побудительным мотивом для некоторых ранних компьютерных исследований обучения, большая часть занимавшихся этим ученых затем обратились к другим вещам: классификации образов, обучению с учителем или адаптивному управлению, а то и вовсе забросили исследования в области обучения. В результате специальным вопросам, связанным с тем, как обучиться получать что-нибудь от среды, было уделено сравнительно мало внимания. Оглядываясь назад, можно сказать, что интерес к этой идее стал важнейшим шагом, приведшим в движение всю эту ветвь исследований. Мало чего можно было бы достичь в плане вычислительного обучения с подкреплением, не осознав, что столь фундаментальная идея ранее не была досконально исследована.

С тех пор эта область науки прошла долгий путь, развивалась в нескольких направлениях и стала зрелой дисциплиной. Обучение с подкреплением постепенно стало одним из самых активных направлений исследований в машинном обучении, искусственном интеллекте и нейронных сетях. Было подведено солидное математическое основание и созданы впечатляющие приложения. Компьютерные исследования обучения с подкреплением превратились в обширную область, в которой трудятся сотни ученых по всему миру, занимающиеся такими разными дисциплинами, как психология, теория управления, искусственный интеллект

и нейронауки. Особенно важны результаты, устанавливающие и развивающие связи с теорией оптимального управления и динамическим программированием. В целом проблема обучения путем взаимодействия ради достижения поставленных целей еще далека от решения, но наше понимание стало значительно глубже. Мы теперь можем изучать отдельные направления, например обучение на основе временных различий, динамическое программирование и аппроксимацию функций, в контексте их вклада в решение общей проблемы.

Принимаясь за написание книги, мы ставили цель дать простое и ясное описание ключевых идей и алгоритмов обучения с подкреплением. Мы хотели, чтобы изложение было доступно читателям, работающим в смежных дисциплинах, но не могли охватить их все одинаково подробно. В основном мы ведем изложение с точки зрения искусственного интеллекта и технического конструирования. Рассмотрение связей с иными областями мы оставляем другим авторам или отложим до следующего раза. Мы также решили отказаться от строго формального изложения предмета. Мы не стремились к максимальному уровню математического абстрагирования и не пытались доказывать теоремы. Мы постарались выбрать такой уровень математических деталей, который указал бы читателям с математическим складом ума верное направление, но не отвлекал от простоты и потенциальной общности базовых идей.

В некотором смысле мы работали над этой книгой тридцать лет, так что нам есть кого благодарить. Прежде всего мы благодарим людей, лично помогавших нам в разработке идей, представленных в книге: Гарри Клопфа (Harri Klopff), который помог нам осознать, что обучение с подкреплением нуждается в новой жизни; Криса Уоткинса (Chris Watkins), Димитрия Бертсекаса (Dimitri Bertsekas), Джона Цициклиса (John Tsitsiklis) и Пода Вербоса (Paul Werbos), которые помогли нам понять ценность связей с динамическим программированием; Джона Мура (John Moore) и Джима Кехоу (Jim Kehoe) за идеи из теории обучения животных; Оливера Селфриджа (Oliver Selfridge) за подчеркивание широты и важности адаптации; и вообще наших коллег и студентов, помогавших самыми разными способами: Рона Уильямса (Ron Williams), Чарльза Андерсона (Charles Anderson), Сатиндера Сингха (Satinder Singh), Сридхара Махадевана (Sridhar Mahadevan), Стива Браттке (Steve Bradtke), Боба Крайтца (Bob Crites), Питера Дайяна (Peter Dayan) и Лимона Бэрда (Leemon Baird). На наши воззрения на обучение с подкреплением оказали большое влияние беседы с Полом Коэном (Paul Cohen), Полом Утгоффом (Paul Utgoff), Мартой Стинструп (Martha Steenstrup), Джерри Тезауро (Gerry Tesauro), Майком Джорданом (Mike Jordan), Лесли Кэлблингом (Leslie Kaelbling), Эндрю Муром (Andrew Moore), Крисом Аткисоном (Chris Atkeson), Томом Митчеллом (Tom Mitchell), Нильсом Нильсоном (Nils Nilsson), Стюартом Расселом (Stuart Russell), Томом Диттерихом (Tom Dietterich), Томом Дином (Tom Dean) и Бобом Нарендра (Bob Narendra).

Мы благодарны Майклу Литтману, Джерри Тезауро, Майклу Крайтсу, Сатиндеру Сингху и Вэй Чжану (Wei Zhang) за наполнение конкретикой разделов 4.7, 15.1, 15.4, 15.5 и 15.6 соответственно. Мы благодарим Управление научно-исследовательских работ ВВС США, Национальный научный фонд и лаборатории GTE за длительную финансовую поддержку, нацеленную на перспективу. Мы также выражаем признательность многим людям, которые читали черновые варианты книги и делились ценными замечаниями: Тому Калту (Tom Kalt), Джону Цициклису,

Павлу Чихошу (Pawel Cichosz), Олле Гэллмо (Olle Gällmo), Чаку Андерсону (Chuck Anderson), Стюарту Расселу, Бену ван Рою (Ben Van Roy), Полу Стинструпу (Paul Steenstrup), Полу Коэну, Сридхару Махадевану, Джетте Рандлов (Jette Randlov), Брайану Шеппарду (Brian Sheppard), Томасу О'Коннелу (Thomas O'Connell), Ричарду Коггинсу (Richard Coggins), Кристине Версино (Cristina Versino), Джону Х. Хайетту (John H. Hiatt), Андреасу Баделту (Andreas Badelt), Джею Понте (Jay Ponte), Джо Беку (Joe Beck), Юстусу Пиатеру (Justus Piater), Марту Стинstrup, Сатиндеру Сингху, Томми Яаколла (Jaakkola), Димитрию Бертсекасу (Dimitri Bertsekas), Торбьёрну Экману (Torbjörn Ekman), Кристине Бьёркман (Christina Björkman), Якобу Карлстрёму (Jakob Carlström) и Олле Палмгрену (Olle Palmgren). Наконец, мы благодарим Гвин Митчелл (Gwyn Mitchell) за разнообразную помощь, а также Гарри Стэнтона (Harry Stanton) и Боба Приора (Bob Prior), которые опекали нас в издательстве MIT Press.

Обозначения

Заглавными буквами обозначаются случайные величины, строчными – значения случайных величин и скалярные функции. Вещественные векторы записываются строчными полужирными буквами (даже если они являются случайными величинами), матрицы – заглавными полужирными буквами.

$\doteq$	равенство, имеющее место по определению
$\approx$	приближенное равенство
$\propto$	пропорционально
$\Pr\{X = x\}$	вероятность, что случайная величина X принимает значение x
$X \sim p$	случайная величина X выбрана из распределения $p(x) \doteq \Pr\{X = x\}$
$\mathbb{E}[X]$	математическое ожидание случайной величины X , т. е. $\mathbb{E}[X] \doteq \sum_x p(x)x$
$\operatorname{argmax}_a f(a)$	значение a , в котором $f(a)$ достигает максимума
$\ln x$	натуральный логарифм x
e^x	основание натуральных логарифмов, число $e \approx 2.71828$, возведенное в степень x ; $e^{\ln x} = x$
$\mathbb{R}$	множество вещественных чисел
$f: \mathcal{X} \rightarrow \mathcal{Y}$	функция f , отображающая элементы множества $\mathcal{X}$ в элементы множества $\mathcal{Y}$
$\leftarrow$	присваивание
$(a, b]$	интервал вещественной оси между a и b , включающий b , но не включающий a
ε	вероятность предпринять случайное действие в ε -жадной стратегии
α, β	параметры, определяющие размер шага
γ	коэффициент обесценивания
λ	коэффициент затухания для следов приемлемости
$\mathbb{1}_{\text{predicate}}$	индикаторная функция ($\mathbb{1}_{\text{predicate}} \doteq 1$, если предикат <i>predicate</i> равен true, в противном случае 0)

В задаче о многоруких бандитах:

k	количество действий (рук)
t	дискретный временной шаг или номер игры
$q_*(a)$	истинное значение (ожидаемое вознаграждение) действия a
$Q_t(a)$	оценка $q_*(a)$ в момент t
$N_t(a)$	сколько раз действие a выбиралось до момента t
$H_t(a)$	обученное предпочтение действию a в момент t
$\pi_t(a)$	вероятность выбора действия a в момент t
$\bar{R}_t$	оценка ожидаемого вознаграждения в момент t при условии π_t

В марковском процессе принятия решений:

s, s'	состояния
a	действие
r	вознаграждение
$\mathcal{S}$	множество всех незаключительных состояний
$\mathcal{S}^+$	множество всех состояний, включая заключительное
$\mathcal{A}(s)$	множество всех действий, допустимых в состоянии s
$\mathcal{R}$	множество всех возможных вознаграждений, конечное подмножество $\mathbb{R}$
$\subset$	подмножество (например, $\mathcal{R} \subset \mathbb{R}$)
$\in$	элемент множества, например $s \in \mathcal{S}, r \in \mathcal{R}$
$ \mathcal{S} $	количество элементов во множестве $\mathcal{S}$ (мощность $\mathcal{S}$)
t	дискретный временной шаг
$T, T(t)$	конечный временной шаг эпизода или эпизод, включающий временной шаг t
A_t	действие в момент t
S_t	состояние в момент t , обычно стохастически зависящее от S_{t-1} и A_{t-1}
R_t	вознаграждение в момент t , обычно стохастически зависящее от S_{t-1} и A_{t-1}
π	стратегия (правило принятия решения)
$\pi(s)$	действие, предпринятое в состоянии s при <i>детерминированной</i> стратегии π
$\pi(a s)$	вероятность предпринять действие a в состоянии s при <i>стохастической</i> стратегии π
G_t	доход, начиная с момента t
h	горизонт, на какое время можно заглянуть вперед в прямом представлении
$G_{t:t+n}, G_{t:h}$	доход за n шагов с $t+1$ до $t+n$ или до h (обесцененный и скорректированный)
$\bar{G}_{t:h}$	плоский доход (необесцененный и нескорректированный) за шаги от $t+1$ до h (раздел 5.8)
G_t^λ	λ -доход (раздел 12.1)
$G_{t:h}^\lambda$	усеченный скорректированный λ -доход (раздел 12.3)
$G_t^{\lambda s}, G_t^{\lambda a}$	λ -доход, скорректированный на оценки ценности состояния или действия (раздел 12.8)
$p(s', r s, a)$	вероятность перехода в состояние s' с вознаграждением r из состояния s после действия a
$p(s' s, a)$	вероятность перехода в состояние s' из состояния s после действия a
$r(s, a)$	ожидаемое немедленное вознаграждение в состоянии s после действия a

$r(s, a, s')$	ожидаемое немедленное вознаграждение при переходе из s в s' после действия a
$v_\pi(s)$	ценность состояния s при стратегии π (ожидаемый доход)
$v_*(s)$	ценность состояния s при оптимальной стратегии
$q_\pi(s, a)$	ценность выполнения действия a в состоянии s при стратегии π
$q_*(s, a)$	ценность выполнения действия a в состоянии s при оптимальной стратегии
V, V_t	массив оценок функции ценности состояний v_π или v_*
Q, Q_t	массив оценок функции ценности действий q_π или q_*
$\bar{V}_t(s)$	ожидаемая приближенная ценность действия, например $V_t(s) \doteq \sum_a \pi(a s) Q_t(s, a)$
U_t	цель для оценки в момент t
δ_t	ошибка временного различия (TD-ошибка) в момент t (случайная величина) (раздел 6.1)
δ_t^s, δ_t^a	формы TD-ошибки для состояния и действия (раздел 12.9)
n	в n -шаговых методах n – количество шагов бутстрэппинга
$\ v\ _\mu^2$	μ -взвешенная квадратичная норма функции ценности, $\ v\ _\mu^2 \doteq \sum_{s \in \mathcal{S}} \mu(s) v(s)^2$
d	размерность – количество элементов $\mathbf{w}$
d'	альтернативная размерность – количество элементов θ
$\mathbf{w}, \mathbf{w}_t$	d -мерный вектор весов, определяющий приближенную функцию ценности
$w_i, w_{t,i}$	i -й элемент обучаемого вектора весов
$\hat{v}(s, \mathbf{w})$	приближенная ценность состояния s при условии вектора весов $\mathbf{w}$
$v_{\mathbf{w}}(s)$	альтернативное обозначение $\hat{v}(s, \mathbf{w})$
$\hat{q}(s, a, \mathbf{w})$	приближенная ценность пары состояние–действий s, a при заданном векторе весов $\mathbf{w}$
$\nabla \hat{v}(s, \mathbf{w})$	вектор-столбец частных производных $\hat{v}(s, \mathbf{w})$ по $\mathbf{w}$
$\nabla \hat{q}(s, a, \mathbf{w})$	вектор-столбец частных производных $\hat{q}(s, a, \mathbf{w})$ по $\mathbf{w}$
$\mathbf{x}(s)$	вектор признаков, видимых в состоянии s
$\mathbf{x}(s, a)$	вектор признаков, видимых, когда в состоянии s предпринимается действие a
$x_i(s), x_i(s, a)$	i -й элемент вектора $\mathbf{x}(s)$ или $\mathbf{x}(s, a)$
$\mathbf{x}_t$	сокращенное обозначение $\mathbf{x}(S_t)$ или $\mathbf{x}(S_t, A_t)$
$\mathbf{w}^T \mathbf{x}$	скалярное произведение векторов, $\mathbf{w}^T \mathbf{x} \doteq \sum_i w_i x_i$; например, $\hat{v}(s, \mathbf{w}) \doteq \mathbf{w}^T \mathbf{x}(s)$

$\mathbf{v}, \mathbf{v}_t$	вторичный d -мерный вектор весов, используемый для обучения $\mathbf{w}$ (глава 11)
$\mathbf{z}_t$	d -мерный вектор следов приемлемости в момент t (глава 12)
θ, θ_t	вектор параметров целевой стратегии (глава 13)
$\pi(a s, \theta)$	вероятность выбора действия a в состоянии s при условии параметрического вектора θ
π_θ	стратегия, соответствующая параметрам θ
$\nabla\pi(a s, \theta)$	вектор-столбец частных производных $\pi(a s, \theta)$ по θ
$J(\theta)$	мера качества для стратегии π_θ
$\nabla J(\theta)$	вектор-столбец частных производных $J(\theta)$ по θ
$h(s, a, \theta)$	предпочтение выбору действия a в состоянии s , основанное на θ
$b(a s)$	поведенческая стратегия, применяемая для выбора действий в процессе обучения целевой стратегии π
$b(s)$	базовая функция $b: \mathcal{S} \mapsto \mathbb{R}$ для методов градиента стратегии
b	коэффициент ветвления для МППР или дерева поиска
$\rho_{t:h}$	коэффициент выборки по значимости для временных шагов от t до h (раздел 5.5)
ρ_t	коэффициент выборки по значимости для одного только шага t , $\rho_t = \rho_{t:t}$
$r(\pi)$	среднее вознаграждение (коэффициент вознаграждения) для стратегии π (раздел 10.3)
$\bar{R}_t$	оценка $r(\pi)$ в момент t
$\mu(s)$	распределение состояний с единой стратегией (раздел 9.2)
μ	$ \mathcal{S} $ -мерный вектор $\mu(s)$ для всех $s \in \mathcal{S}$
$\ v\ _\mu^2$	μ -взвешенная норма функции ценности v , т. е. $\ v\ _\mu^2 \doteq \sum_s \mu(s)v(s)^2$ (раздел 11.4)
$\eta(s)$	ожидаемое количество посещений состояния s в одном эпизоде (стр. 240)
Π	оператор проекции для функций ценности (стр. 316)
B_π	оператор Беллмана для функций ценности (раздел 11.4)
$\mathbf{A}$	матрица $\mathbf{A} \doteq \mathbb{E}[\mathbf{x}_t(\mathbf{x}_t - \gamma\mathbf{x}_{t+1})^\top]$ размерности $d \times d$
$\mathbf{b}$	d -мерный вектор $\mathbf{b} \doteq \mathbb{E}[R_{t+1}\mathbf{x}_t]$
$\mathbf{w}_{\text{TD}}$	неподвижная точка TD $\mathbf{w}_{\text{TD}} \doteq \mathbf{A}^{-1}\mathbf{b}$ (d -мерный вектор, раздел 9.4)
$\mathbf{I}$	единичная матрица
$\mathbf{P}$	матрица $ \mathcal{S} \times \mathcal{S} $ вероятностей перехода состояний при стратегии π
$\mathbf{D}$	диагональная матрица $ \mathcal{S} \times \mathcal{S} $ со значениями μ на диагонали
$\mathbf{X}$	матрица $ \mathcal{S} \times d$ с векторами-строками $\mathbf{x}(s)$
$\overline{\text{VE}}(\mathbf{w})$	среднеквадратическая ошибка $\overline{\text{VE}}(\mathbf{w}) \doteq \ \mathbf{v}_w - \mathbf{v}_\pi\ _\mu^2$ (раздел 9.2)
$\bar{\delta}_w(s)$	беллмановская ошибка (математическое ожидание TD-ошибки), когда состоянием s является $\mathbf{v}_w$ (раздел 11.4)

$\bar{\delta}_{\mathbf{w}}, \text{BE}$	беллмановский вектор ошибок с элементами $\bar{\delta}_{\mathbf{w}}(s)$
$\overline{\text{BE}}(\mathbf{w})$	среднеквадратическая беллмановская ошибка $\overline{\text{BE}}(\mathbf{w}) \doteq \ \bar{\delta}_{\mathbf{w}}\ _{\mu}^2$
$\overline{\text{PBE}}(\mathbf{w})$	среднеквадратическая спроецированная беллмановская ошибка $\overline{\text{PBE}}(\mathbf{w}) \doteq \ \Pi \bar{\delta}_{\mathbf{w}}\ _{\mu}^2$
$\overline{\text{TDE}}(\mathbf{w})$	среднеквадратическая ошибка временных различий $\overline{\text{TDE}}(\mathbf{w}) \doteq \mathbb{E}_b[\rho_t \delta_t^2]$ (раздел 11.5)
$\overline{\text{RE}}(\mathbf{w})$	ошибка среднеквадратического дохода (раздел 11.6)

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии данной книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Список литературы

Список использованной литературы лежит на сайте dmkpress@gmail.com на странице книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и The MIT Press очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли применить санкции.

Пожалуйста, свяжитесь с нами по адресу dmkpress@gmail.com со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, помогающую нам предоставлять вам качественные материалы.

Глава 1

Введение

Идея о том, что мы обучаемся, взаимодействуя с окружающей средой, – вероятно, первое, что приходит в голову, когда мы задумываемся о природе обучения. Когда младенец играет, размахивает ручками или оглядывается вокруг, у него нет определенного учителя, однако имеется прямая сенсорно-двигательная связь со средой. Использование этой связи дает разнообразную информацию о причинах и следствиях, о последовательности действий и о том, что делать, чтобы достичь цели. На протяжении всей нашей жизни такие взаимодействия, несомненно, являются основным источником знаний об окружающей среде и о нас самих. Учимся ли мы водить машину или поддерживать беседу, мы всегда точно знаем, как среда реагирует на наши действия, и стремимся повлиять на происходящее с помощью своего поведения. Обучение посредством взаимодействия – фундаментальная идея, лежащая в основе почти всех теорий обучения и интеллекта.

В этой книге мы исследуем вычислительный подход к обучению посредством взаимодействия. Вместо того чтобы строить теории о том, как обучаются люди и животные, мы будем в основном рассматривать идеализированные ситуации и оценивать эффективность различных методов обучения¹. То есть займем позицию исследователя или конструктора искусственного интеллекта. Мы будем изучать конструкции машин, которые эффективно решают проблемы обучения, представляющие научный или экономический интерес, оценивать эти конструкции методами математического анализа или с помощью компьютерных экспериментов. Принятый нами подход называется *обучением с подкреплением*, он в гораздо большей степени ориентирован на целеустремленное обучение посредством взаимодействия, чем другие подходы к машинному обучению.

1.1. ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ

Обучение с подкреплением – это обучение тому, что делать, т. е. как отобразить ситуации на действия, чтобы максимизировать численный сигнал – вознаграждение. Обучаемому не говорят, какие действия предпринимать, он должен сам понять, какие действия приносят максимальное вознаграждение, пробуя их. В наиболее интересных и трудных случаях действия могут влиять не только на непосредственное вознаграждение, но и на следующую ситуацию, а значит, на все последующие вознаграждения. Эти две характеристики – поиск методом проб

¹ Связи с психологией и нейронауками излагаются в главах 14 и 15.

и ошибок и отложенное вознаграждение – являются наиболее важными отличительными чертами обучения с подкреплением.

Обучение с подкреплением, как и многие вещи, названия которых оканчиваются на «ние», например машинное обучение или скалолазание, одновременно является задачей, классом методов решения, хорошо работающих для этой задачи, и областью знаний, в которой изучается сама задача и методы ее решения. Удобно использовать одно название для всех трех вещей, понимая при этом, что концептуально они различны. В частности, различие между задачей и методами ее решения очень важно в обучении с подкреплением, и пренебрежение этим различием является источником множества недоразумений.

Мы формализуем задачу обучения с подкреплением, применяя идеи из теории динамических систем, а точнее как задачу оптимального управления не полностью известным марковским процессом принятия решений. Детали формализации подождут до главы 3, но основную мысль можно сформулировать просто – требуется уловить наиболее важные аспекты реальной проблемы, стоящей перед обучающимся агентом, который взаимодействует во времени с окружающей средой для достижения некоторой цели. Обучающийся агент должен уметь в какой-то степени воспринимать состояние среды и предпринимать действия, изменяющие это состояние. У агента также должна быть цель или несколько целей, как-то связанных с состоянием окружающей среды. Марковские процессы принятия решений включают все три аспекта – восприятие, действие и цель – в простейшей возможной форме, не сводя, однако, ни один аспект к тривиальному. Любой метод, подходящий для решения таких задач, будет рассматриваться нами как метод обучения с подкреплением.

Обучение с подкреплением отличается от *обучения с учителем*, еще одного вида обучения, который изучается в большинстве современных работ по машинному обучению. В случае обучения с учителем имеется обучающий набор помеченных примеров, подготовленный квалифицированным внешним учителем. Каждый пример представляет собой описание ситуации и спецификацию – метку – правильного действия, которое система должна предпринять в этой ситуации. Часто метка определяет категорию, которой принадлежит ситуация. Цель такого обучения – добиться, чтобы система смогла экстраполировать, или обобщить, свою реакцию на ситуации, которые не были предъявлены в обучающем наборе. Это важный вид обучения, но, взятый сам по себе, он не подходит для обучения с помощью взаимодействия. В интерактивных задачах часто практически невозможно получить примеры желаемого поведения, которые правильно представляли бы все ситуации, в которых агенту предстоит действовать. На неизведанной территории – там, где от обучения как раз и ожидают плодов, – агент должен уметь действовать, исходя из своего опыта.

Обучение с подкреплением отличается и от так называемого *обучения без учителя*, которое обычно имеет целью обнаружение структуры, скрытой в наборе немеченных данных. Кажется, что термины «обучение с учителем» и «обучение без учителя» исчерпывают все возможные парадигмы машинного обучения, однако это не так. Может возникнуть соблазнительная мысль о том, что обучение с подкреплением – разновидность обучения без учителя, поскольку отсутствуют примеры правильного поведения. Но в действительности цель обучения с подкреплением – максимизировать вознаграждение, а не выявить скрытую структуру

ру. Выявление структуры в опыте агента, конечно, может быть полезно, но само по себе не решает задачу максимизации вознаграждения, стоящую перед обучением с подкреплением. Поэтому мы считаем обучение с подкреплением третьей парадигмой машинного обучения наряду с обучением с учителем, без учителя и, возможно, еще какими-то парадигмами.

Один из вызовов, стоящих перед обучением с подкреплением, но отсутствующий в других видах обучения, – нахождение компромисса между исследованием и использованием. Чтобы получить большое вознаграждение, обучающийся с подкреплением агент должен предпочитать действия, которые были испытаны в прошлом и принесли вознаграждение. Но чтобы найти такие действия, он должен пробовать действия, которые раньше не выбирал. Агент должен *использовать* уже приобретенный опыт, чтобы получить вознаграждение, но должен *продолжать исследования*, чтобы выбирать более эффективные действия в будущем. Дилемма состоит в том, что одного лишь исследования или использования недостаточно для успешного решения задачи. Агент должен пробовать разные действия и неуклонно отдавать предпочтение тем, которые кажутся наилучшими. В стохастической задаче каждое действие необходимо испытать много раз, чтобы получить надежную оценку ожидаемого вознаграждения. Проблема исследования-использования интенсивно изучалась математиками на протяжении многих десятилетий, но и по сей день остается нерешенной. Пока что просто заметим, что вопрос о нахождении баланса между исследованием и использованием вообще не возникает в обучении с учителем и без учителя, по крайней мере в чистых вариантах этих парадигм.

Еще одна важная черта обучения с подкреплением – явное рассмотрение целостной проблемы целеустремленного агента, взаимодействующего с неопределенной окружающей средой. Этим оно отличается от многих подходов, в которых рассматриваются подзадачи, не задумываясь об их месте в общей картине. Например, мы упомянули, что во многих работах по машинному обучению изучается обучение с учителем, но явно не ставится вопрос о конечной пользе приобретенных способностей. Другие ученые разрабатывали теории планирования с целями общего вида, но не рассматривали роль планирования в принятии решений в режиме реального времени и не задавались вопросом, откуда возьмутся прогностические модели, необходимые для планирования. Хотя эти подходы принесли много полезных результатов, их сосредоточенность на изолированных подзадачах является серьезным ограничением.

В обучении с подкреплением принят противоположный подход – все начинается с полного интерактивного агента, преследующего некоторую цель. У всех обучающихся с подкреплением агентов имеются явные цели, все они могут воспринимать аспекты окружающей среды и выбирать действия, оказывающие влияние на среду. Более того, обычно с самого начала предполагается, что агент должен действовать, невзирая на значительную неопределенность окружающей его среды. Если обучение с подкреплением включает планирование, то оно должно учитывать взаимное влияние планирования и выбора действий в реальном времени, а также ответить на вопрос о том, откуда поступают и как совершенствуются модели. Если обучение с подкреплением включает обучение с учителем, то тому есть конкретные причины, которые определяют, какие способности критичны, а какие нет. Чтобы добиться прогресса в исследованиях по обучению, необ-

ходимо выделить и изучить важные подзадачи, но эти подзадачи должны играть понятные роли в полных интерактивных целеустремленных агентах, даже если детали полного агента еще только предстоит определить.

Под полным интерактивным целеустремленным агентом мы не всегда понимаем нечто вроде целостного организма или робота. Это возможные примеры, но полный интерактивный целеустремленный агент может быть также компонентом более крупной системы, наделенной определенным поведением. В таком случае агент прямо взаимодействует с остальными частями объемлющей системы и косвенно с окружением этой системы. Простой пример – агент, который следит за уровнем заряда батареи робота и посылает команды архитектуре управления роботом. Окружение этого агента – остальные части робота и окружение робота. Чтобы в полной мере оценить общность идеи обучения с подкреплением, нужно не ограничиваться очевидными примерами агентов.

Один из самых захватывающих аспектов современного обучения с подкреплением – его содержательные и плодотворные связи с другими инженерными и научными дисциплинами. Обучение с подкреплением – часть наблюдающейся уже несколько десятилетий тенденции к более тесной интеграции между искусственным интеллектом и машинным обучением, с одной стороны, и статистикой, оптимизацией и другими разделами математики – с другой. Например, способность некоторых методов обучения с подкреплением обучаться с помощью параметризованных аппроксиматоров решает классическую проблему «проклятия размерности» в исследовании операций и теории управления. Еще более заметна сильная связь обучения с подкреплением с психологией и нейронауками, приносящая ощутимые выгоды обеим сторонам. Из всех видов машинного обучения именно обучение с подкреплением ближе всего к способам обучения людей и животных, и истоки многих ключевых алгоритмов обучения с подкреплением следует искать в биологических самообучающихся системах. Обучение с подкреплением также вернуло долг – как в виде психологической модели обучения животных, лучше соответствующей некоторым эмпирическим данным, так и в виде влиятельной модели частей системы вознаграждения мозга. В этой книге развиваются идеи в основном обучения с подкреплением, относящиеся к конструированию и искусственному интеллекту, а связи с психологией и нейронауками описаны в главах 14 и 15.

Наконец, обучение с подкреплением – часть более широкой тенденции возвращения искусственного интеллекта к простым общим принципам. Начиная с конца 1960-х годов многие исследователи искусственного интеллекта предполагали, что нет никаких общих принципов, а интеллект обязан своим появлением разнообразным специальным приемам, процедурам и эвристикам. Иногда высказывалось мнение, что если бы мы могли заложить в машину достаточно фактов, скажем миллион или миллиард, то она обрела бы интеллект. Методы, основанные на общих принципах, таких как поиск или обучение, объявлялись «слабыми методами», тогда как основанные на специальных знаниях – «сильными методами». Эта точка зрения распространена и сегодня, но не является доминирующей. На наш взгляд, она просто была преждевременной: слишком мало усилий было вложено в поиск общих принципов, чтобы делать вывод об их отсутствии. В современных работах по искусственному интеллекту много внимания уделяется общим принципам обучения, поиска и принятия решений. Не ясно, насколько далеко назад

качнулся маятник, но обучение с подкреплением – безусловно, часть этого возвратного движения к простым и немногочисленным общим принципам искусственного интеллекта.

1.2. ПРИМЕРЫ

Чтобы понять суть обучения с подкреплением, полезно рассмотреть несколько примеров и возможных приложений, которые стали стимулами для его разработки.

- Мастер-шахматист делает ход. Выбор продиктован планированием – предвидением возможных ответов и продолжений – и непосредственными интуитивными оценками желательности конкретных позиций и ходов.
- Адаптивный контроллер в режиме реального времени подстраивает параметры работы нефтеперерабатывающего завода. Контроллер оптимизирует соотношение между выходом готовой продукции, стоимостью и качеством, основываясь на заданных предельных затратах и не придерживаясь строго штатных режимов, предложенных конструкторами.
- Детеныш газели с трудом встает на ножки через несколько минут после рождения. Спустя полчаса он уже мчится со скоростью 20 миль в час.
- Подвижный робот решает, нужно ли ему войти в новое помещение для уборки мусора или уже пора искать станцию подзарядки. Он принимает решение, исходя из текущего уровня заряда батареи и того, насколько легко и быстро ему удавалось найти зарядное устройство в прошлом.
- Фил готовит завтрак. При ближайшем рассмотрении даже в этой, казалось бы, рутинной деятельности обнаруживается сложная паутина условных типов поведения и взаимных связей между целью и подцелями: подойти к кухонному шкафчику, открыть его, выбрать коробку с хлопьями, потянуться за ней, ухватить и вытащить. Другая не менее сложная, зависящая от внешних условий интерактивная последовательность действий необходима, чтобы достать тарелку, ложку и пачку молока. На каждом шаге мы наблюдаем многочисленные движения глаз для получения информации и управления передвижениями. Все время принимаются быстрые решения о том, как перемещать предметы и нужно ли поставить их на стол, перед тем как доставать другие. Каждый шаг диктуется целями, например взять ложку или открыть холодильник, и, в свою очередь, служит для достижения других целей, например иметь в распоряжении ложку, когда хлопья будут готовы, и в конечном итоге насытиться. Сознательно или нет, Фил получает информацию о состоянии собственного тела, на основании которой определяет, пора ли поесть, насколько он проголодался и что хочет покушать.

У этих примеров есть одна общая черта: они настолько привычны, что на них легко не обратить внимания. В каждом из них имеется *взаимодействие* между активным агентом, принимающим решения, и окружающей его средой, находясь внутри которой, агент стремится достичь *цели*, несмотря на *недетерминированность* среды. Действия агента могут влиять на будущее состояние среды (например, следующую позицию в шахматной партии, уровень заполнения хранилищ НПЗ, следующее местоположение робота и будущий уровень заряда его батареи) и тем самым на действия агента и доступные ему возможности в будущие момен-

ты времени. Для правильного выбора необходимо учитывать косвенные, отложенные последствия действий, поэтому может потребоваться прогнозирование или планирование.

При этом во всех рассмотренных примерах последствия действий невозможно предсказать достоверно; поэтому агент должен часто наблюдать за окружающей средой и действовать соответственно. Например, Фил должен следить, сколько молока он налил в тарелку с хлопьями, чтобы оно не пролилось на стол. Во всех примерах цели определены явно в том смысле, что агент может судить о продвижении к цели, полагаясь на свое непосредственное восприятие. Шахматист знает, выигрывает он или нет, контроллер НПЗ знает, сколько бензина произведено, детеныш газели знает, что упал, подвижный робот знает, когда заряд батареи подходит к концу, а Фил знает, завтракает он уже или нет.

Во всех приведенных примерах агент может использовать свой опыт для повышения результативности действий в будущем. Шахматист оттачивает интуицию при оценке позиций, а значит, улучшает свою игру; детеныш газели увеличивает эффективность бега; Фил учится не делать лишних движений во время приготовления завтрака. От знаний, с которыми агент приступает к выполнению задачи, – полученных из предыдущего опыта решения аналогичных задач или встроенных в него конструкторами или эволюцией, – зависит, чему полезно или легко обучиться, но только взаимодействие с окружающей средой позволит подкорректировать поведение для использования конкретных особенностей задачи.

1.3. ЭЛЕМЕНТЫ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

Помимо агента и окружающей среды, можно выделить четыре главных элемента системы обучения с подкреплением: *стратегия*, *сигнал вознаграждения*, *функция ценности* и, факультативно, *модель* окружающей среды.

Стратегия определяет, как обучающийся агент поведет себе в данный момент времени. Грубо говоря, стратегия – это отображение множества воспринимаемых состояний среды на действия, предпринимаемые в этих состояниях. Она соответствует тому, что в психологии называется множеством правил, или ассоциаций, стимул–реакция. В некоторых случаях стратегия может быть простой функцией или таблицей соответствия, тогда как в других требует большого объема вычислений, например выполнения поиска. Стратегия лежит в основе обучающегося с подкреплением агента, поскольку ее одной достаточно для определения поведения. В общем случае стратегии могут быть стохастическими, т. е. задавать вероятности каждого действия.

Сигнал вознаграждения определяет цель в задаче обучения с подкреплением. На каждом временном шаге среда посылает обучающемуся агенту одно число, называемое *вознаграждением*. Единственное стремление агента – максимизировать полное вознаграждение, полученное в течение длительного времени работы. Таким образом, сигнал вознаграждения определяет, что для агента хорошо, а что плохо. В биологической системе аналогами вознаграждения являются удовольствие или боль. Вознаграждения – прямые и определяющие характеристики проблемы, стоящей перед агентом. Сигнал вознаграждения – главная причина изменения стратегии; если выбранное стратегией действие влечет низкое возна-

граждение, то стратегию следует изменить, так чтобы в будущем в такой ситуации выбиралось другое действие. В общем случае сигнал вознаграждения может быть стохастической функцией состояния среды и предпринятого действия.

Если сигнал вознаграждения показывает, что хорошо прямо сейчас, то *функция ценности* говорит, что хорошо в длительной перспективе. Грубо говоря, ценность состояния – это полное вознаграждение, которого агент может ожидать в будущем, если начнет работу в этом состоянии. Вознаграждение определяет непосредственную внутренне присущую желательность состояний окружающей среды, а ценность – долговременную желательность состояний с учетом тех состояний, которые с большой вероятностью встретятся позже, и вознаграждений в этих состояниях. Например, состояние может всегда приносить низкое немедленное вознаграждение, но при этом иметь высокую ценность, поскольку за ним регулярно следуют состояния, приносящие высокое вознаграждение. Обратное тоже может иметь место. Проводя аналогию с человеком, мы можем уподобить вознаграждение удовольствию (если оно высокое) или неудовольствию (если низкое), а ценность соответствует более продуманному и прозорливому суждению о том, насколько мы довольны или недовольны конкретным состоянием окружающей нас среды.

В некотором смысле вознаграждение первично, тогда как ценность, будучи прогнозом вознаграждения, вторична. Без вознаграждения не было бы ценности, а единственный смысл оценки ценности – получить большее вознаграждение. Тем не менее именно ценность стоит на первом месте, когда мы принимаем решения и оцениваем их последствия. Действия выбираются, исходя из суждений о ценности. Мы ищем те действия, которые приводят в состояния с наибольшей ценностью, а не те, которые приносят наибольшее вознаграждение, поскольку именно первые обещают максимальное вознаграждение в длительной перспективе. Вознаграждение мы получаем от среды непосредственно, а ценность необходимо оценивать и переоценивать, соотносясь с последовательностями наблюдений, сделанных агентом за все время существования. На самом деле важнейшая часть почти всех рассматриваемых алгоритмов обучения с подкреплением – метод эффективного оценивания ценности. Центральная роль оценивания ценности – это, пожалуй, самое важное, что мы узнали об обучении с подкреплением за последние шестьдесят лет.

Четвертый и последний элемент некоторых систем обучения с подкреплением – *модель* окружающей среды. Это то, что имитирует поведение окружающей среды или, более общо, то, что позволяет делать выводы о том, как поведет себя среда. Например, зная состояние и действие, модель могла бы предсказать следующее состояние и следующее вознаграждение. Модели используются для *планирования*, под которым мы понимаем любой способ выбора порядка действий путем рассмотрения возможных будущих ситуаций, до того как они фактически произошли. Методы решения задач обучения с подкреплением, в которых используются модели и планирование, называются *основанными на модели*, в отличие от более простых *безмодельных* методов, в которых обучаемый явно действует методом проб и ошибок, считаясь чуть ли не полной *противоположностью* планированию. В главе 8 мы будем рассматривать системы обучения с подкреплением, которые одновременно обучаются методом проб и ошибок, в результате обучения строят модель окружающей среды и используют эту модель для планирования. Современное обучение с подкреплением охватывает весь спектр си-

стем – от низкоуровневого обучения методом проб и ошибок до высокоуровневого обоснованного планирования.

1.4. ОГРАНИЧЕНИЯ И КРУГ ВОПРОСОВ

Обучение с подкреплением в значительной мере опирается на понятие состояния – оно служит входной информацией для стратегии и функции ценности, а также входной и выходной информацией модели. Неформально можно считать состояние сигналом, доносящим до агента представление о том, как «выглядит окружающая среда» в конкретный момент времени. Формальное определение состояния дается в контексте марковских процессов принятия решения в главе 3. Но вообще мы призываем читателя опираться на неформальный смысл и рассматривать состояние как любую доступную агенту информацию об окружающей его среде. По существу, мы предполагаем, что сигнал состояния порождается какой-то системой предобработки, номинально являющейся частью окружающей агента среды. В этой книге мы не затрагиваем вопросы конструирования, изменения и обучения сигналу состояния (разве что очень кратко в разделе 17.3). Такой подход принят не потому, что мы считаем представление состояния чем-то несущественным, а чтобы полностью сосредоточиться на вопросах принятия решений. Иными словами, в этой книге нас интересует не конструирование сигнала состояния, а решение о том, какое действие предпринять при данном сигнале состояния.

Большинство рассматриваемых в книге методов обучения с подкреплением построено на оценивании функций ценности, но, вообще говоря, это необязательно. Например, в таких методах, как генетические алгоритмы, генетическое программирование, имитация отжига и другие методы оптимизации, функция ценности вообще не оценивается. В этих методах применяется несколько статических стратегий, каждая из которых на протяжении длительного времени взаимодействует со своим экземпляром среды. Стратегии, принесшие максимальное вознаграждение, и их случайные вариации, переходят в следующее поколение стратегий, после чего процесс повторяется. Мы называем такие методы *эволюционными*, потому что их работа напоминает то, как в результате биологической эволюции появляются организмы, обладающие осмысленным поведением, хотя они не обучались ему на протяжении собственной жизни. Если пространство стратегий достаточно мало или может быть организовано так, что хорошие стратегии встречаются часто или легко находятся (или если их поиску можно уделить много времени), то эволюционные методы могут оказаться эффективными. Кроме того, эволюционные методы имеют преимущества в задачах, где обучающийся агент не может воспринять полное состояние окружающей среды.

Наше внимание приковано к методам обучения с подкреплением, которые обучаются, взаимодействуя со средой, чего эволюционные методы не умеют. Методы, способные извлекать пользу из деталей поведения в отдельных актах взаимодействия, во многих случаях оказываются гораздо эффективнее эволюционных методов. Эволюционные методы игнорируют значительную часть полезной структуры, присутствующей в задаче обучения с подкреплением: они не используют тот факт, что искомая стратегия является функцией, отображающей состояния в действия; они не замечают, через какие состояния проходит индивидуум

на протяжении своей жизни и какие действия он выбирает. В некоторых случаях эта информация может сбивать с толку (например, когда состояния восприняты неправильно), но чаще повышают эффективность поиска. Хотя эволюция и обучение имеют много общего и естественно работают рука об руку, мы не считаем, что эволюционные методы сами по себе хорошо подходят для задач обучения с подкреплением, и потому не рассматриваем их в этой книге.

1.5. РАЗВЕРНУТЫЙ ПРИМЕР: ИГРА В КРЕСТИКИ-НОЛИКИ

Чтобы проиллюстрировать общую идею обучения с подкреплением и сравнить ее с другими подходами, мы подробно рассмотрим один пример.

Возьмем знакомую с детства игру в крестики-нолики. Два игрока по очереди делают ходы на доске 3×3 . Один игрок ставит крестики (X), другой – нолики (O), до тех пор пока три одинаковых значка не выстроятся по горизонтали, по вертикали или по диагонали. Если доска заполнена, но трех значков подряд нет, считается, что игра закончилась вничью. Поскольку опытный игрок всегда может избежать проигрыша, предположим, что мы играем против неумехи, который иногда делает неправильные ходы и дает нам возможность выиграть. На некоторое время будем считать, что проигрыш и ничья одинаково плохи для нас. Как сконструировать игрока, который будет находить изъяны в игре противника и обучаться максимизировать свои шансы на выигрыш?

X	O	O
O	X	X
		X

Это простая задача, но и ее трудно решить, применяя классические методы. Например, классическое «минимаксное» решение из теории игр не годится, потому что в нем предполагается, что противник играет определенным способом. Так, минимаксный игрок никогда не перейдет в состояние игры, в котором мог бы проиграть, даже если в действительности он всегда выигрывает в этом состоянии из-за неправильной игры противника. Классические методы оптимизации для задач с последовательным принятием решений, например динамическое программирование, могут вычислить оптимальное решение для любого противника, но требуют, чтобы на вход было подано полное описание противника, в частности вероятности выбора им каждого хода в любом состоянии на доске. Предположим, что эта информация заранее недоступна, именно так обстоит дело в большинстве задач, представляющих практический интерес. С другой стороны, ее можно оценить на основе опыта, сыграв много игр с противником. Едва ли не лучшее, что можно сделать в этой задаче, – сначала обучить модель поведения противника с некоторым уровнем доверия, а затем применить динамическое программирование для нахождения оптимального решения при данной приближенной модели противника. В конечном итоге это не так уж сильно отличается от некоторых методов обучения с подкреплением, которые мы будем изучать в данной книге.

Применение эволюционного метода к этой задаче означало бы, что мы должны произвести прямой поиск в пространстве возможных стратегий в попытке найти стратегию с высокой вероятностью выигрыша у противника. Здесь стратегией является правило, сообщающее игроку, какой ход сделать в каждом состоянии игры,

т. е. любой из возможных конфигураций крестиков и ноликов на доске 3×3 . Для каждой рассматриваемой стратегии нужно было бы получить оценку вероятности выигрыша, сыграв несколько партий с противником. В результате такого вычисления мы получили бы указание, какую стратегию или стратегии рассматривать на следующем шаге. Типичный эволюционный метод осуществлял бы «восхождение на гору» в пространстве стратегий, последовательно генерируя и оценивая стратегии в попытке добиться постепенного улучшения. Или можно было бы использовать генетический алгоритм, который хранил бы и оценивал популяцию стратегий. В общем, можно было бы применить буквально сотни различных методов оптимизации.

А вот как к игре в крестики-нолики можно было бы применить метод, в котором используется функция ценности. Сначала нужно было бы подготовить таблицу чисел, по одному для каждого возможного состояния игры. Каждое число было бы равно последней оценке вероятности выиграть, начав с этого состояния. Эту оценку мы считаем ценностью состояния, а всю таблицу – обученной функцией ценности. Состояние А ценнее, или «лучше», состояния В, если текущая оценка вероятности выигрыша в А больше, чем в В. Предположим, что мы всегда играем крестиками, тогда для всех состояний с тремя X подряд вероятность выигрыша равна 1, поскольку мы уже выиграли. Аналогично для всех состояний с тремя O подряд, а также для заполненной доски вероятность выигрыша равна 0, поскольку в них мы выиграть не можем. Начальную ценность всех остальных состояний мы полагаем равной 0.5, т. е. мы думаем, что шанс выиграть составляет 50 %.

Затем мы играем много игр с противником. Для выбора хода мы рассматриваем состояния, которые получаются в результате каждого возможного хода (их столько, сколько пустых позиций на доске), и ищем их ценности в таблице. По большей части мы делаем ход жадно, т. е. выбираем такой ход, который ведет в состояние с наибольшей ценностью, т. е. с наибольшей оценкой вероятности выигрыша. Но иногда мы выбираем ход случайно. Такие ходы называются разведочными, поскольку приводят в состояния, которые мы иначе могли бы никогда не посетить. Последовательность сделанных и рассмотренных ходов за всю игру можно наглядно изобразить, как показано на рис. 1.1.

В процессе игры мы изменяем ценность состояний, в которых оказывались. Мы хотим, чтобы они более точно оценивали вероятность выигрыша. Для этого мы «переносим» ценность состояния после каждого жадного хода в состояние до этого хода, как показывают стрелки на рис. 1.1. Точнее, текущая ценность предыдущего состояния обновляется, так чтобы стать ближе к ценности следующего состояния. Это можно сделать, немного сдвинув ценность предыдущего состояния в направлении ценности следующего состояния. Если обозначить S_t состояние до жадного хода, а S_{t+1} – состояние после этого хода, то обновление оценки ценности S_t , обозначаемой $V(S_t)$, можно записать в виде:

$$V(S_t) \leftarrow V(S_t) + \alpha[V(S_{t+1}) - V(S_t)],$$

где α – небольшой положительный коэффициент, называемый параметром *размера шага*, который влияет на скорость обучения. Это правило обновления – пример метода обучения на основе *временных различий*. Он называется так, потому что изменения зависят от разности $V(S_{t+1}) - V(S_t)$ между оценками в соседние моменты времени.

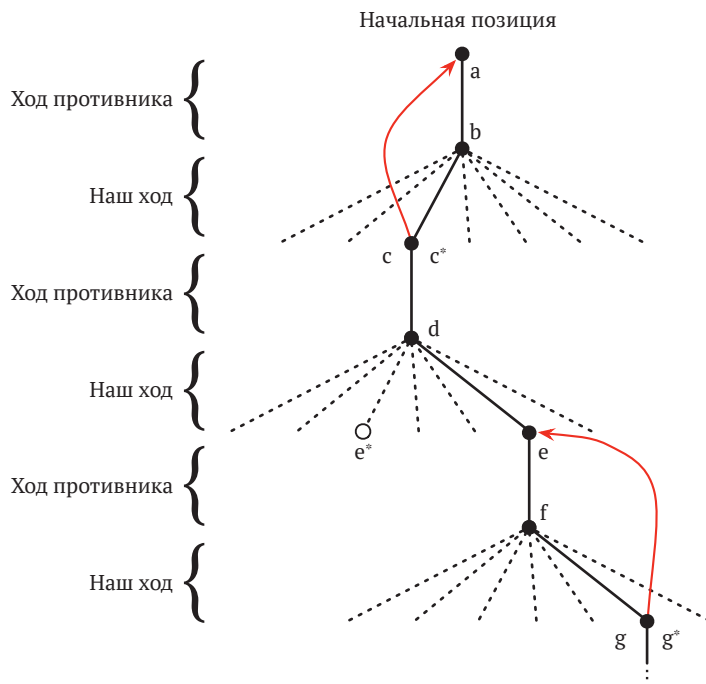


Рис. 1.1 ❖ Последовательность ходов в игре в крестики-нолики. Сплошными черными линиями показаны ходы, сделанные в игре; пунктирными линиями – ходы, которые мы (наш игрок, обучаемый с подкреплением) рассматривали, но не сделали. Второй ход был разведочным, т. е. он был сделан, несмотря на то что альтернативный ход (ведущий в состояние e^*) оценивался выше. Разведочные ходы никак не сказываются на обучении, но все остальные ходы сказываются, а именно приводят к обновлениям, показанным красными линиями: оценки значений поднимаются вверх по дереву от более поздних узлов к более ранним, как объяснено в тексте

Описанный выше метод дает хорошие результаты на этой задаче. Так, если размер шага подходящим образом уменьшать со временем, то для любого фиксированного противника этот метод сходится к истинным вероятностям выигрыша в каждом состоянии при условии оптимальной игры нашим игроком. Более того, сделанные ходы (за исключением разведочных) действительно являются оптимальными ходами в игре против данного (неидеального) противника. Иными словами, метод сходится к оптимальной стратегии игры с данным противником. Если размер шага не уменьшается со временем до нуля, то этот игрок будет хорошо играть и с противниками, которые медленно изменяют манеру игры.

Этот пример иллюстрирует различие между эволюционными методами и методами, обучающими функцию ценности. Для вычисления стратегии эволюционный метод фиксирует стратегию и играет много игр с противником либо имитирует много игр с моделью противника. Частота выигрышей дает несмещенную оценку вероятности выиграть при данной стратегии и может использоваться при выборе следующей стратегии. Но любое изменение стратегии производится только после большого числа сыгранных игр, и используется лишь конечный результат каждой игры; что происходило в процессе игры, игнорируется. Например, если

игрок выиграл, то *все* его поведение в игре считается удачным независимо от того, насколько важными для выигрыша оказались отдельные ходы. Удачными считаются даже ходы, которые никогда не были сделаны! Напротив, методы на основе функции ценности позволяют оценивать отдельные состояния. В конечном итоге те и другие методы производят поиск в пространстве стратегий, но при обучении функции ценности задействуется информация, доступная по ходу игры.

Этот простой пример иллюстрирует некоторые ключевые особенности методов обучения с подкреплением. Во-первых, упор делается на обучение во взаимодействии с окружающей средой, в данном случае с противником по игре. Во-вторых, имеется ясная цель, и для выработки правильного поведения необходимо планирование или прогнозирование, учитывающее отложенные последствия выбора. Например, простой обучающийся с подкреплением игрок мог бы научиться расставлять многоходовые ловушки для близорукого противника. Поражительной особенностью обучения с подкреплением является тот факт, что оно может обеспечить эффект планирования и заглядывания в будущее, не используя модель противника и не выполняя явный поиск по всем возможным последовательностям будущих состояний и действий.

Хотя этот пример и иллюстрирует некоторые ключевые особенности обучения с подкреплением, он настолько прост, что может создаться впечатление, будто возможности обучения с подкреплением более ограничены, чем на самом деле. В крестики-нолики играют два человека, но обучение с подкреплением применимо и в случае, когда не существует внешнего противника, как, например, в «игре против природы». Кроме того, обучение с подкреплением не ограничивается задачами, в которых поведение распадается на отдельные эпизоды, как, например, серия игр в крестики-нолики, когда вознаграждение выплачивается только в конце каждого эпизода. В равной мере оно применимо и тогда, когда поведение не ограничено во времени, а вознаграждение разной величины можно получать в любой момент. Обучение с подкреплением применимо и к задачам, которые вообще не распадаются на дискретные временные шаги, как крестики-нолики. Общие принципы действуют и для непрерывных задач, хотя теория становится сложнее, и в этом введении мы ее опустим.

В крестиках-ноликах конечное и относительно небольшое множество состояний, а обучение с подкреплением можно использовать, и когда множество состояний очень велико и даже бесконечно. Например, в работах Gerry Tesauro (1992, 1995) описанный выше алгоритм объединен с искусственной нейронной сетью для игры в нарды, где количество состояний порядка 10^{20} . При таком числе состояний экспериментально исследовать можно только небольшую часть. Программа Тезауро научилась играть гораздо лучше всех предшествующих и в конце концов превзошла лучших в мире игроков-людей (см. раздел 16.1). Нейронная сеть наделяет программу способностью обобщать полученный опыт, так что в новых состояниях она выбирает ходы, опираясь на сохраненную информацию о похожих (с точки зрения сети) состояниях, наблюдавшихся в прошлом. Насколько хорошо система обучения с подкреплением может работать в задачах с таким обширным множеством состояний, сильно зависит от того, как успешно она обобщает предшествующий опыт. Именно для этого нам так нужно сочетание методов обучения с учителем и обучения с подкреплением. Искусственные нейронные сети и глубокое обучение (раздел 9.7) – не единственный и необязательно лучший способ добиться этого.

В примере с крестиками-ноликами обучение начиналось без какой-либо априорной информации, помимо знаний о правилах игры, но обучение с подкреплением ни в коем случае не подразумевает взгляд на обучение и интеллект как на чистую доску. Напротив, априорную информацию можно включить в процесс обучения разными способами, от которых критически зависит эффективность обучения (см., например, разделы 9.5, 17.4 и 13.1). В крестиках-ноликах у нас был доступ к истинному состоянию, но обучение с подкреплением применимо и в случае, когда часть состояния скрыта или когда различные состояния воспринимаются обучаемым как одно.

Наконец, в крестиках-ноликах игрок может заглянуть вперед и узнать, какие состояния станут результатом каждого из возможных ходов. Для этого ему нужна модель игры, позволяющая предвидеть, как изменится среда в ответ на ходы, которые он, возможно, никогда не сделает. Таких задач много, но встречаются и другие, когда отсутствует даже краткосрочная модель последствий действий. Обучение с подкреплением применимо в обоих случаях. Модель необязательна, но если она имеется или может быть обучена, то ее легко использовать (глава 8).

С другой стороны, существуют методы обучения с подкреплением, которые вообще не нуждаются в модели окружающей среды. Безмодельные системы даже помыслить не могут о том, как изменится среда в ответ на одиночное действие. В этом смысле игрок в крестики-нолики является безмодельной системой по отношению к противнику: у него нет никакой модели противника. Поскольку модель полезна, только если достаточно верна, то безмодельные методы могут иметь преимущество над более сложными в случаях, когда трудность решения задачи связана именно со сложностью построения достаточно точной модели окружающей среды. Кроме того, безмодельные методы часто являются важными структурными элементами методов, основанных на модели. В этой книге мы посвятим несколько глав безмодельным методам и только потом обсудим, как использовать их в качестве компонентов более сложных методов на основе моделей.

Обучение с подкреплением можно использовать как на верхних, так и на нижних уровнях системы. В крестиках-ноликах игрок обучился лишь простейшим ходам, но ничто не мешает включить обучение с подкреплением в верхние уровни, где само «действие» заключается в применении некоторого, возможно, изощренного метода решения задач. В иерархических самообучающихся системах обучение с подкреплением может работать сразу на нескольких уровнях.

Упражнение 1.1. Игра с самим собой. Предположим, что вместо игры со случайным противником описанный выше алгоритм обучения с подкреплением играет против себя самого, и обе стороны обучаются. Как вы думаете, что произойдет в таком случае? Обучится ли игрок другой стратегии выбора ходов? □

Упражнение 1.2. Симметрии. Многие позиции в крестиках-ноликах выглядят различными, но на самом деле отличаются только симметрией. Как можно изменить описанный выше процесс обучения, чтобы воспользоваться этим фактом? Как это изменение могло бы улучшить процесс обучения? А теперь подумайте еще раз. Предположим, что противник не пользуется преимуществами симметрии. Должны ли тогда это делать мы? И верно ли, что позиции, совпадающие с точностью до симметрии, имеют одну и ту же ценность? □

Упражнение 1.3. Жадная игра. Предположим, что обучающийся с подкреплением игрок жадный, т. е. всегда выбирает ход, который переводит его в позицию с наивысшим рейтингом. Мог бы он научиться играть лучше нежадного игрока? Или хуже? Какие проблемы могли бы возникнуть? □

Упражнение 1.4. Обучение на результатах разведки. Предположим, что обучение подвергается обновлению после *любого* хода, включая разведочные. Если со временем размер шага уменьшается подходящим образом (но не уменьшается стремление к исследованию), то ценности состояний будут сходиться к другому набору вероятностей. Что концептуально представляют собой два набора вероятностей, вычисленных, когда мы обучаемся и не обучаемся на разведочных ходах? В предположении, что мы продолжаем совершать разведочные ходы, какой набор вероятностей предпочтительнее обучить? Какой принесет больше выигрышей? □

Упражнение 1.5. Другие улучшения. Можете ли вы предложить иные способы улучшить игрока, обучающегося с подкреплением? Можете ли вы придумать лучший способ решения задачи об игре в крестики-нолики в том виде, в котором она представлена?

1.6. РЕЗЮМЕ

Обучение с подкреплением – вычислительный подход к пониманию и автоматизации обучения и принятия решений, направляемых стремлением к достижению цели. Оно отличается от других вычислительных подходов упором на обучение агента в процессе прямого взаимодействия с окружающей средой, без посредничества учителя и без полной модели среды. По нашему мнению, обучение с подкреплением – первая дисциплина, в которой серьезно изучаются вычислительные проблемы, возникающие при обучении в процессе взаимодействия с окружающей средой ради достижения долгосрочных целей.

В обучении с подкреплением используется формализм марковских процессов принятия решений для описания взаимодействия обучающегося агента со средой в терминах состояний, действий и вознаграждений. Этот формализм задуман как простой способ представить существенные особенности проблемы искусственного интеллекта, к которым относятся восприятие причин и следствий, восприятие неопределенности и недетерминированности и существование явных целей.

Понятия ценности и функции ценности – ключ к большинству методов обучения с подкреплением, рассматриваемых в этой книге. Мы считаем, что функции ценности важны для эффективного поиска в пространстве стратегий. Применение функций ценности отличает методы обучения с подкреплением от эволюционных методов, которые производят поиск непосредственно в пространстве стратегий, руководствуясь оценкой стратегий в целом.

1.7. ИСТОРИЯ РАННИХ ЭТАПОВ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

В ранней истории обучения с подкреплением есть два основных направления, которые долго и плодотворно развивались независимо, прежде чем переплелись

в современном обучении с подкреплением. Одно направление связано с обучением методом проб и ошибок, оно уходит корнями в психологию обучения животных. Это направление можно проследить в самых ранних работах по искусственному интеллекту, и оно привело к возрождению обучения с подкреплением в начале 1980-х годов. Второе направление связано с задачей оптимального управления и ее решением с помощью функций ценности и динамического программирования. По большей части оно не соприкасается с обучением. Оба направления были практически независимы, но оказались в какой-то мере взаимосвязаны в третьем, не столь отчетливо выделяющемся направлении, связанном с методами на основе временных различий типа того, что мы применили в примере с крестиками и ноликами. Все три направления сошлись в конце 1980-х годов, когда сформировалась современная дисциплина обучения с подкреплением в том виде, в каком она представлена в этой книге.

Направление, ставящее во главу угла обучение методом проб и ошибок, – то, с чем мы лучше всего знакомы и о чем будет больше всего сказано в этом кратком историческом очерке. Но прежде все-таки посвятим несколько слов оптимальному управлению.

Термин «оптимальное управление» вошел в обиход в конце 1950-х годов и применялся для описания задачи о проектировании устройства управления, которое должно было минимизировать или максимизировать некоторую характеристику поведения динамической системы во времени. Один из подходов к решению этой задачи был разработан в середине 1950-х годов Ричардом Беллманом и другими учеными путем обобщения теории Гамильтона–Якоби, созданной в XIX веке. В этом подходе понятия состояния динамической системы и функции ценности, или «оптимальной функции выгоды», используются для вывода функционального уравнения, которое теперь часто называют уравнением Беллмана. Класс методов решения задач оптимального управления путем решения этого уравнения называется динамическим программированием (Bellman, 1957a). В работе Bellman (1957b) описана также дискретная стохастическая версия задачи оптимального управления, известная под названием «марковский процесс принятия решений» (МППР, англ. MDP). В работе Ronald Howard (1960) предложен метод итерации по стратегиям для МППР. Все это – существенные элементы, лежащие в основе теории и алгоритмов современного обучения с подкреплением.

Общепризнано, что динамическое программирование – единственный практически применимый способ решения общих стохастических задач оптимального управления. Оно страдает от того, что Беллман называл «проклятием размерности», т. е. требования к вычислительной мощности растут экспоненциально с ростом числа переменных состояния. Но все равно оно гораздо более эффективно и распространено, чем любой другой общий метод. Тема динамического программирования активно разрабатывалась с конца 1950-х годов, были предложены обобщения на частично наблюдаемые МППР (обзор см. в работе Lovejoy, 1991), многочисленные приложения (обзор см. в работах White, 1985, 1988, 1993), приближенные методы (обзор см. в работе Rust, 1996) и асинхронные методы (Bertsekas, 1982, 1983). Есть много отличных учебников по динамическому программированию (например, Bertsekas, 2005, 2012; Puterman, 1994; Ross, 1983; Whittle, 1982, 1983). В работе Bryson (1996) подробно изложена история оптимального управления.

Наличие связей между оптимальным управлением и динамическим программированием, с одной стороны, и обучением – с другой, осознавалось медленно. Мы не уверены, откуда взялось такое разделение, но, вероятно, основная причина – в разрыве между соответствующими дисциплинами и различии их целей. Возможно, свой вклад внесло преобладающее представление о динамическом программировании как пакетном методе вычислений, который сильно зависит от наличия точной модели системы и аналитических решений уравнения Беллмана. К тому же простейшая форма динамического программирования – вычисление, происходящее в обратном направлении по времени, поэтому трудно понять, как его можно применить в процессе обучения, который по необходимости протекает в прямом направлении. Про некоторые из самых первых работ по динамическому программированию, например Bellman and Dreyfus (1959), сегодня можно сказать, что они находятся в русле подхода к обучению. Работа Witten (1977), обсуждаемая ниже, точно может быть классифицирована как сочетание идей обучения и динамического программирования. В работе Werbos (1987) приводятся явные аргументы в пользу более тесной связи между динамическим программированием и методами обучения и доказывается, что динамическое программирование имеет прямое отношение к пониманию работы нейронов и когнитивных механизмов. Для нас полное слияние методов динамического программирования с онлайн-обучением случилось только после знакомства с работой Криса Уоткинса (Chris Watkins) 1989 года, в которой обучение с подкреплением излагалось с позиций формализма МППР. С тех пор эти связи активно разрабатывались многими исследователями, в особенности Димитрием Бертсеркасом и Джоном Цициклисом (Dimitri Bertsekas and John Tsitsiklis, 1996), которые предложили термин «нейродинамическое программирование», описывающий комбинацию динамического программирования с нейронными сетями. Еще один термин, широко употребляемый в настоящее время, – «приближенное динамическое программирование». В каждом из этих подходов на первое место выдвигаются различные грани предмета, но все они разделяют с обучением с подкреплением интерес к преодолению классических недостатков динамического программирования.

Мы считаем, что все работы по оптимальному управлению в какой-то мере являются работами по обучению с подкреплением. Мы определяем метод обучения с подкреплением как любой эффективный способ решения задач обучения с подкреплением, и теперь понятно, что эти задачи тесно связаны с задачами оптимального управления, особенно со стохастическими, подобными формулируемым в терминах МППР. Соответственно, мы должны считать методы решения задач оптимального управления, например динамическое программирование, также методами обучения с подкреплением. Поскольку почти все традиционные методы требуют полного знания об управляемой системе, кажется не вполне естественным говорить, что они часть обучения с подкреплением. С другой стороны, многие алгоритмы динамического программирования инкрементные и итеративные. Как и методы обучения, они постепенно приходят к правильному ответу путем последовательных приближений. Далее в этой книге мы покажем, что это отнюдь не поверхностное сходство. Теории и методы решения для случаев полного и неполного знания настолько тесно связаны, что, по нашему мнению, они должны рассматриваться вместе как части одного предмета.

А теперь вернемся к другому направлению, которое привело к современному обучению с подкреплением, – направлению, в основе которого лежит идея обучения методом проб и ошибок. Здесь мы затронем только основные положения, а более детальное рассмотрение отложим до раздела 14.3. Согласно американскому психологу Р. С. Вудворту (R. S. Woodworth, 1938), идея обучения методом проб и ошибок восходит к трудам Александра Бэна (Alexander Bain) 1850-х годов, в которых обсуждалось обучение «методом поисков и находок вслепую», а еще точнее – к британскому этологу и психологу Конвею Ллойд Моргану (Conway Lloyd Morgan), который в 1894 году употребил этот термин для описания своих наблюдений за поведением животных. Быть может, первым, кто кратко выразил существо обучения методом проб и ошибок, был Эдвард Торндайк (Edward Thorndike):

Из нескольких возможных реакций на одну и ту же ситуацию, при прочих равных условиях, более тесно окажутся связанными с ситуацией те, что сопровождаются сразу или в недалеком будущем удовлетворенностью животного, поэтому, когда ситуация повторится, с большой вероятностью повторятся и эти реакции. А те реакции, которые сопровождаются сразу или в недалеком будущем неудовлетворенностью животного, будут иметь ослабленную связь с ситуацией, поэтому, когда ситуация повторится, эти реакции повторятся с меньшей вероятностью. Чем сильнее удовлетворенность или неудовлетворенность, тем сильнее или слабее связь (Thorndike, 1911, p. 244).

Торндайк называл это «законом эффекта», поскольку он описывает эффект, который подкрепляющие события оказывают на тенденцию выбора действий. Впоследствии Торндайк модифицировал этот закон, чтобы точнее учесть влияние последующих данных на обучение животного (например, различие между эффектами вознаграждения и наказания), и различные формы этого закона вызывали жаркие споры между теоретиками обучения (см., например, Gallistel, 2005; Herrnstein, 1970; Kimble, 1961, 1967; Mazur, 1994). Но, несмотря на это, закон эффекта – в той или иной форме – широко признан как основной принцип, определяющий многие виды поведения (Hilgard and Bower, 1975; Dennett, 1978; Campbell, 1960; Cziko, 1995). Это основа влиятельных теорий обучения Кларка Халла (Clark Hull, 1943, 1952) и общепринятых экспериментальных методов Б. Ф. Скиннера (B. F. Skinner, 1938).

Термин «подкрепление» в контексте обучения животных вошел в употребление спустя много лет после того, как Торндайк сформулировал закон эффекта, впервые он появился в этом контексте (насколько нам известно) в переводе на английский язык монографии Павлова об условных рефлексах, вышедшей в 1927 году. Павлов описывал подкрепление как усиление модели поведения вследствие получения животным стимула – подкрепителя – в непосредственной временной связи с другим стимулом или с реакцией. Некоторые психологи развили идею подкрепления, включив в нее не только усиление, но и ослабление поведения, а идею подкрепителя дополнили возможностью пропуска или прекращения стимулирования. Подкрепитель может считаться таковым, только если усиление или ослабление продолжается после его удаления; стимул, который просто привлекает внимание животного или инициирует его поведение, не порождая устойчивых изменений, не является подкрепителем.

Идея реализации обучения методом проб и ошибок в компьютере принадлежит к числу самых ранних идей о возможности искусственного интеллекта. В отчете 1948 года Алан Тьюринг описал конструкцию «системы удовольствие–неудовольствие», работавшей в духе закона эффекта:

При достижении конфигурации, в которой действие не определено, производится случайный выбор недостающих данных, соответствующая запись в порядке эксперимента вносится в описание и применяется. Если имеет место стимул неудовольствия, то все экспериментальные записи отменяются, а если удовольствия, то все они становятся постоянными (Turing, 1948).

Было сконструировано много хитроумных электромеханических устройств, демонстрирующих обучение методом проб и ошибок. Возможно, самое первое построил Томас Росс (Thomas Ross, 1933), оно умело находить выход из простого лабиринта и запоминало путь с помощью установки переключателей. В 1951 году У. Грей Уолтер построил вариант «механической черепахи» (Walter, 1950), способной к простым формам обучения. В 1952-м Клод Шеннон продемонстрировал мышь по кличке Тесей, которая применяла метод проб и ошибок для поиска выхода из лабиринта, при этом сам лабиринт запоминал приведшие к успеху направления с помощью магнитов и реле, расположенных под полом (см. также Shannon, 1951). В работе J. A. Deutsch (1954) описана машина для решения лабиринтов, основанная на его теории поведения (Deutsch, 1953), которая в некоторых отношениях напоминала основанное на модели обучение с подкреплением (глава 8). Марвин Мински в своей докторской диссертации (Marvin Minsky, 1954) рассматривал вычислительные модели обучения с подкреплением и описал конструкцию аналоговой машины, состоящей из компонентов, которые он называл SNARC'ами (Stochastic Neural-Analog Reinforcement Calculators (стохастические нейроаналоговые калькуляторы подкрепления) и которые моделировали изменяемые синаптические связи в мозге (глава 15). На сайте cyberneticzoo.com имеется много информации об этих и многих других электромеханических обучающихся машинах.

Построение электромеханических самообучающихся машин уступило место программированию цифровых компьютеров для осуществления разнообразных типов обучения, в т. ч. методом проб и ошибок. В работе Farley and Clark (1954) описано цифровое имитационное моделирование машины с нейронной сетью, обучающейся методом проб и ошибок. Но вскоре их научные интересы сместились в сторону обобщения и распознавания образов, т. е. от обучения с подкреплением к обучению с учителем (Clark and Farley, 1955). Это положило начало путанице в части, касающейся связей между этими типами обучения. Многие ученые полагали, что занимаются обучением с подкреплением, тогда как на самом деле изучали обучение с учителем. Например, такие пионеры искусственного интеллекта, как Rosenblatt (1962) и Widrow and Hoff (1960), безусловно, вдохновлялись обучением с подкреплением – они оперировали терминами «вознаграждение» и «наказание», – но при этом изучали системы обучения с учителем, пригодные для распознавания образов и перцептивного обучения. Даже в наши дни некоторые исследователи и авторы учебников преуменьшают или затушевывают различия между этими типами обучения. Например, в некоторых учебниках по нейронным сетям использовался термин «метод проб и ошибок» для описания сетей,

которые обучаются на примерах. Эта путаница понятна, поскольку в таких сетях информация об ошибках используется для обновления весов связей. Но при этом упускается из виду существенная характеристика обучения методом проб и ошибок – выбор действия на основе оценочной обратной связи без знания о том, какое действие правильно.

Отчасти из-за этой путаницы исследования по настоящему обучению методом проб и ошибок стали редкостью в 1960-х и 1970-х годах, хотя есть и заметные исключения. В 1960-х годах термины «подкрепление» и «обучение с подкреплением» начали использоваться в технической литературе для описания инженерных применений обучения методом проб и ошибок (см., например, Waltz and Fu, 1965; Mendel, 1966; Fu, 1970; Mendel and McClaren, 1970). Особенно большое влияние оказала работа Мински «На пути к созданию искусственного интеллекта» (Minsky, 1961), где обсуждались некоторые вопросы, относящиеся к обучению методом проб и ошибок, в т. ч. прогнозирование, ожидание и то, что он называл *базовой задачей распределения поощрения для сложных систем обучения с подкреплением*: как распределить поощрение между многими решениями, которые могут быть причастны к успеху? Все обсуждаемые в этой книге методы в каком-то смысле направлены на решение этой задачи. Эта работа Мински заслуживает прочтения и сегодня.

В следующих абзацах мы обсудим ряд других исключений и частичных исключений из царившего в тот период пренебрежения к изучению истинного обучения методом проб и ошибок с вычислительной и теоретической точек зрения.

Одним из исключений стала работа новозеландского ученого Джона Андреа (John Andreae), разработавшего систему STeLLA, которая обучалась методом проб и ошибок во взаимодействии с окружающей средой. Система включала внутреннюю модель мира, а впоследствии еще и «внутренний монолог» для решения проблемы скрытого состояния (Andreae, 1963, 1969a, b). В более поздней работе (Andreae, 1977) упор сделан больше на обучение с учителем, но все равно включена тема обучения методом проб и ошибок, причем порождение новых событий провозглашалось одной из целей системы. Примечательной особенностью этой работы стал «процесс подсоса», более полно разработанный в работе Andreae (1998). Это была реализация механизма распределения поощрения, похожая на операцию обновления, описанную нами выше. К сожалению, его пионерские исследования не получили широкой известности и не оказали существенного влияния на последующие изыскания в области обучения с подкреплением. Имеется написанный недавно реферат (Andreae, 2017a, b).

Более влиятельной оказалась работа Дональда Мичи (Donald Michie). В 1961 и 1963 годах он описал простую систему обучения игре в крестики-нолики методом проб и ошибок, которую назвал MENACE (Matchbox Educable Naughts and Crosses Engine). В ней каждой возможной игровой позиции соответствовал спичечный коробок (matchbox), содержащий ряд разноцветных бусин для каждого возможного в этой позиции хода. Извлечение случайной бусины из коробка, соответствующего текущей позиции, определяло ход MENACE. По завершении игры в коробки добавлялись или убирались использованные бусины, таким образом, ходы MENACE получали вознаграждение или наказание. В работе Michie and Chambers (1968) описана еще одна обучающаяся с подкреплением система игры в крестики-нолики, названная GLEE (Game Learning Expectimaxing Engine), и контроллер

обучения с подкреплением BOXES. Авторы применяли BOXES к задаче обучения, состоявшей в балансировании стержня, шарнирно закрепленного на движущейся тележке. В ней сигнал неудачи поступал, только когда стержень падал или тележка достигала конца дорожки. Эта задача ранее рассматривалась в работе Widrow and Smith (1964), в которой использовались методы обучения с учителем, в предположении, что инструкции поступают от учителя, который уже умеет балансировать стержень. Решение, предложенное Мичи и Чемберсом, – один из лучших ранних примеров задачи обучения с подкреплением в условиях неполного знания. Оно оказало влияние на многие более поздние работы по обучению с подкреплением, в т. ч. и наши собственные исследования (Barto, Sutton, and Anderson, 1983; Sutton, 1984). Мичи настойчиво подчеркивал роль метода проб и ошибок и обучения как существенных составных частей искусственного интеллекта (Michie, 1974).

В работе Widrow, Gupta, and Maitra (1973) был модифицирован алгоритм минимальной среднеквадратической ошибки (Least-Mean-Square – LMS), впервые изложенный в работе Widrow and Hoff (1960). Его целью было породить правило обучения с подкреплением, которое позволяло бы обучаться на сигналах об успехе и неудаче, а не на специально подготовленных обучающих примерах. Они называли эту форму обучения «избирательным бутстрэппингом» и описывали как «обучение с критиком», в отличие от «обучения с учителем». Они проанализировали это правило и показали, как алгоритм можно было обучить игре в блэкджек. Это был единичный экскурс Уидроу в область обучения с подкреплением, основные его работы, гораздо более влиятельные, относятся к обучению с учителем. Наше употребление термина «критик» заимствовано из статьи Widrow, Gupta, and Maitra. В работе Buchanan, Mitchell, Smith, and Johnson (1978) независимо использовался тот же термин в контексте машинного обучения (см. также Dietterich and Buchanan, 1984), но в их понимании критик – это экспертная система, способная на большее, чем просто оценка качества работы.

Исследования по *самообучающимся автоматам* оказали более прямое влияние на направление, связанное с методом проб и ошибок, и проложили путь к современным работам по обучению с подкреплением. Это методы решения неассоциативной чисто селекционной задачи обучения, известной под названием *k-рукий бандит* – по аналогии с игральным автоматом, или «одноруким бандитом», только рычагов не один, а k (см. главу 2). Самообучающиеся автоматы – это простые машины с небольшим объемом памяти, которые повышают вероятность получения вознаграждения в таких задачах. Их изучению положила начало работа русского математика и физика М. Л. Цетлина, выполненная совместно с коллегами в 1960-х годах (опубликована посмертно в Tsetlin, 1973). Затем эта тема активно разрабатывалась в интересах техники (см. Narendra and Thathachar, 1974, 1989). Изучались в том числе *стохастические самообучающиеся автоматы*, т. е. методы обновления вероятностей действий на основе сигналов вознаграждения. Алгоритм Alopex (Algorithm of pattern extraction), хотя и созданный не в традиции стохастических самообучающихся автоматов (Harth and Tzanakou, 1974), является стохастическим методом выявления корреляций между действиями и подкреплениями. Он оказал влияние на некоторые наши ранние исследования (Barto, Sutton, and Brouwer, 1981). Предвестником стохастических самообучающихся автоматов были более ранние работы по психологии, начиная с попыток Уильяма Эстеса (William Estes, 1950) создать статистическую теорию обучения, которая

затем получила развитие в работах других ученых (см., например, Bush and Mosteller, 1955; Sternberg, 1963).

Статистические теории обучения, разработанные в психологии, были подхвачены исследователями в области экономики, что заложило в этой области направление, посвященное обучению с подкреплением. Эта работа началась в 1973 году приложением теории обучения Буша и Мостеллера к ряду классических экономических моделей (Cross, 1973). Одной из целей было изучение искусственных агентов, которые действовали бы в большей степени как реальные люди, чем традиционные идеализированные экономические агенты (Arthur, 1991). Этот подход был распространен на изучение обучения с подкреплением в контексте теории игр. Обучение с подкреплением в экономике развивалось в значительной мере независимо от ранних работ по обучению с подкреплением в искусственном интеллекте, хотя теория игр остается темой, вызывающей интерес в обеих областях (но она выходит за рамки этой книги). В работе Camerer (2011) обсуждается традиция обучения с подкреплением в экономике, а в работе Nowé, Vrancx, and De Hauwere (2012) приведен обзор предмета с точки зрения многоагентных обобщений подхода, описываемого в этой книге. Подкрепление в контексте теории игр во многом отличается от того обучения с подкреплением, которое используется в таких игровых программах, как крестики-нолики, шашки и прочие развлечения. Обзор этого аспекта обучения с подкреплением и игр см., например, в работе Szita (2012).

В работе John Holland (1975) намечены контуры общей теории адаптивных систем, основанных на селекционных принципах. В его ранних работах метод проб и ошибок рассматривался в основном в неассоциативной форме, как в эволюционных методах и *k*-руких бандитах. В 1976 и затем более подробно в 1986 году он описал *системы классификации*, настоящие системы обучения с подкреплением, включающие ассоциацию и функции ценности. Ключевым компонентом систем классификации Холланда был «алгоритм пожарной цепочки» для распределения поощрения, тесно связанный с алгоритмом на основе временных различий, который использовался в нашем примере игры в крестики-нолики и обсуждается в главе 6. Еще одним важным компонентом был *генетический алгоритм* – эволюционный метод, роль которого заключалась в эволюционном порождении полезных представлений. Системы классификации всесторонне изучались многими учеными и стали крупной ветвью исследований по обучению с подкреплением (см. обзор Urbanowicz and Moore, 2009), но генетические алгоритмы – которые сами по себе, на наш взгляд, не являются системами обучения с подкреплением – удостоились гораздо большего внимания, как и другие подходы к эволюционным вычислениям (см., например, Fogel, Owens and Walsh, 1966, и Koza, 1992).

Человеком, который больше всех сделал для возрождения интереса к методу проб и ошибок в обучении с подкреплением в контексте искусственного интеллекта, стал Гарри Клопф (Harry Klopff, 1972, 1975, 1982). Клопф осознал, что существенные аспекты адаптивного поведения теряются, поскольку исследователи, занимающиеся обучением, чуть ли не поголовно увлечены одним лишь обучением с учителем. Согласно Клопфу, при этом утрачиваются гедонистические аспекты поведения, стремление добиться каких-то результатов от окружающей среды, управлять средой для приближения к желаемой цели прочь от нежелательной (раздел 15.9). Это основная идея обучения методом проб и ошибок. Идеи Клопфа оказали очень сильное влияние на авторов, поскольку в процессе их оценки (Barto and

Sutton, 1981a) мы пришли к пониманию различия между обучением с учителем и обучением с подкреплением и решили заняться последним. Значительная часть ранних работ, выполненных нами и нашими коллегами, была направлена на доказательство того, что обучение с учителем и обучение с подкреплением – действительно разные вещи (Barto, Sutton, and Brouwer, 1981; Barto and Sutton, 1981b; Barto and Anandan, 1985). Другие исследования показали, что обучение с подкреплением можно применить к решению важных задач обучения нейронных сетей, в частности к порождению алгоритмов обучения многослойных сетей (Barto, Anderson, and Sutton, 1982; Barto and Anderson, 1985; Barto, 1985, 1986; Barto and Jordan, 1987).

Теперь обратимся к третьему направлению в истории обучения с подкреплением – обучению на основе временных различий. Такие методы обучения отличаются тем, что основаны на различиях между оценками одной и той же величины (например, вероятности выигрыша в крестики-нолики), сделанными в соседние моменты времени. Это направление не такое обширное и не так выделяется, как два других, но оно сыграло особенно важную роль в этой области, отчасти потому, что методы на основе временных различий – нечто новое и существующее только в обучении с подкреплением.

Истоки обучения на основе временных различий следует искать в том числе в психологии обучения животных и, в частности, в понятии вторичных подкрепителей. Вторичным подкрепителем называется стимул, который подавался в паре с первичным подкрепителем, например пищей или болью, и в результате приобрел сходные свойства в плане подкрепления. Мински (Minsky, 1954), возможно, был первым, кто понял, что этот психологический принцип мог бы оказаться важным для систем обучения искусственного интеллекта. В работе Arthur Samuel (1959) впервые был предложен и реализован метод обучения, включавший идеи временных различий, это была часть его знаменитой программы игры в шашки (раздел 16.2).

Сэмюэл не ссылаясь на работу Мински и на возможные связи с обучением животных. Он, по-видимому, черпал вдохновение из предположения Клода Шеннона (Claude Shannon, 1950) о том, что компьютер можно запрограммировать для игры в шахматы с использованием функции оценки и что, возможно, его игру можно улучшить, если модифицировать эту функцию динамически. (Не исключено, что эти идеи Шеннона оказали также влияние на Беллмана, но у нас нет никаких подтверждений данной гипотезы.) Мински в своей работе 1961 года «На пути к...» подробно обсуждал работу Сэмюэла и предлагал связь с теориями вторичного подкрепления в обучении естественных и искусственных систем.

Как мы говорили, за десять лет, последовавших за исследованиями Мински и Сэмюэла, в области обучения методом проб и ошибок было выполнено мало вычислительных работ и уж совсем никаких по обучению на основе временных различий. В 1972 году Клопф соединил обучение методом проб и ошибок с важным компонентом обучения на основе временных различий. Клопфа интересовали принципы, которые масштабировались бы на обучение в больших системах, а потому он был заинтригован понятием локального подкрепления, посредством которого части системы обучения могли бы подкреплять друг друга. Он развил идею «обобщенного подкрепления», согласно которой каждый компонент (номинально: каждый нейрон) рассматривает все свои входы в терминах подкрепления: возбудители – как вознаграждение, а ингибиторы – как наказание. Это не

совсем та идея, которую мы теперь знаем под названием обучения на основе временных различий, и, оглядываясь назад, можно сказать, что она дальше от него, чем работа Сэмюэла. С другой стороны, Клопф связал эту идею с обучением методом проб и ошибок и сопоставил ее с большим объемом эмпирических данных по психологии обучения животных.

В работах Sutton (1978a, b, c) идеи Клопфа получили дальнейшее развитие, особенно в части связей с теориями обучения животных, в которых описывались правила обучения, основанные на изменениях предсказаний, сделанных в соседние моменты времени. Саттон вместе с Барто уточнили эти идеи и разработали психологическую модель классического обусловливания, положив в ее основу обучение на базе временных различий (Sutton and Barto, 1981a; Barto and Sutton, 1982). Затем последовали другие оказавшие заметное влияние модели того же вида (например, Klopff, 1988; Moore et al., 1986; Sutton and Barto, 1987, 1990). Некоторые нейронаучные модели, разработанные в то время, хорошо интерпретируются в терминах обучения на основе временных различий (Hawkins and Kandel, 1984; Byrne, Gingrich, and Baxter, 1990; Gelperin, Hop_eld, and Tank, 1985; Tesauro, 1986; Friston et al., 1994), хотя в большинстве случаев никаких исторических связей не прослеживается.

На наши ранние работы по обучению на основе временных различий оказали большое влияние теории обучения животных и работа Клопфа. Связи со статьей Мински «На пути к...» и с программами игры в шашки Сэмюэла были осознаны лишь впоследствии. Но к 1981 году мы прекрасно знали обо всех вышеупомянутых работах предшественников в контексте направлений, связанных с методом проб и ошибок и идеей временных различий. В то время мы разработали метод использования обучения на основе временных различий в сочетании с обучением методом проб и ошибок, получивший название *архитектура исполнитель–критик*, и применили его к задаче балансирования стержня Мичи и Чамберса (Barto, Sutton, and Anderson, 1983). Этот метод был подробно исследован в докторской диссертации Саттона (Sutton, 1984) и обобщен на нейронные сети с обратным распространением в докторской диссертации Андерсона (Anderson, 1986). Примерно в то же время Холланд (1986) явно включил идеи временных различий в свои системы классификации в форме алгоритма пожарной цепочки. Ключевой шаг был сделан в работе Sutton (1988), где обучение на основе временных различий было отделено от управления и рассмотрено как общий метод прогнозирования. В той же работе был сформулирован алгоритм TD(λ) и доказаны некоторые его свойства сходимости.

Когда в 1981 году работа над архитектурой исполнитель–критик подходила к концу, мы наткнулись на статью Ian Witten (1977, 1976a), которая, похоже, является самой ранней публикацией на тему правила обучения на основе временных различий. Автор предложил метод, который мы теперь называем табличным TD(0), для использования в составе адаптивного контроллера для МППР. Эта работа была предложена для публикации в журнале в 1974 году и вошла также в докторскую диссертацию Виттена, написанную в 1976 году. Работа Виттена шла по следам ранних экспериментов Андрэ с системой STeLLA и других систем обучения методом проб и ошибок. Таким образом, статья Виттена 1977 года охватывала оба основных направления исследований по обучению с подкреплением – метод проб и ошибок и оптимальное управление – и при этом внесла весомый ранний вклад в обучение на основе временных различий.

Направления, связанные с временными различиями и оптимальным управлением, полностью слились в 1989 году, когда Крис Уоткинс разработал Q-обучение. Эта работа знаменовала обобщение и объединение предшествующих исследований по всем трем направлениям обучения с подкреплением. Пал Вербос (Paul Werbos, 1987) внес вклад в это объединение тем, что начиная с 1977 года ратовал за конвергенцию обучения методом проб и ошибок и динамического программирования. К моменту выхода работы Уоткинса наблюдался колоссальный рост количества исследований по обучению с подкреплением, в основном в области машинного обучения, но также в контексте искусственных нейронных сетей и искусственного интеллекта в широком смысле. 1992 год ознаменовался замечательным успехом программы игры в нарды Джерри Тезауро, которая привлекла дополнительное внимание к этой тематике.

За время, прошедшее с момента выхода первого издания этой книги, получила развитие и донныне процветает подобласть нейронауки, изучающая связи между алгоритмами обучения с подкреплением и обучением с подкреплением в нервной системе. Причина тому – необыкновенное сходство между поведением алгоритмов на основе временных различий и активностью нейронов мозга, продуцирующих дофамин, на что указывали многие исследователи (Friston et al., 1994; Barto, 1995a; Houk, Adams, and Barto, 1995; Montague, Dayan, and Sejnowski, 1996; and Schultz, Dayan, and Montague, 1997). В главе 15 приведено введение в эту увлекательную область обучения с подкреплением. Важных достижений в недавней истории обучения с подкреплением слишком много, чтобы можно было упомянуть их все в этом кратком очерке, но многие отмечаются в конце тех глав, где о них идет речь.

БИБЛИОГРАФИЧЕСКИЕ ЗАМЕЧАНИЯ

За дополнительными сведениями об обучении с подкреплением в целом отсылаем читателя к книгам Szepesvári (2010), Bertsekas and Tsitsiklis (1996), Kaelbling (1993a) и Sugiyama, Hachiya, and Morimura (2013). Из книг, рассматривающих предмет с позиций теории управления или исследования операций, отметим Si, Barto, Powell, and Wunsch (2004), Powell (2011), Lewis and Liu (2012) и Bertsekas (2012). В обзоре Cao (2009) обучение с подкреплением поставлено в один ряд с другими подходами к обучению и оптимизации стохастических динамических систем. Три специальных выпуска журнала «Machine Learning» посвящены обучению с подкреплением: Sutton (1992a), Kaelbling (1996) и Singh (2002). Упомянем полезные обзоры: Barto (1995b); Kaelbling, Littman, and Moore (1996) и Keerthi and Ravindran (1997). В томе под редакцией Weiring and van Otterlo (2012) имеется отличный обзор недавних достижений.

1.2 Пример с завтраком Фила заимствован из работы Agre (1988).

1.5 Метод на основе временных различий, использованный в примере игры в крестики-нолики, разрабатывается в главе 6.

ТАБЛИЧНЫЕ МЕТОДЫ РЕШЕНИЯ

В этой части книги описываются почти все основные идеи алгоритмов обучения с подкреплением в их простейшей форме – когда пространства состояний и действий настолько малы, что приближенные функции ценности можно представить в виде массивов или таблиц. В этом случае методы часто находят точное решение, т. е. оптимальную функцию ценности и оптимальную стратегию. А в следующей части книги будут описаны приближенные методы, которые могут найти только приближенное решение, но зато допускают эффективное применение к гораздо более крупным задачам.

В первой главе этой части описываются методы решения для частного случая задачи обучения с подкреплением, в котором есть только одно состояние. Это так называемые задачи о бандите. Во второй главе приводится общая постановка задачи, с которой мы будем иметь дело в остальной книге, – конечные марковские процессы принятия решений – и основные связанные с ней идеи, включая уравнение Беллмана и функции ценности.

В следующих трех главах описываются три фундаментальных класса методов для решения конечных задач МППР: динамическое программирование, методы Монте-Карло и обучение на основе временных различий. У каждого класса методов есть свои сильные и слабые стороны. Методы динамического программирования хорошо разработаны математически, но нуждаются в полной и точной модели окружающей среды. Методам Монте-Карло модель не нужна, и концептуально они просты, но плохо приспособлены для пошаговых инкрементных вычислений. Наконец, методам обучения на основе временных различий модель тоже не нужна, и они инкрементны от начала до конца, но более сложны для анализа. Кроме того, методы различаются по эффективности и скорости сходимости.

В оставшихся двух главах описывается, как можно комбинировать методы этих трех классов для получения лучшего из того, что может дать каждый. Сначала мы покажем, как сильные стороны методов Монте-Карло можно объединить с сильными сторонами методов на основе временных различий посредством методов многоступенчатого бутстрэппинга. А в последней главе продемонстрируем комбинацию методов на основе временных различий с методами планирования (к какому относится и динамическое программирование) для получения полного унифицированного решения задачи табличного обучения с подкреплением.

Глава 2

Многорукие бандиты

Самое важное отличие обучения с подкреплением от других видов обучения заключается в природе обучающей информации: требуется *оценивать* действия, а не получать *инструкции* о том, какие действия правильны. Именно это приводит к необходимости активного исследования, т. е. явного поиска хорошего поведения. Чисто оценочная обратная связь показывает, насколько хорошим было выбранное действие. С другой стороны, чисто наставническая обратная связь указывает, какое действие правильно, независимо от того, какое фактически было выбрано. Такой вид обратной связи – основа обучения с учителем, к которому в значительной своей части относятся классификация образов, искусственные нейронные сети и идентификация систем. В своих чистых формах эти два вида обратной связи совершенно различны: оценочная целиком зависит от выбранного действия, а наставническая от выбранного действия вообще не зависит.

В этой главе мы изучаем оценочный аспект обучения с подкреплением в упрощенной постановке – той, что не требует от обучаемого действовать более чем в одной ситуации. Именно для такой *неассоциативной* постановки была проделана большая часть предшествующей работы, относящейся к оценочной обратной связи. Так мы сможем избежать сложностей, присущих полной постановке задачи обучения с подкреплением. Изучение этого случая позволит нам максимально отчетливо увидеть, чем оценочная обратная связь отличается от наставнической и как их, тем не менее, можно комбинировать.

В качестве конкретной неассоциативной задачи с оценочной обратной связью мы исследуем простой вариант задачи о k -руком бандите. На примере этой задачи мы познакомимся с рядом базовых методов обучения, которые в последующих главах обобщим на полную постановку задачи обучения с подкреплением. В конце этой главы мы на шаг приблизимся к полной постановке, обсудив, что происходит, когда задача о бандите становится ассоциативной, т. е. когда действия предпринимаются более чем в одной ситуации.

2.1. Задача о k -руком бандите

Рассмотрим следующую задачу обучения. Вы многократно стоите перед выбором из k разных вариантов, или действий. После каждого выбора вы получаете чис-

ленное вознаграждение, выбираемое из стационарного распределения вероятностей, которое зависит от выбранного действия. Ваша цель – максимизировать ожидаемое полное вознаграждение за период, например, после выбора действия 1000 раз (за 1000 временных шагов).

Это исходная постановка задачи о k -руком бандите, названной так по аналогии с игровым автоматом, или «одноруким бандитом». Только в этом случае рычагов не один, а k . Каждый выбор действия соответствует опусканию одного из рычагов игрового автомата, а вознаграждения – это выплаты за выпадение джекпота. Благодаря повторному выбору действий мы стремимся максимизировать свой выигрыш, концентрируя усилия только на лучших рычагах. Еще одна аналогия – врач, выбирающий различные экспериментальные способы лечения нескольких пациентов, страдающих серьезной болезнью. Каждое действие – это выбор лечения, а вознаграждение – выживание или выздоровление пациента. В настоящее время термин «задача о бандите» иногда означает обобщение описанной выше задачи, но в этой книге мы ограничимся только простым случаем.

В нашей задаче о k -руком бандите с каждым из k действий связано ожидаемое, или среднее, вознаграждение при условии выбора этого действия; будем называть его ценностью действия. Обозначим A_t действие, выбранное на временном шаге t , а R_t – соответствующее ему вознаграждение. Тогда ценность произвольно-го действия a , обозначаемая $q_*(a)$, – это математическое ожидание вознаграждения при условии выбора a :

$$q_*(a) \doteq \mathbb{E}[R_t | A_t = a].$$

Если бы мы знали ценность каждого действия, то задача о k -руком бандите решалась бы тривиально: нужно было бы всегда выбирать действие с максимальной ценностью. Предположим, что достоверно ценности действий неизвестны, но имеются оценки. Обозначим $Q_t(a)$ оценку ценности действия a на временном шаге t . Мы хотели бы, чтобы $Q_t(a)$ было близко к $q_*(a)$.

Если мы запоминаем оценки ценности действий, то на любом временном шаге имеется по крайней мере одно действие с максимальной оценкой. Назовем эти действия *жадными*. Если выбирается любое из таких действий, то мы *используем* текущие знания о ценности действий. Если же выбирается какое-то нежадное действие, то мы занимаемся *исследованием*, поскольку это позволяет улучшить оценку ценности нежадного действия. Использование дает возможность максимизировать ожидаемое вознаграждение на одном шаге, а исследование может дать большее суммарное вознаграждение в длительной перспективе. Предположим, например, что ценность жадного действия известна достоверно, тогда как оценки других действий почти такие же хорошие, но со значительной долей недоверности. Недостоверность такова, что по меньшей мере одно из этих прочих действий на самом деле лучше жадного, но какое именно, мы не знаем. Если впереди много временных шагов, на которых можно выбирать действия, то может оказаться выгоднее исследовать нежадные действия и выяснить, какие из них лучше жадных. В краткосрочной перспективе вознаграждение во время исследования будет ниже, но в долгосрочной – выше, потому что, обнаружив лучшее действие, мы сможем использовать его много раз. Поскольку при выборе одного действия нельзя одновременно исследовать и использовать, часто говорят о «конфликте» между исследованием и использованием.

В каждом конкретном случае решение о том, что лучше – исследовать или использовать, – сложным образом зависит от точных значений оценок, недостоверности и количества оставшихся шагов. Есть много изоощренных методов нахождения баланса между исследованием и использованием для различных математических постановок задачи о k -руком бандите и родственных ей. Но в большинстве из них делаются сильные предположения о стационарности и априорной информации, которые либо нарушаются, либо не поддаются проверке в приложениях и в полной задаче обучения с подкреплением, которую мы будем рассматривать в последующих главах. Наличие для этих методов гарантий оптимальности или ограниченности потери слабо утешает, если предположения теории неприменимы.

В этой книге нас не будут интересовать сложные компромиссы между исследованием и использованием, нам интересен хоть какой-нибудь компромисс. Мы представим в этой главе несколько простых методов балансирования для задачи о k -руком бандите и покажем, что они работают намного лучше, чем методы, которые всегда прибегают к использованию. Необходимость поиска компромисса между исследованием и использованием – проблема, характерная для обучения с подкреплением, а простота нашей постановки задачи о k -руком бандите позволит продемонстрировать это максимально отчетливо.

2.2. МЕТОДЫ ЦЕННОСТИ ДЕЙСТВИЙ

Начнем с более пристального рассмотрения методов оценивания ценности действий и использования оценок при решении о выборе действия. Будем называть их *методами ценности действий*. Напомним, что истинная ценность действия – это среднее вознаграждение при условии выбора этого действия. Эту величину можно естественно оценить, усреднив фактически полученные вознаграждения:

$$Q_t(a) \doteq \frac{\text{Сумма вознаграждений при выборе } a \text{ до момента } t}{\text{Сколько раз } a \text{ выбиралось до момента } t} \\ = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}, \quad (2.1)$$

где $\mathbb{1}_{predicate}$ обозначает случайную величину, равную 1, если предикат *predicate* равен true, и 0 в противном случае. Если знаменатель равен 0, то в качестве $Q_t(a)$ принимается какое-нибудь значение по умолчанию, например 0. Когда знаменатель стремится к бесконечности, $Q_t(a)$, по закону больших чисел, сходится к $q_*(a)$. Назовем это методом *выборочного среднего* для оценки ценности значений, поскольку каждая оценка является средним по выборке релевантных действию вознаграждений. Разумеется, это лишь один из способов оценивания ценности действий, и необязательно самый лучший. Но пока остановимся на нем и займемся вопросом о том, как можно использовать оценки для выбора действий.

Простейшее правило – выбирать действие с наибольшей оценкой ценности, т. е. одно из жадных действий в смысле предыдущего раздела. Если жадных действий несколько, то выбирается любое из них, например, случайным образом. Этот метод выбора жадного действия мы записываем так:

$$A_t \doteq \operatorname{argmax}_a Q_t(a), \quad (2.2)$$

где argmax_a обозначает действие, для которого следующее далее выражение принимает максимум (неоднозначности опять-таки разрешаются произвольно). При выборе жадного действия всегда используется текущее знание, чтобы максимизировать немедленное вознаграждение; мы вообще не тратим времени на попытку проверить, не окажутся ли, на первый взгляд, менее доходные действия в действительности более выгодными. Простая альтернатива – вести себя жадно большую часть времени, но иногда, скажем с малой вероятностью ε , случайным образом выбирать какое-то из прочих действий с одинаковой вероятностью, не зависящей от оценок ценности действий. Будем называть методы, в которых применяется такое почти жадное правило выбора действия, ε -жадными. Их преимущество состоит в том, что в пределе, когда число шагов стремится к бесконечности, каждое действие будет случайно выбрано бесконечное число раз, поэтому все $Q_t(a)$ сходятся к $q_*(a)$. Отсюда, конечно, следует, что вероятность выбора оптимального действия сходится к числу, большему $1 - \varepsilon$, т. е. оптимальный выбор почти гарантирован. Но эти гарантии асимптотические, а о практической эффективности они мало что говорят.

Упражнение 2.1. Рассмотрим ε -жадный выбор действия для случая двух действий и $\varepsilon = 0.5$. Чему равна вероятность выбора жадного действия? $\square$

2.3. 10-РУКИЙ ИСПЫТАТЕЛЬНЫЙ СТЕНД

Чтобы грубо оценить относительную эффективность жадных и ε -жадных методов ценности действий, мы сравнили их численно на наборе тестовых задач. В набор вошли 2000 случайно сгенерированных задач о k -руких бандитах при $k = 10$. Для каждой задачи, например показанной на рис. 2.1, ценности действий $q_*(a)$, $a = 1, \dots, 10$, выбирались из нормального распределения со средним 0 и дисперсией 1. Затем, когда примененный к этой задаче метод обучения выбирал на шаге t действие A_t , из нормального распределения со средним $q_*(A_t)$ и дисперсией 1 выбиралось вознаграждение R_t . Эти распределения показаны серым цветом на рис. 2.1. Такой набор тестовых задач мы будем называть *10-руким испытательным стендом*. Для любого метода обучения мы можем измерить улучшение качества и поведения с обретением опыта на протяжении 1000 временных шагов в применении к одной из задач о бандите. Это будет один *прогон* теста. Выполнив 2000 независимых прогонов, каждый раз с новой задачей о бандите, мы получим представление о среднем поведении алгоритма обучения.

На рис. 2.2 показаны результаты сравнения жадного метода с двумя описанными выше ε -жадными (при $\varepsilon = 0.01$ и $\varepsilon = 0.1$) на 10-руком испытательном стенде. Во всех методах оценки ценности значений были получены усреднением по выборке. На верхнем графике мы видим увеличение среднего вознаграждения по мере накопления опыта. Жадный метод показывает более высокий темп увеличения в самом начале, но затем выходит на плато, расположенное ниже, чем в остальных методах. Достигнутое вознаграждение в расчете на один шаг составляет приблизительно 1, тогда как наилучшее вознаграждение в этом тесте равно

примерно 1.55. В долгосрочной перспективе жадный метод работает значительно хуже, потому что часто застревает в неоптимальных действиях. На нижнем графике видно, что жадный метод находил оптимальное действие примерно в одной трети задач. В остальных двух третях выбор начального действия был никуда не годным, но метод никогда не пересматривал его. ε -жадные методы в конечном итоге показывали лучшие результаты, потому что продолжали исследовать и повышать шансы на отыскание оптимального действия. При $\varepsilon = 0.1$ метод посвящал исследованию больше времени и обычно находил оптимальное действие раньше, но никогда не выбирал это действие более чем в 91 % случаев. При $\varepsilon = 0.01$ улучшение происходило медленнее, но в конечном счете он дал бы лучшие результаты, чем для $\varepsilon = 0.1$, с точки зрения обоих показанных на рисунке показателей. Можно также уменьшать ε с течением времени, чтобы попытаться получить лучшее из обоих миров.

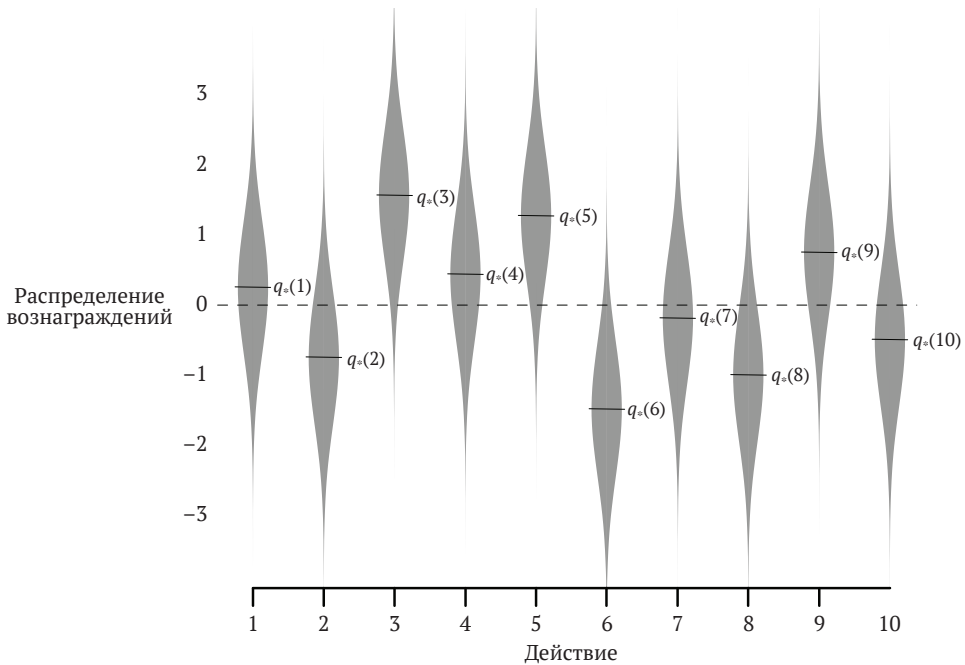


Рис. 2.1 ❖ Пример задачи о бандите из 10-рукого испытательного стенда. Истинное значение $q_*(a)$ для каждого из десяти действий выбиралось из нормального распределения со средним 0 и дисперсией 1, а затем вознаграждение выбиралось из нормального распределения со средним $q_*(a)$ и дисперсией 1 (распределения показаны серым цветом)

Будут ли ε -жадные методы лучше жадных, зависит от задачи. Предположим, например, что дисперсия вознаграждения больше, скажем, не 1, а 10. Тогда для нахождения оптимального действия понадобилось бы больше исследовать, и ε -жадные методы оказались бы еще лучше жадных. С другой стороны, если дисперсия вознаграждения равна нулю, то жадный метод будет знать истинную ценность каждого действия, выбрав его всего один раз. В таком случае жадный метод

мог бы работать лучше, потому что скоро нашел бы оптимальное действие, вообще не тратя времени на исследование. Но даже в детерминированном случае исследование дает заметные преимущества, если ослабить другие предположения. Например, предположим, что задача о бандите нестационарна, т. е. истинные ценности действий изменяются со временем. Тогда исследование необходимо даже в детерминированном случае, чтобы быть уверенным, что никакое нежадное действие не стало лучше жадного. В следующих главах мы увидим, что нестационарный случай чаще всего встречается в обучении с подкреплением. Но даже если задача стационарная и детерминированная, перед обучаемым стоит набор задач по принятию решений, каждое из которых изменяется со временем по мере того, как процесс обучения продвигается и стратегия принятия решений агентом претерпевает изменения. В обучении с подкреплением необходим баланс между исследованием и использованием.

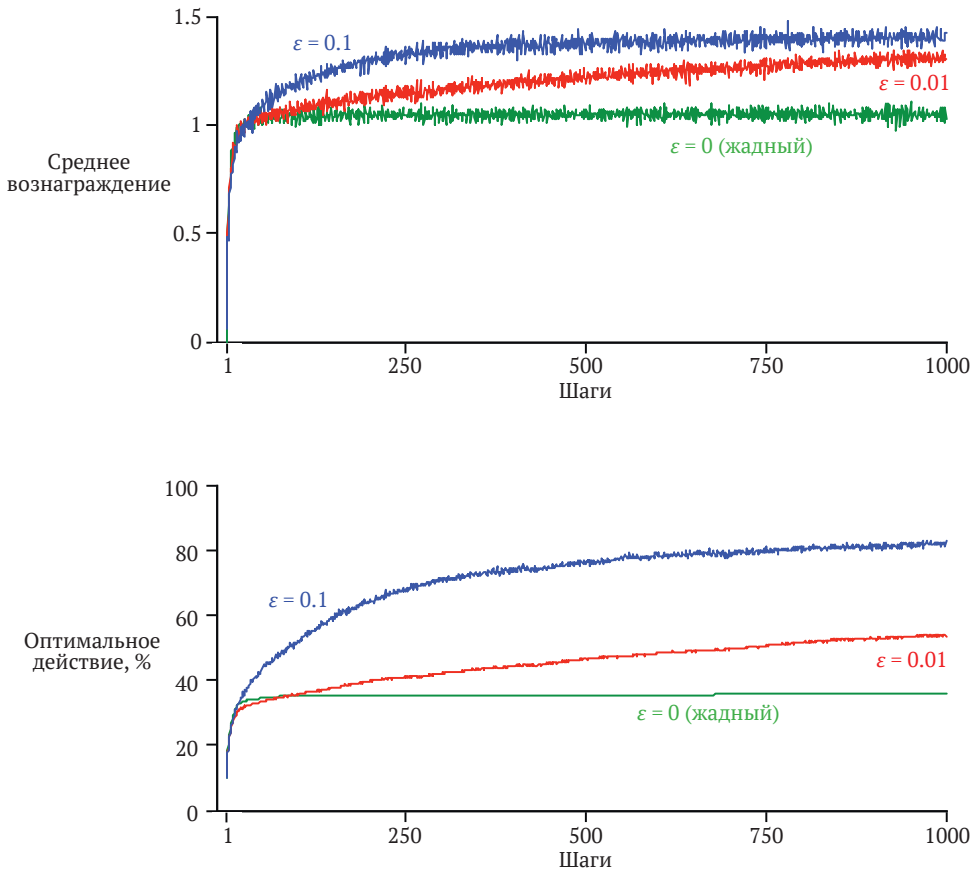


Рис. 2.2 ❖ Среднее качество ϵ -жадных методов оценки действий на 10-руком испытательном стенде. Данные получены усреднением по 2000 прогонов с разными задачами о бандите. Во всех методах использовалось выборочное усреднение оценок ценности действий

Упражнение 2.2. Пример бандита. Рассмотрим задачу о k -руком бандите с $k = 4$ действиями, обозначенными 1, 2, 3, 4. Разберем применение к этой задаче алгоритма бандита с ε -жадным выбором действия, выборочным усреднением оценок ценности действий и начальными оценками $Q_1(a) = 0$ для всех a . Предположим, что начало последовательности действий и вознаграждений выглядит так: $A_1 = 1, R_1 = 1, A_2 = 2, R_2 = -1, A_3 = 2, R_3 = -2, A_4 = 2, R_4 = 2, A_5 = 3, R_5 = 0$. На некоторых из этих временных шагов мог иметь место ε -случай, т. е. действие выбирается случайным образом. На каких шагах так оно заведомо и было? На каких шагах это могло случиться, а могло и не случиться? □

Упражнение 2.3. Глядя на рис. 2.2, можете ли вы сказать, какой метод будет работать лучше других в долгосрочной перспективе в терминах суммарного вознаграждения и вероятности выбора оптимального действия? Насколько лучше он будет? Выразите свой ответ количественно. □

2.4. ИНКРЕМЕНТНАЯ РЕАЛИЗАЦИЯ

Все рассмотренные до сих пор методы ценности действий оценивают ценность действия как среднее по выборке наблюдаемых вознаграждений. Теперь обратимся к вопросу о том, как можно эффективно вычислить эти средние, желательное с постоянным объемом потребляемой памяти и постоянными вычислительными затратами на одном временном шаге.

Чтобы упростить обозначения, сосредоточимся на одном действии. Обозначим R_i вознаграждение, полученное после i -го выбора *этого действия*, а Q_n – оценку его ценности, после того как оно было выбрано $n - 1$ раз. Это можно записать так:

$$Q_n \doteq \frac{R_1 + R_2 + \dots + R_{n-1}}{n - 1}.$$

Напрашивающаяся реализация – хранить информацию обо всех вознаграждениях и выполнять это вычисление всякий раз, как понадобится значение оценки. Но тогда требования к памяти и объем вычислений будут расти со временем, по мере поступления новых вознаграждений. Для каждого вознаграждения нужна дополнительная память, увеличивается также время вычисления суммы в числителе.

Как вы, наверное, догадались, это необязательно. Легко вывести инкрементную формулу для обновления средних, которая будет требовать небольшого и постоянного времени вычислений при каждом новом вознаграждении. Если известны Q_n и n -е вознаграждение R_n , то новое среднее по всем n вознаграждениям можно вычислить по формуле:

$$\begin{aligned}
Q_{n+1} &= \frac{1}{n} \sum_{i=1}^n R_i \\
&= \frac{1}{n} \left(R_n + \sum_{i=1}^{n-1} R_i \right) \\
&= \frac{1}{n} \left(R_n + (n-1) \frac{1}{n-1} \sum_{i=1}^{n-1} R_i \right) \\
&= \frac{1}{n} (R_n + (n-1)Q_n) \\
&= \frac{1}{n} (R_n + nQ_n - Q_n) \\
&= Q_n + \frac{1}{n} [R_n - Q_n],
\end{aligned} \tag{2.3}$$

которая справедлива даже для $n = 1$, поскольку дает $Q_2 = R_1$, каково бы ни было Q_1 . При такой реализации нужна только память для хранения Q_n и n и простое вычисление (2.3) для каждого нового вознаграждения.

Правило обновлений (2.3) имеет форму, которая часто будет встречаться в этой книге. В общем виде это выглядит так:

$$\text{НоваяОценка} \leftarrow \text{СтараяОценка} + \text{РазмерШага}[\text{Цель} - \text{СтараяОценка}] \tag{2.4}$$

Выражение $[\text{Цель} - \text{СтараяОценка}]$ называется *ошибкой* оценки. Она уменьшается благодаря шагу в направлении Цели. Предполагается, что цель указывает желательное направление движения, но она может быть искажена шумом. В данном случае целью является n -е вознаграждение.

Заметим, что параметр размера шага (РазмерШага), используемый в инкрементном методе (2.3), изменяется на каждом шаге. При обработке n -го вознаграждения за действие a используется размер шага $1/n$. В этой книге мы будем обозначать размер шага α или в более общем случае $\alpha_t(a)$.

Псевдокод полного алгоритма бандита с использованием инкрементно вычисляемых выборочных средних и ε -жадным выбором действий приведен во врезке ниже. Предполагается, что функция $\text{bandit}(a)$ выполняет действие и возвращает соответствующее вознаграждение.

Простой алгоритм бандита

Инициализировать для a от 1 до k :

$$Q(a) \leftarrow 0$$

$$N(a) \leftarrow 0$$

Повторять бесконечно:

$$A \leftarrow \begin{cases} \operatorname{argmax}_a Q(a) & \text{с вероятностью } 1 - \varepsilon \\ \text{случайное действие} & \text{с вероятностью } \varepsilon \end{cases} \quad \begin{matrix} \text{(неоднозначность} \\ \text{разрешается} \end{matrix}$$

$$R \leftarrow \text{bandit}(A) \quad \text{случайным образом)}$$

$$N(A) \leftarrow N(A) + 1$$

$$Q(A) \leftarrow Q(A) + 1/N(A) [R - Q(A)]$$

2.5. НЕСТАЦИОНАРНАЯ ЗАДАЧА

Обсуждавшиеся до сих пор методы усреднения годятся для стационарной задачи о бандите, т. е. для случая, когда вероятности вознаграждений не меняются со временем. Как уже отмечалось выше, в обучении с подкреплением часто встречаются задачи, которые стационарными не являются. В таких случаях имеет смысл придавать недавним вознаграждениям больший вес, чем полученным давно. Один из самых популярных способов такого рода – взять постоянный размер шага. Например, правило (2.3) инкрементного обновления среднего Q_n $n - 1$ прошлых вознаграждений теперь записывается в виде:

$$Q_{n+1} \doteq Q_n + \alpha[R_n - Q_n], \quad (2.5)$$

где размер шага – постоянное число $\alpha \in (0, 1]$. В результате Q_{n+1} оказывается взвешенным средним прошлых вознаграждений и начальной оценки Q_1 :

$$\begin{aligned} Q_{n+1} &= Q_n + \alpha[R_n - Q_n] \\ &= \alpha R_n + (1 - \alpha)Q_n \\ &= \alpha R_n + (1 - \alpha)[\alpha R_{n-1} + (1 - \alpha)Q_{n-1}] \\ &= \alpha R_n + (1 - \alpha)\alpha R_{n-1} + (1 - \alpha)^2 Q_{n-1} \\ &= \alpha R_n + (1 - \alpha)\alpha R_{n-1} + (1 - \alpha)^2 \alpha R_{n-2} + \dots + (1 - \alpha)^{n-1} \alpha R_1 + (1 - \alpha)^n Q_1 \\ &= (1 - \alpha)^n Q_1 + \sum_{i=1}^n \alpha (1 - \alpha)^{n-i} R_i. \end{aligned} \quad (2.6)$$

Мы называем эту величину взвешенным средним, потому что сумма весов равна $(1 - \alpha)^n + \sum_{i=1}^n \alpha (1 - \alpha)^{n-i} = 1$, что вы можете легко проверить самостоятельно. Отметим, что вес $\alpha (1 - \alpha)^{n-i}$, назначенный вознаграждению R_i , зависит от того, сколько временных шагов $(n - i)$ прошло с момента его наблюдения. Величина $1 - \alpha$ меньше 1, поэтому вес, назначенный R_i , убывает с ростом количества полученных после него вознаграждений. На самом деле вес убывает экспоненциально, как степень $1 - \alpha$. (Если $1 - \alpha = 0$, то весь вес отходит последнему вознаграждению R_n , поскольку, по соглашению, $0^0 = 1$.) Поэтому иногда встречается термин *экспоненциальное среднее, взвешенное по степени новизны*.

Бывает удобно изменять параметр размера на каждом шаге. Обозначим $\alpha_n(a)$ размер шага, используемый для обработки вознаграждения, полученного после n -го выбора действия a . Как мы уже видели, если $\alpha_n(a) = 1/n$, то получается метод выборочного среднего, который гарантированно сходится к истинным ценностям действий, по закону больших чисел. Но, конечно, сходимость можно гарантировать не для любого выбора последовательности $\{\alpha_n(a)\}$. Согласно хорошо известному результату из теории стохастической аппроксимации, сходимость гарантируется с вероятностью 1 при выполнении условий:

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \quad \text{и} \quad \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty. \quad (2.7)$$

Первое условие необходимо, чтобы гарантировать, что шаги достаточно большие, для того чтобы в конце концов превзойти любые начальные условия или

случайные флуктуации. А второе условие гарантирует, что рано или поздно шаги станут достаточно мелкими для обеспечения сходимости.

Заметим, что оба условия сходимости удовлетворяются для случая выборочного среднего, $\alpha_n(a) = 1/n$, но не для случая, когда размер шага постоянный, $\alpha_n(a) = \alpha$. Во втором случае не удовлетворяется второе условие, а это значит, что оценки никогда не сойдутся, а будут все время изменяться в ответ на недавно полученные вознаграждения. Как мы уже отмечали, такое поведение даже желательно в нестационарной окружающей среде, а в обучении с подкреплением чаще всего встречаются как раз нестационарные задачи. Кроме того, последовательность размеров шагов, удовлетворяющая условиям (2.7), часто сходится очень медленно или нуждается в серьезной настройке для достижения приемлемой скорости сходимости. Хотя последовательности размеров шагов, отвечающие этим условиям сходимости, зачастую используются в теоретических работах, в приложениях и эмпирических исследованиях они применяются редко.

Упражнение 2.4. Если размеры шагов α_n непостоянны, то оценка Q_n является взвешенным средним ранее полученных вознаграждений с весами, отличными от представленных в формуле (2.6). По аналогии с (2.6) выразите вес каждого предшествующего вознаграждения в терминах последовательности размеров шагов в общем случае. $\square$

Упражнение 2.5. (Требует программирования.) Спланируйте и поставьте эксперимент, демонстрирующий трудности, с которыми сталкиваются методы выборочного среднего в нестационарных задачах. Используйте модифицированную версию 10-рукого испытательного стенда, в которой все $q_*(a)$ вначале равны, а затем вычисляются путем независимых случайных блужданий (скажем, путем прибавления нормально распределенного приращения со средним 0 и стандартным отклонением 0.01 ко всем $q_*(a)$ на каждом шаге). Постройте графики, как на рис. 2.2, для метода ценности действий, в котором используются инкрементно вычисляемые выборочные средние, и еще одного метода, в котором размер шага постоянный, $\alpha = 0.1$. Положите $\varepsilon = 0.1$, а количество шагов возьмите большим, скажем 10 000. $\square$

2.6. ОПТИМИСТИЧЕСКИЕ НАЧАЛЬНЫЕ ЗНАЧЕНИЯ

Все рассмотренные ранее методы в той или иной степени зависят от начальных оценок ценности действий, $Q_1(a)$. На языке статистики это означает, что методы являются *смещенными* в силу начальных оценок. Для методов с выборочным средним смещение исчезает, после того как каждое действие было выбрано хотя бы один раз, но для методов с постоянным α смещение остается, хотя и убывает со временем, как видно из формулы (2.6). На практике такой вид смещения обычно не составляет проблемы, а иногда даже бывает очень полезным. Недостаток заключается в том, что начальные оценки становятся, по существу, набором параметров, которые должен задавать пользователь, пусть даже все они полагаются равными нулю. А достоинство в том, что мы получаем простой способ включить априорную информацию об ожидаемом уровне вознаграждений.

Начальные ценности действий можно использовать также для поощрения исследования. Давайте зададим начальные ценности равными не 0, как в 10-руком

испытательном стенде, а +5. Напомним, что в этой задаче $q_*(a)$ выбираются из нормального распределения со средним 0 и дисперсией 1. Таким образом, начальное значение +5 весьма оптимистично. Но этот оптимизм поощряет методы ценности действий к исследованию. Какое бы действие ни выбрать вначале, вознаграждение окажется меньше начальной оценки; обучаемый склоняется к другим действиям, «разочаровавшись» в полученном вознаграждении. В результате каждое действие будет опробовано несколько раз, прежде чем оценки ценности сойдутся. Система уделяет много времени исследованию, даже если каждый раз выбирает жадное действие.

На рис. 2.3 показано поведение 10-рукого испытательного стенда для жадного метода с $Q_1(a) = +5$ для всех a . Для сравнения показан также ε -жадный метод с $Q_1(a) = 0$. Вначале оптимистический метод работает хуже, потому что больше исследует, но в конечном итоге он выходит на первое место, поскольку со временем объем исследования уменьшается. Мы называем эту технику поощрения исследования *оптимистическими начальными значениями*. Мы считаем ее простым приемом, который может оказаться весьма эффективным в стационарных задачах, но очень далек от подхода к поощрению исследования, полезного в общем случае. Например, он не годится для нестационарных задач, потому что создаваемое им побуждение к исследованию носит принципиально временный характер. Если задача изменяется, порождая потребность в новом исследовании, то этот метод ничем не поможет. Вообще, любой метод, основанный на специальном задании начальных условий, вряд ли окажется полезным в общем нестационарном случае. Начало времён бывает только один раз, поэтому не стоит придавать ему слишком большого значения. Это критическое замечание относится также к методам с выборочным средним, в которых начало времён тоже рассматривается как специальное событие, так что все последующие вознаграждения усредняются с равными весами. Тем не менее все эти методы очень просты, и одного из них – или простой комбинации нескольких – на практике часто бывает достаточно. Далее в данной книге мы нередко будем пользоваться этими простыми способами поощрения исследования.

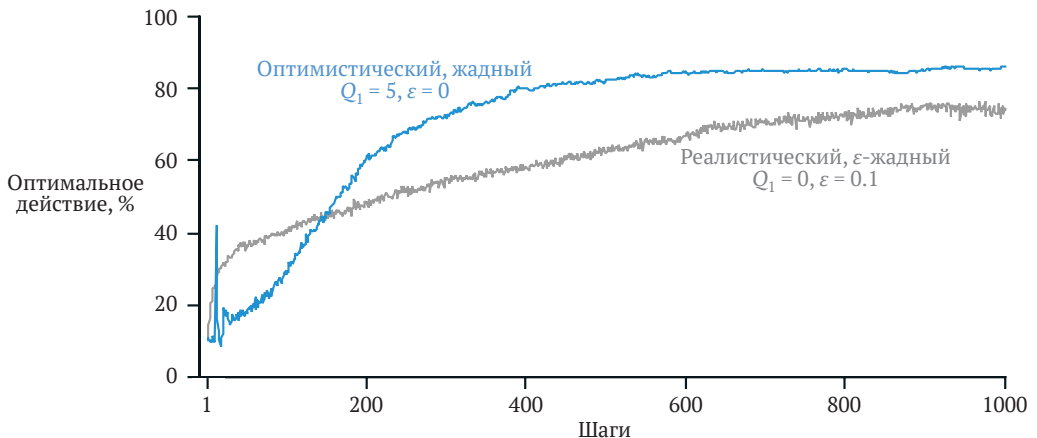


Рис. 2.3 ❖ Демонстрация оптимистических начальных оценок ценности действий на 10-руком испытательном стенде. В обоих методах размер шага постоянный, $\varepsilon = 0.1$

Упражнение 2.6. Таинственные пики. Результаты, показанные на рис. 2.3, должны быть вполне надежными, поскольку получены усреднением по 2000 независимых, случайно выбранных задач о 10-руком бандите. Но тогда откуда берутся осцилляции и пики в начальной части кривой для оптимистического метода? Иными словами, что могло бы заставить этот метод работать существенно хуже или лучше в среднем именно на первых шагах? $\square$

Упражнение 2.7. Несмещенный постоянный размер шага. В большей части этой главы мы использовали выборочные средние для оценки ценности действий, поскольку при этом отсутствует начальное смещение, характерное для постоянного размера шага (см. анализ в формуле (2.6)). Однако выборочное среднее – не вполне удовлетворительное решение, поскольку оно может плохо работать в нестационарных задачах. Можно ли избежать смещения в методах с постоянным размером шага, не отказываясь от их преимуществ в нестационарных задачах? Один из способов – использовать размер шага

$$\beta_n \doteq \alpha / \bar{o}_n \quad (2.8)$$

для обработки n -го вознаграждения за конкретное действие, где $\alpha > 0$ – обычный постоянный размер шага, а $\bar{o}_n$ – рекуррентная последовательность, начинающаяся в 0:

$$\bar{o}_n \doteq \bar{o}_{n-1} + \alpha(1 - \bar{o}_{n-1}) \quad \text{для } n \geq 0, \text{ где } \bar{o}_0 \doteq 0. \quad (2.9)$$

Проведите анализ, подобный формуле (2.6), и покажите, что Q_n – экспоненциальное среднее, взвешенное по степени новизны, без начального смещения. $\square$

2.7. ВЫБОР ДЕЙСТВИЯ, ДАЮЩЕГО ВЕРХНЮЮ ДОВЕРИТЕЛЬНУЮ ГРАНИЦУ

Исследование необходимо, потому что всегда имеется неуверенность в точности оценок ценности действий. Жадные действия выглядят наилучшими на данный момент, но может оказаться, что другие действия на самом деле лучше. ε -жадный выбор действий заставляет пробовать нежадные действия, но без разбора, не отдавая предпочтения почти жадным или особенно недостоверным. Было бы лучше производить выбор среди нежадных действий по их потенциалу оказаться оптимальными, т. е. принимать во внимание как близость их оценок к максимальным, так и недостоверность этих оценок. Эффективный способ добиться этой цели – выбирать такое действие, что

$$A_t \doteq \operatorname{argmax}_a \left[Q_t(a) + c \sqrt{\frac{\ln t}{N_t(a)}} \right], \quad (2.10)$$

где $\ln t$ – натуральный логарифм t (т. е. степень, в которую нужно возвести число $e \approx 2.71828$, чтобы получить t), $N_t(a)$ – сколько раз действие a выбиралось до момента t (знаменатель в формуле (2.1)), а число $c > 0$ контролирует степень исследования. Если $N_t(a) = 0$, то a считается максимизирующим действием.

Идея выбора действия, для которого достигается *верхняя доверительная граница* (ВДГ), заключается в том, что член, содержащий квадратный корень, является мерой недоверности, или дисперсией оценки ценности a . Поэтому максимизируемая величина в некотором роде ограничивает сверху возможную истинную ценность действия a , а c определяет уровень доверия. Можно предположить, что при каждом выборе a недоверность уменьшается: величина $N_t(a)$ увеличивается на 1, а, поскольку она находится в знаменателе, член, описывающий недоверность, уменьшается. С другой стороны, при каждом выборе действия, отличного от a , t увеличивается, но $N_t(a)$ остается прежним; поскольку t находится в числителе, оценка недоверности увеличивается. Благодаря использованию натурального логарифма эти увеличения со временем становятся меньше, но они не ограничены; в конечном итоге будут выбраны все действия, но действия с меньшими оценками ценности или те, которые уже выбирались часто, будут выбираться с уменьшающейся частотой.

Результаты тестирования метода, выбирающего ВДГ-действие, на 10-руком испытательном стенде показаны на рис. 2.4. Этот метод часто дает хорошие результаты, но его труднее, чем ε -жадные методы, обобщить на задачи обучения с подкреплением, отличные от задачи о бандите. Одна из трудностей связана с применением к нестационарным задачам; понадобятся методы более сложные, чем представленные в разделе 2.5. Другая трудность связана с большими пространствами состояний, особенно при использовании приближенных функций, описанных в части II. В таких более сложных постановках идея выбора действия, доставляющего максимум ВДГ, обычно не оправдывает себя.

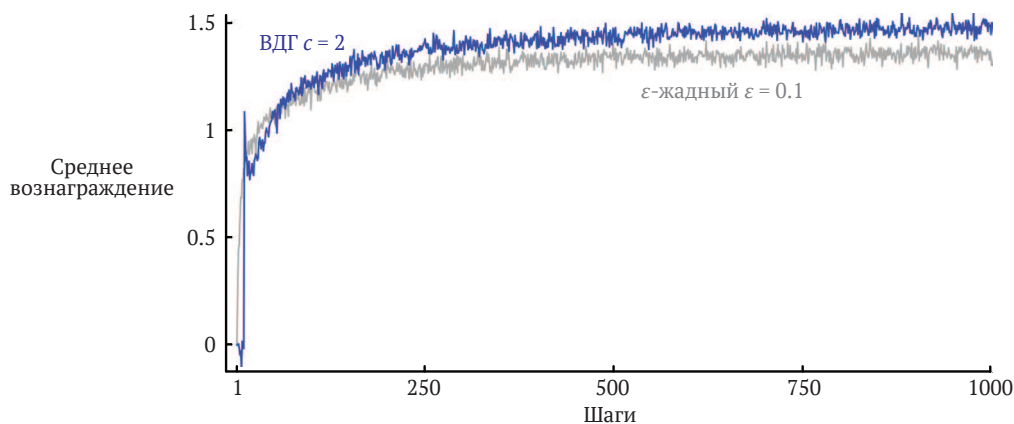


Рис. 2.4 ❖ Среднее вознаграждение за выбор ВДГ-действия на 10-руком испытательном стенде. Как видно, ВДГ, вообще говоря, работает лучше, чем ε -жадный выбор действия, за исключением первых k шагов, когда он случайно выбирает из еще не опробованных действий

Упражнение 2.8. Пики ВДГ. На рис. 2.4 график, соответствующий методу ВДГ, имеет отчетливый пик на 11-м шаге. Почему? Ответ будет считаться полным, если он объясняет, почему вознаграждение увеличивается на 11-м шаге и почему уменьшается на последующих шагах. Указание: если $c = 1$, то пик не так сильно выражен. □

2.8. ГРАДИЕНТНЫЕ АЛГОРИТМЫ БАНДИТА

До сих пор мы рассматривали в этой главе методы, которые оценивают ценность действий и используют полученные оценки для выбора действий. Часто это оказывается хорошим подходом, но он не единственный. В этом разделе мы рассмотрим, как обучить численное предпочтение каждому действию a , которое будем обозначать $H_t(a)$. Чем выше предпочтение, тем чаще выбирается действие, но никакой интерпретации в терминах вознаграждения у предпочтения нет. Важна только относительная предпочтительность двух действий; если прибавить 1000 к предпочтениям всех действий, то это не окажет никакого влияния на вероятности выбора действий, которые вычисляются согласно *распределению softmax* (т. е. *распределению Гиббса* или *Больцмана*) следующим образом:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^k e^{H_t(b)}} \doteq \pi_t(a), \quad (2.11)$$

Мы ввели здесь новое полезное обозначение $\pi_t(a)$ для вероятности выбора действия a в момент t . Первоначально предпочтения всех действий одинаковы (например, $H_1(a) = 0$ для всех a), т. е. у всех действий одинаковая вероятность быть выбранными.

Упражнение 2.9. Покажите, что в случае двух действий распределение softmax совпадает с определяемым логистической, или сигмоидной, функцией, часто используемой в статистике и в искусственных нейронных сетях. $\square$

Для такой постановки существует естественный алгоритм обучения, основанный на идее стохастического градиентного подъема. На каждом шаге после выбора действия A_t и получения вознаграждения R_t предпочтения действий обновляются следующим образом:

$$\begin{aligned} H_{t+1}(A_t) &\doteq H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t)) \quad \text{и} \\ H_{t+1}(a) &\doteq H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a) \quad \text{для всех } a \neq A_t, \end{aligned} \quad (2.12)$$

где $\alpha > 0$ – размер шага, а $\bar{R}_t \in \mathbb{R}$ – среднее по всем вознаграждениям до момента t включительно, которое можно вычислить инкрементно, как описано в разделе 2.4 (или 2.5 для нестационарной задачи). Член $\bar{R}_t$ служит базой, с которой сравнивается вознаграждение. Если вознаграждение больше базового, то вероятность выбрать A_t в будущем возрастает, а если меньше, то убывает. Невыбранные действия двигаются в противоположном направлении.

На рис. 2.5 показаны результаты градиентного алгоритма бандита, полученные с помощью варианта 10-рукого испытательного стенда, в котором истинные ожидаемые вознаграждения выбирались из нормального распределения со средним +4, а не 0 (и с единичной дисперсией, как и раньше). Такой сдвиг вверх всех вознаграждений не оказывает никакого влияния на градиентный алгоритм из-за члена базового вознаграждения, который мгновенно адаптируется к новому уровню. Но если бы мы опустили базу (т. е. если бы $\bar{R}_t$ в формуле (2.12) был равен нулю), то качество метода оказалось бы существенно хуже, что хорошо видно на рисунке.

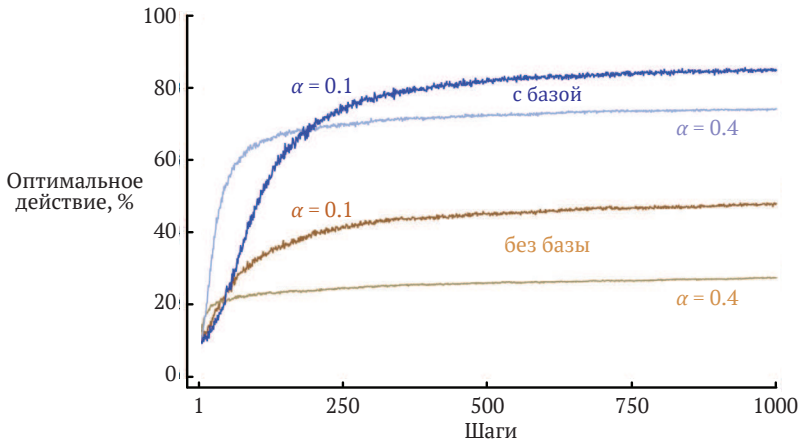


Рис. 2.5 ❖ Среднее качество градиентного алгоритма бандита с базой и без нее. Получено на 10-руком испытательном стенде, в котором $q_*(a)$ выбиралось в окрестности +4, а не нуля

Градиентный алгоритм бандита как алгоритм стохастического градиентного подъема

Глубже разобраться в градиентном алгоритме бандита можно, если интерпретировать его как стохастическую аппроксимацию градиентного подъема. В точном алгоритме градиентного подъема предпочтение каждого действия $H_t(a)$ увеличивалось бы пропорционально влиянию приращения на качество:

$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)}, \quad (2.13)$$

где мерой качества является ожидаемое вознаграждение:

$$\mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x),$$

а мерой влияния приращения — *частная производная* этой меры качества по предпочтению действия. Конечно, в нашем случае невозможно реализовать градиентный подъем точно, потому что, по предположению, мы не знаем величины $q_*(x)$, но на самом деле обновления в алгоритме (2.12) равны (2.13) в отношении ожидаемой ценности, так что мы получили алгоритм *стохастического градиентного подъема*. Доказывающие это вычисления требуют лишь знания основ математического анализа, но состоят из нескольких шагов. Сначала внимательно взглянем на точный градиент качества:

$$\begin{aligned}
\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)} \left[\sum_x \pi_t(x) q_*(x) \right] \\
&= \sum_x q_*(x) \frac{\partial \pi_t(x)}{\partial H_t(a)} \\
&= \sum_x (q_*(x) - B_t) \frac{\partial \pi_t(x)}{\partial H_t(a)},
\end{aligned}$$

где член B_t , называемый *базой*, может быть любым скаляром, не зависящим от x . Мы можем включить сюда базу, не нарушая равенства, поскольку сумма градиентов по всем действиям равна нулю, $\sum_x \frac{\partial \pi_t(x)}{\partial H_t(a)}$ – когда $H_t(a)$ изменяется, вероятности одних действий увеличиваются, других – уменьшаются, но сумма изменений должна быть равна нулю, потому что сумма вероятностей всегда равна 1.

Далее умножим каждый член суммы на $\pi_t(x)/\pi_t(x)$:

$$\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} = \sum_x \pi_t(x) (q_*(x) - B_t) \frac{\partial \pi_t(x)}{\partial H_t(a)} / \pi_t(x).$$

Теперь равенство имеет вид формулы математического ожидания, где вычисляется сумма произведений всех возможных значений x случайной величины A_t на вероятности этих значений. Таким образом:

$$\begin{aligned}
&= \mathbb{E} \left[(q_*(A_t) - B_t) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} / \pi_t(A_t) \right] \\
&= \mathbb{E} \left[(R_t - \bar{R}_t) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} / \pi_t(A_t) \right],
\end{aligned}$$

где мы выбрали базу $B_t = \bar{R}_t$ и подставили R_t вместо $q_*(A_t)$, что возможно, поскольку $\mathbb{E}[R_t | A_t] = q_*(A_t)$. Вскоре мы установим, что $\frac{\partial \pi_t(x)}{\partial H_t(a)} = \pi_t(x)(\mathbb{1}_{a=x} - \pi_t(a))$,

где $\mathbb{1}_{a=x}$ определена как 1, если $a = x$, и 0 в противном случае. Пока примем это на веру, тогда имеем:

$$\begin{aligned}
&= \mathbb{E}[(R_t - \bar{R}_t) \pi_t(A_t) (\mathbb{1}_{a=A_t} - \pi_t(a)) / \pi_t(A_t)] \\
&= \mathbb{E}[(R_t - \bar{R}_t) (\mathbb{1}_{a=A_t} - \pi_t(a))].
\end{aligned}$$

Напомним, что наш план заключался в том, чтобы записать градиент качества в виде математического ожидания чего-то, из чего можно делать выборку на каждом шаге, как мы только что сделали, а затем производить на каждом шаге обновление, пропорциональное выбранному значению. Подстановка выборки математического ожидания выше вместо градиента качества в формулу (2.13) дает:

$$H_{t+1}(a) = H_t(a) + \alpha(R_t - \bar{R}_t)(\mathbb{1}_{a=A_t} - \pi_t(a)) \quad \text{для всех } a,$$

и, как легко видеть, эта формула эквивалентна нашему исходному алгоритму (2.12).

Остается только показать, что $\frac{\partial \pi_t(x)}{\partial H_t(a)} = \pi_t(x)(\mathbb{1}_{a=x} - \pi_t(a))$, как мы и предположили. Напомним стандартное правило дифференцирования частного:

$$\frac{\partial}{\partial x} \left[\frac{f(x)}{g(x)} \right] = \frac{\frac{\partial f(x)}{\partial x} g(x) - f(x) \frac{\partial g(x)}{\partial x}}{g(x)^2}.$$

Применяя его, можно записать:

$$\begin{aligned} \frac{\partial \pi_t(x)}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)} \pi_t(x) \\ &= \frac{\partial}{\partial H_t(a)} \left[\frac{e^{H_t(x)}}{\sum_{y=1}^k e^{H_t(y)}} \right] \\ &= \frac{\frac{\partial e^{H_t(x)}}{\partial H_t(a)} \sum_{y=1}^k e^{H_t(y)} - e^{H_t(x)} \frac{\partial \sum_{y=1}^k e^{H_t(y)}}{\partial H_t(a)}}{\left(\sum_{y=1}^k e^{H_t(y)} \right)^2} \quad \left(\text{в силу правила дифференцирования частного} \right) \\ &= \frac{\mathbb{1}_{a=x} e^{H_t(x)} \sum_{y=1}^k e^{H_t(y)} - e^{H_t(x)} e^{H_t(a)}}{\left(\sum_{y=1}^k e^{H_t(y)} \right)^2} \quad \left(\text{так как } \frac{\partial e^x}{\partial x} = e^x \right) \\ &= \frac{\mathbb{1}_{a=x} e^{H_t(x)}}{\sum_{y=1}^k e^{H_t(y)}} - \frac{e^{H_t(x)} e^{H_t(a)}}{\left(\sum_{y=1}^k e^{H_t(y)} \right)^2} \\ &= \mathbb{1}_{a=x} \pi_t(x) - \pi_t(x) \pi_t(a) \\ &= \pi_t(x)(\mathbb{1}_{a=x} - \pi_t(a)). \end{aligned}$$

Мы только что показали, что ожидаемое обновление в градиентном алгоритме бандита равно градиенту ожидаемого вознаграждения, и, следовательно, этот алгоритм является примером стохастического градиентного подъема. Отсюда вытекает, что алгоритм обладает свойством устойчивой сходимости.

Отметим, что мы не требовали от эталона вознаграждения никаких свойств, кроме независимости от выбранного действия. Мы могли бы положить его равным 0 или 1000, и алгоритм все равно остался бы примером стохастического градиентного подъема. Выбор базы не влияет на математическое ожидание обновления, но влияет на его дисперсию, а значит, на скорость сходимости (как видно, например, на рис. 2.5). Выбирать в этом качестве среднее вознаграждение, возможно, не самое лучшее решение, но оно простое и хорошо работает на практике.

2.9. Ассоциативный поиск (контекстуальные бандиты)

До сих пор в этой главе мы рассматривали только неассоциативные задачи, т. е. такие, в которые нет нужды ассоциировать разные действия с разными ситуациями. В таких задачах обучаемый либо пытается найти единственное лучшее действие, если задача стационарна, либо проследить, как лучшее действие изменяется со временем, если нестационарна. Но в общей задаче обучения с подкреплением может быть несколько ситуаций, и цель заключается в обучении стратегии: отображении ситуаций на действия, оптимальные в этих ситуациях. Для постановки задачи в общем виде мы коротко рассмотрим простейший способ обобщения неассоциативных задач на ассоциативный контекст.

В качестве примера предположим, что имеется несколько разных задач о k -руких бандитах и что на каждом шаге нам предлагается одна из них, выбранная наугад. Таким образом, на каждом шаге задача о бандите случайным образом изменяется. Для нас это выглядело бы как одна нестационарная задача о k -руком бандите, в которой истинные ценности действий случайным образом изменяются от шага к шагу. Можно было бы попробовать применить к ней любой из описанных в этой главе методов решения нестационарных задач, но если истинные ценности действий изменяются достаточно быстро, то эти методы будут работать плохо. Предположим, однако, что вместе с предлагаемой задачей о бандите нам сообщается некая информация о том, что отличает ее от других (но не о ценности действий). Быть может, мы имеем дело с настоящим игровым автоматом, который меняет цвет экрана при изменении ценности действий. Теперь можно обучить стратегию, ассоциирующую с каждой задачей, идентифицируемой цветом экрана, оптимальное действие в этой задаче, например: если экран красный, то выбирать рычаг 1, а если зеленый, то рычаг 2. Такая стратегия обычно позволяет достичь гораздо лучшего результата, чем при полном отсутствии информации, отличающей одну задачу от другой.

Это пример *задачи ассоциативного поиска*. Она названа так, потому что сочетает как обучение методом проб и ошибок, направленное на поиск наилучших действий, так и *ассоциирование* этих действий с ситуациями, в которых они являются наилучшими. В литературе задачи ассоциативного поиска часто называются *контекстуальными бандитами*. Они лежат посередине между задачей о k -руком бандите и полной задачей обучения с подкреплением. Они похожи на полную задачу обучения с подкреплением тем, что включают обучение стратегии, а на наш вариант задачи о k -руком бандите – тем, что каждое действие влияет только на немедленное вознаграждение. Если действиям разрешено влиять не только на вознаграждение, но и на *следующую ситуацию*, то мы получаем полную задачу обучения с подкреплением. Мы представим ее в следующей главе и будем рассматривать ее разновидности на протяжении всей книги.

Упражнение 2.10. Пусть имеется задача о 2-руком бандите, в которой истинные ценности действий случайным образом меняются от шага к шагу. Точнее, предположим, что на любом временном шаге истинные ценности действий 1 и 2 равны соответственно 0.1 и 0.2 с вероятностью 0.5 (случай А) и 0.9 и 0.8 с вероятностью

0.5 (случай В). Если мы не можем сказать, какой случай имеет место на данном шаге, то каково максимально возможное математическое ожидание успеха и как следует себя вести, чтобы достичь его? Теперь предположим, что на каждом шаге нам сообщают, какой случай, А или В, имеет место (хотя истинной ценности действий мы по-прежнему не знаем). Это задача ассоциативного поиска. Какого максимального математического ожидания успеха можно достичь в этой задаче и как следует себя вести для его достижения? □

2.10. РЕЗЮМЕ

В этой главе мы рассказали о нескольких простых способах нахождения компромисса между исследованием и использованием. ϵ -жадные методы изредка выбирают действие случайным образом, тогда как ВДГ-методы производят выбор детерминированно, но обеспечивают исследование, отдавая на каждом шаге небольшое предпочтение действиям, которые до этого выбирались меньшее число раз. Градиентные алгоритмы бандита оценивают не ценности действий, а их предпочтения, и благоволят более предпочтительным действиям, применяя распределение вероятностей softmax. Простой прием – оптимистическая инициализация оценок позволяет даже жадным методам вволю насладиться исследованием.

Естественно спросить, какие из этих методов наилучшие. В общем случае это трудный вопрос, но мы, безусловно, можем прогнать их на 10-руком испытательном стенде и сравнить результаты. Сложность в том, что у всех у них есть параметр, и, чтобы сравнение было осмысленным, мы должны рассматривать качество метода как функцию от его параметра. На приведенных выше графиках была показана временная кривая обучения для каждого алгоритма и нескольких значений параметров. Если бы мы строили кривые обучения для всех алгоритмов и значений параметра, то график получился бы слишком сложным и громоздким, так что провести сравнение было бы затруднительно. Вместо этого построим сводную кривую обучения, усреднив значения за 1000 шагов; это значение пропорционально площади под кривой обучения. На рис. 2.6 данный показатель представлен для различных алгоритмов решения задачи о бандите, описанных в этой главе; каждый алгоритм представлен функцией от своего параметра, которые показаны на единой шкале по оси x . Такой вид графика называется *параметрическим исследованием*. Заметим, что соседние значения параметров отличаются множителем 2, т. е. шкала логарифмическая. Обратите также внимание на характерную форму кривых качества всех алгоритмов – в виде перевернутой буквы U; каждый алгоритм достигает наилучших результатов при каком-то промежуточном значении параметра, не слишком большом и не слишком малом. Оценивая метод, мы должны смотреть не только на то, как он ведет себя при оптимальном значении параметра, но и на то, насколько он чувствителен к изменению параметра. Все описанные алгоритмы не слишком чувствительны, они хорошо работают в широком диапазоне параметров, различающихся чуть ли не на порядок. В целом же для этой задачи метод ВДГ представляется наилучшим.

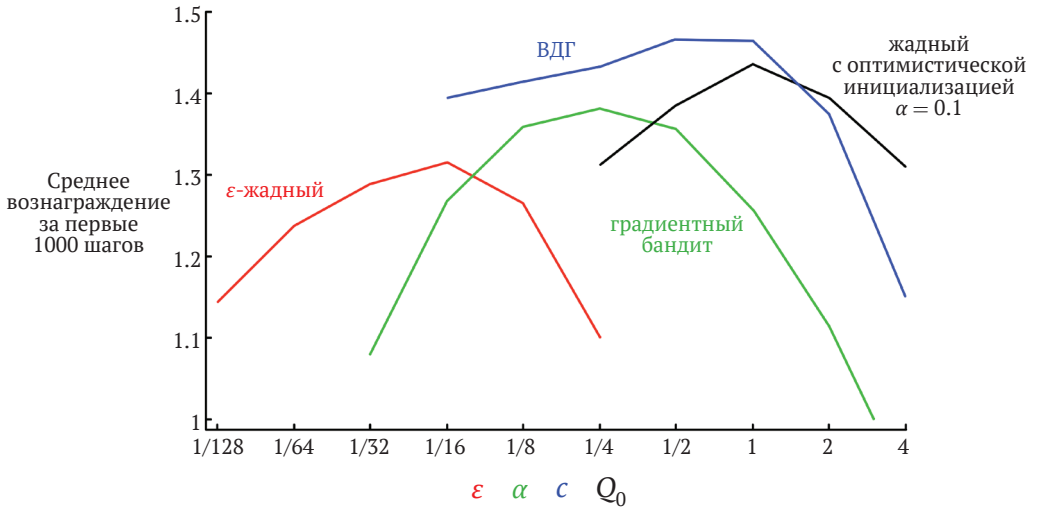


Рис. 2.6 ❖ Параметрическое исследование различных алгоритмов решения задачи о бандите, описанных в этой главе. Каждая точка – это среднее вознаграждение, полученное алгоритмом за 1000 шагов при определенном значении параметра

Несмотря на простоту, методы, представленные в этой главе, на наш взгляд, можно с полным правом считать современными. Есть более изощренные методы, но их сложность и допущения делают их практически неприменимыми к полной задаче обучения с подкреплением, которая нас занимает. Начиная с главы 5 мы будем описывать методы обучения для решения полной задачи обучения с подкреплением, в которых в качестве составной части используются простые методы, изученные в этой главе.

Хотя рассмотренные выше простые методы – лучшее, что мы можем сделать в настоящий момент, они далеки от вполне удовлетворительного решения проблемы о балансе между исследованием и использованием.

Один из хорошо изученных подходов к нахождению баланса в задачах о k -руком бандите состоит в том, чтобы вычислить специальный вид ценности действий, называемый *индексом Гиттинса*. В некоторых важных частных случаях это вычисление практически реализуемо и прямо приводит к оптимальным решениям, хотя требует полной информации об априорном распределении возможных задач, которое в общем случае недоступно. Кроме того, ни с теоретической, ни с вычислительной точки зрения этот подход не обобщается на полную задачу обучения с подкреплением.

Подход на основе индекса Гиттинса – это пример байесовских методов, в которых предполагается известным начальное распределение ценностей действий, а затем это распределение обновляется на каждом шаге (в предположении, что истинные ценности действий стационарны). В общем случае вычисления, производимые при обновлении, могут быть очень сложными, но для некоторых специальных распределений (называемых *сопряженными априорными распределениями*) они оказываются простыми. Тогда мы можем на каждом шаге выбирать действия согласно их апостериорной вероятности оказаться лучшими. Этот метод, который иногда называют *апостериорной выборкой*, или *выборкой Томпсона*, зачастую дает

результаты, не уступающие лучшим методам, не использующим распределение, из числа описанных в этой главе.

В байесовской постановке теоретически даже можно вычислить *оптимальный* баланс между исследованием и использованием. Для каждого возможного действия можно вычислить вероятность каждого возможного немедленного вознаграждения и результирующие апостериорные распределения вероятностей ценности действий. Это изменяющееся распределение становится *информационным состоянием* задачи. На горизонте, скажем, 1000 шагов можно рассмотреть все возможные действия, все возможные результирующие вознаграждения, все возможные следующие действия, следующие вознаграждения и т. д. для всей тысячи шагов. В этих предположениях можно определить вознаграждения и вероятности на любой возможной цепочке событий, так что останется только выбрать лучшую. Но дерево возможностей растет чрезвычайно быстро; даже при наличии всего двух действий и двух вознаграждений в дереве было бы 22 000 листьев. В общем случае точно выполнить такое гигантское вычисление невозможно, но, быть может, удалось бы найти эффективную аппроксимацию. При таком подходе задача о бандите, по сути дела, превратилась бы в полную задачу обучения с подкреплением. В конечном итоге нам, возможно, удастся использовать приближенные методы обучения с подкреплением типа представленных во второй части книги для реализации подобного подхода к поиску оптимального решения. Но это тема для научных исследований, которая выходит за рамки введения.

Упражнение 2.11. (Требует программирования.) Создайте рисунок, аналогичный рис. 2.6, только для нестационарного случая, описанного в упражнении 2.5. Включите ε -жадный алгоритм с постоянным размером шага и $\varepsilon = 0.1$. Количество шагов пусть будет равным 200 000, а в качестве показателя качества каждого алгоритма и значения параметра возьмите среднее вознаграждение за последние 100 000 шагов. □

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

- 2.1 Задачи о бандитах изучались в статистике, технических дисциплинах и психологии. В статистике они относятся к теме «последовательного планирования эксперимента», начатой в работах Thompson (1933, 1934) и Robbins (1952) и исследованной Беллманом (Bellman, 1956). В работе Berry and Fristedt (1985) имеется широкий обзор задач о бандитах с точки зрения статистики. В работе Narendra and Thathachar (1989) задачи о бандитах рассматриваются с инженерной точки зрения и приводится хорошее обсуждение различных теоретических традиций, в которых они играют главную роль. В психологии задачи о бандитах сыграли роль в статистической теории обучения (см., например, Bush and Mosteller, 1955; Estes, 1950).

Термин «жадный» часто употребляется в литературе по эвристическому поиску (например, Pearl, 1984). Конфликт между исследованием и использованием известен в технике автоматического управления как конфликт между идентификацией (или оцениванием) и управлением (например, Witten, 1976b). В работе Feldbaum (1965) он назван *задачей двойного управления*; под

этим понимается, что при попытке управлять системой в условиях неопределенности необходимо одновременно решать обе задачи, идентификации и управления. Обсуждая некоторые аспекты генетических алгоритмов, Холланд (Holland, 1975) подчеркивал важность этого конфликта, называя его конфликтом между необходимостью использовать имеющуюся информацию и необходимостью получать новую.

- 2.2 Методы ценности действий для нашей задачи о k -руком бандите впервые были предложены в работе Thathachar and Sastry (1985). В литературе по самообучающимся автоматам их часто называют *алгоритмами оценивания*. Термин *ценность действия* ввел в обращение Уоткинс (Watkins, 1989). Возможно, он же (Watkins, 1989, p. 187) первым использовал ε -жадные методы, но идея настолько проста, что, вероятно, кто-то прибегал к ней и раньше.
- 2.4–5 Этот материал относится к общей категории стохастических итеративных алгоритмов, которая хорошо освещена в работе Bertsekas and Tsitsiklis (1996).
- 2.6 Оптимистическая инициализация использовалась в обучении с подкреплением в работе Sutton (1996).
- 2.7 Ранние примеры использования оценок верхней доверительной границы для выбора действий см. в работах Lai and Robbins (1985), Kaelbling (1993b) и Agrawal (1995). Представленный здесь ВДГ-алгоритм в литературе называется UCB1 и впервые был изложен в работе Auer, Cesa-Bianchi, and Fischer (2002).
- 2.8 Градиентные алгоритмы бандита – частный случай градиентных алгоритмов обучения с подкреплением, которые впервые были описаны в работе Williams (1992), а впоследствии развились в алгоритмы исполнитель–критик и градиента стратегии, рассматриваемые в этой книге ниже. На наше изложение повлияли идеи Баларамана Равиндрана (частное сообщение). Дальнейшее обсуждение выбора базы приведено там же и в работах Green-smith, Bartlett, and Baxter (2002, 2004) и Dick (2015). Первые систематические исследования подобных алгоритмов были выполнены в работе Sutton (1984).

Термин *softmax* для правила выбора действия (2.11) предложен в работе Bridle (1990). Само правило, видимо, было предложено в работе Luce (1959).

- 2.9 Термин *ассоциативный поиск* и соответствующая задача введены в работе Barto, Sutton, and Brouwer (1981). Термин *ассоциативное обучение с подкреплением* также употреблялся для обозначения ассоциативного поиска (Barto, and Anandan, 1985), но мы предпочитаем использовать его как синоним *полной задачи обучения с подкреплением* (как в работе Sutton, 1984). (И, как мы отмечали, в современной литературе для этой задачи также употребляется термин «контекстуальные бандиты»). Отметим, что закон эффекта Торндайка (см. главу 1) описывает ассоциативный поиск как формирование ассоциативных связей между ситуациями (состояниями) и действиями. В терминологии оперантного, или инструментального, обусловливания (см., например, Skinner, 1938) дифференциальным стимулом называется стимул, который сигнализирует о контингентности подкрепления. В наших

терминах различные дифференциальные стимулы соответствуют различным состояниям.

- 2.10** Беллман (Bellman, 1956) первым показал, как можно использовать динамическое программирование для вычисления оптимального баланса между исследованием и использованием в байесовской постановке задачи. Подход на основе индекса Гиттинса изложен в работе Gittins, and Jones (1974). В работе Duff (1995) показано, как можно обучить индексы Гиттинса для задач о бандитах посредством обучения с подкреплением. В обзоре Kumar (1985) приводится хорошее обсуждение байесовских и небайесовских подходов к этим проблемам. Термин *информационное состояние* заимствован из литературы по частично наблюдаемым МППР; см., например, Lovejoy (1991).

Другие теоретические работы посвящены эффективности исследования, особенно вопросу о том, как быстро алгоритм может приблизиться к оптимальной стратегии принятия решений. Один из способов формализовать эффективность исследования – адаптировать к обучению с подкреплением понятие *выборочной сложности* алгоритма обучения с учителем; в этом контексте так называется количество обучающих примеров, необходимых алгоритму для достижения желаемой верности обучения целевой функции. Выборочная сложность исследования в алгоритме обучения с подкреплением определяется как количество временных шагов, в которых алгоритм не выбирает почти оптимальные действия (Kakade, 2003). В обзоре Li (2012), посвященном теоретическим подходам к эффективности исследования, обсуждается этот и некоторые другие подходы. Подробное современное обсуждение выборки Томпсона можно найти в работе Russo et al. (2018).

Глава 3

Конечные марковские процессы принятия решений

В этой главе мы формально поставим задачу о конечных марковских процессах принятия решений, или конечных МППР, которую будем пытаться решить в оставшейся части книги. В этой задаче, как и в задаче о бандите, присутствует оценочная обратная связь, но у нее имеется и ассоциативная сторона – выбор различных действий в разных ситуациях. МППР – классическая формализация последовательного принятия решений, когда от действий зависит не только немедленное вознаграждение, но и последующие ситуации, или состояния, а через них и будущие вознаграждения. Таким образом, в МППР присутствует отложенное вознаграждение и необходимость искать компромисс между немедленным и отложенным вознаграждениями. Если в задаче о бандите мы оценивали ценность $q_*(a)$ каждого действия a , то в МППР мы оцениваем ценность $q_*(s, a)$ действия a в состоянии s или ценность $v_*(s)$ каждого состояния при условии оптимального выбора действия. Эти зависящие от состояния величины необходимы для правильного распределения поощрения между отдаленными последствиями выбора тех или иных действий.

МППР – математически идеализированная форма задачи обучения с подкреплением, для которой можно высказывать точные теоретические утверждения. Мы познакомимся с основными элементами математической постановки задачи: доходом, функциями ценности и уравнениями Беллмана. Мы расскажем о широком круге приложений, которые можно сформулировать как задачи о конечных МППР. Как и всюду в искусственном интеллекте, существует противоречие между широтой применимости и математической разрешимостью. В этой главе мы поясним суть этого противоречия и обсудим, какие компромиссы и проблемы из него вытекают. В главе 17 будут рассмотрены некоторые способы расширения обучения с подкреплением за пределы МППР.

3.1. ИНТЕРФЕЙС МЕЖДУ АГЕНТОМ И ОКРУЖАЮЩЕЙ СРЕДОЙ

МППР предназначены для прямолинейной формулировки задачи обучения в результате взаимодействия для достижения цели. Сторона, которая обучается и принимает решения, называется *агентом*. Сторона, с которой агент взаимодействует, включающая в себя все, что находится вне агента, называется *окружающей средой*,

или просто *средой*. Обе стороны взаимодействуют непрерывно – агент выбирает действия, а среда реагирует на эти действия и предлагает агенту новые ситуации¹. Среда также генерирует *вознаграждения* – числовые значения, которые агент стремится со временем максимизировать посредством выбора действий.

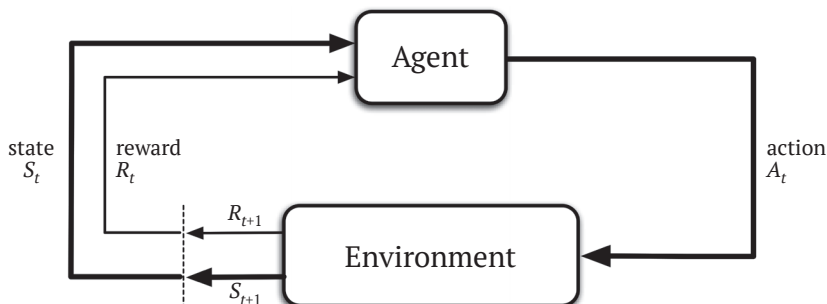


Рис. 3.1 ❖ Взаимодействие между агентом и окружающей средой в марковском процессе принятия решений

Точнее, агент и среда взаимодействуют на каждом шаге дискретной последовательности временных шагов, $t = 0, 1, 2, 3, \dots$ ². На каждом временном шаге t агент получает некоторое представление *состояния* окружающей среды $S_t \in \mathcal{S}$ и, исходя из него, выбирает действие $A_t \in \mathcal{A}(s)$ ³. На следующем шаге агент, отчасти как последствие своего действия, получает числовое вознаграждение $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ и оказывается в новом состоянии S_{t+1} ⁴. Таким образом, МППР и агент совместно порождают последовательность, или траекторию, которая начинается так:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \dots \quad (3.1)$$

В *конечном* МППР множества состояний, действий и вознаграждений ($\mathcal{S}$, $\mathcal{A}$ и $\mathcal{R}$) содержат конечное число элементов. Тогда для случайных величин R_t и S_t корректно определены дискретные распределения вероятностей, зависящие только от предшествующего состояния и действия. То есть для конкретных значений этих случайных величин, $s' \in \mathcal{S}$ и $r \in \mathcal{R}$, определена вероятность возникновения этих значений в момент t , при условии известных значений предшествующего состояния и действия:

$$p(s', r | s, a) \doteq \Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\}, \quad (3.2)$$

¹ Мы употребляем термины *агент*, *окружающая среда* и *действие* вместо привычных инженерам *устройство управления (контроллер)*, *управляемая система (предприятие)* и *управляющий сигнал*, поскольку они понятны более широкой аудитории.

² Мы ограничиваемся дискретным временем, чтобы упростить изложение, но многие идеи можно обобщить на непрерывный случай (см., например, Bertsekas and Tsitsiklis, 1996; Doya, 1996).

³ Чтобы упростить обозначения, мы иногда будем предполагать частный случай, когда множество действий одинаково во всех состояниях, и будем обозначать его просто $\mathcal{A}$.

⁴ Мы обозначаем вознаграждение, полученное вследствие действия A_t , не R_t , а R_{t+1} , чтобы подчеркнуть, что следующее вознаграждение и следующее состояние, R_{t+1} и S_{t+1} , определяются совместно. К сожалению, в литературе широко употребляются оба соглашения.

для всех $s', s \in \mathcal{S}, r \in \mathcal{R}$ и $a \in \mathcal{A}(s)$. Функция p определяет динамику МППР. Точка над знаком равенства в этой формуле напоминает, что это определение (в данном случае функции p), а не факт, следующий из предыдущих определений. Функция динамики $p: \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ – обыкновенная детерминированная функция четырех аргументов. Знак « $\doteq$ » в середине обычно употребляется для обозначения условной вероятности, но здесь он просто напоминает, что p задает распределение вероятностей для каждого выбора s и a , т. е. что

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a) = 1 \quad \text{для всех } s \in \mathcal{S}, a \in \mathcal{A}(s). \quad (3.3)$$

В марковском процессе принятия решений вероятности, задаваемые функцией p , полностью характеризуют динамику окружающей среды. То есть вероятность каждого возможного значения S_t и R_t зависит только от непосредственно предшествующего состояния и действия, S_{t-1} и A_{t-1} , и не зависит от всех предыдущих состояний и действий. Лучше считать это ограничением не на процесс принятия решений, а на *состояние*. Состояние должно включать информацию обо всех аспектах прошлых взаимодействий агента со средой, которые существенны для будущего. Если это так, то говорят, что состояние обладает *марковским свойством*. В этой книге мы будем предполагать, что марковское свойство имеет место, хотя начиная со второй части будем рассматривать приближенные методы, которые от него не зависят, а в главе 17 поговорим о том, как можно обучить марковское состояние и сконструировать его из немарковских наблюдений.

Зная функцию динамики p с четырьмя аргументами, можно вычислить все, что нужно знать о среде, например *вероятности перехода состояний* (которые мы будем обозначать, допуская некоторую вольность в нотации, как функцию трех аргументов $p: \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$):

$$p(s' | s, a) \doteq \Pr\{S_t = s' | S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r | s, a). \quad (3.4)$$

Мы также можем вычислить ожидаемые вознаграждения для пар состояние–действие как функцию двух аргументов $r: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$:

$$r(s, a) \doteq \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r | s, a), \quad (3.5)$$

а ожидаемые вознаграждения для троек состояние–действие–следующее состояние как функцию трех аргументов $r: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$:

$$r(s, a, s') \doteq \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r | s, a)}{p(s' | s, a)}. \quad (3.6)$$

В этой книге обычно употребляется функция p с четырьмя аргументами (3.2), но иногда удобны другие три.

Формализм МППР является абстрактным и гибким, он допускает применение ко многим задачам самыми разными способами. Например, временные шаги не обязательно должны быть фиксированными промежутками реального времени, они могут относиться к произвольным последовательным этапам принятия решений и действий. Действия могут быть низкоуровневыми управляющими воздействиями, например подачей напряжения на моторы руки робота, или высо-

коуровневыми решениями, например позавтракать или идти на занятия. Также и состояния могут принимать разные формы. Они могут полностью определяться низкоуровневым восприятием, например прямым считыванием датчиков, или быть более высокоуровневыми и абстрактными, например символическими описаниями предметов в комнате. Часть того, что образует состояние, может быть основана на воспоминаниях о прошлых ощущениях или даже быть чисто умозрительным либо субъективным. Например, агент может находиться в состоянии неуверенности в том, где находится предмет, или удивления в некотором четко определенном смысле. Действия также могут быть полностью умозрительными или вычислительными. Например, некоторые действия могут управлять тем, о чем агент решить подумать или на чем сфокусирует внимание. Вообще говоря, действиями могут быть любые решения, которые мы хотим научиться принимать, а состояниями – все, что мы можем узнать и что может пригодиться для совершения действий.

В частности, граница между агентом и окружающей средой, как правило, не совпадает с физической границей корпуса робота или тела животного. Обычно граница проходит ближе к агенту. Например, моторы и механические сочленения робота и его аппаратные датчики обычно следует рассматривать как часть среды, а не агента. Аналогично, если применить формализм МППР к человеку или животному, то мышцы, скелет и органы чувств следует рассматривать как часть среды. Вознаграждения тоже, скорее всего, вычисляются внутри физических тел естественных и искусственных самообучающихся систем, но считаются внешними по отношению к агенту.

Общее правило, которому мы следуем, таково: все, что не может быть произвольным образом изменено агентом, считается находящимся вне его, а значит, частью окружающей среды. Мы не предполагаем, что все, относящееся к среде, агенту неизвестно. Например, агент часто знает о том, как вознаграждения вычисляются в виде функций от его действий и состояний, в которых они предприняты. Но мы всегда рассматриваем вычисление вознаграждения как процесс, внешний по отношению к агенту, потому что он определяет стоящую перед агентом задачу, и, следовательно, агент не способен изменять его произвольным образом. На самом деле в некоторых случаях агент может знать *все* о том, как работает среда, и тем не менее стоять перед трудной задачей обучения с подкреплением. Ведь знаем же мы во всех деталях, как устроены головоломки типа кубика Рубика, но решить их все равно не можем. Граница между агентом и средой полагает пределы возможностям *абсолютного контроля* со стороны агента, но не его знаниям.

Граница между агентом и средой может проходить в разных местах в зависимости от целей. В случае сложного робота много различных агентов должны работать совместно, каждый в своих границах. Например, один агент может принимать высокоуровневые решения, которые являются частью состояний, воспринимаемых низкоуровневым агентом, который реализует высокоуровневые решения. На практике граница между агентом и средой определяется после того, как выбраны конкретные состояния, действия и вознаграждения и, следовательно, ясна конкретная задача о принятии решений.

Формализм МППР – важная абстракция задачи обучения целеустремленного агента в процессе взаимодействия. Он говорит, что каковы бы ни были детали механизмов восприятия, памяти и управления и какой бы цели агент ни стремился

достичь, любая задача обучения целенаправленному поведению может быть сведена к трем сигналам, которыми агент обменивается с окружающей средой: для представления выбора, сделанного агентом (действия), для представления оснований для этого выбора (состояния) и для определения цели агента (вознаграждения). Этого формализма не всегда достаточно для полезного представления всех задач обучения принятию решений, но он доказал свою широкую применимость и полезность.

Разумеется, конкретные состояния и действия сильно зависят от задачи, а их представление может оказывать большое влияние на эффективность. В обучении с подкреплением, как и в других видах обучения, выбор представления – скорее искусство, чем наука, по крайней мере в настоящее время. В этой книге мы дадим некоторые рекомендации и приведем примеры хороших представлений состояний и действий, но в основном будем говорить об общих принципах обучения тому, как себя вести, после того как представления выбраны.

Пример 3.1. Биореактор. Рассмотрим применение обучения с подкреплением к определению температур и скоростей перемешивания в биореакторе в последовательные моменты времени (биореактор – это большой чан, заполненный питательными веществами и бактериями с целью производства полезных химикатов). В таком приложении действиями могут быть целевая температура и скорость перемешивания, передаваемые низкоуровневым управляющим системам, которые, в свою очередь, активируют нагревательные элементы и моторы для достижения заданных показателей. Состояниями, наверное, будут показания термомпары и других датчиков, возможно, подвергнутые фильтрации и поступающие с задержкой, а также символические входные данные, описывающие ингредиенты в чане и конечное вещество. Вознаграждениями могут быть дискретные результаты измерения скорости производства полезного химиката биореактором. Отметим, что здесь каждое состояние является списком, или вектором показаний датчиков и символических входных данных, а каждое действие – вектор, состоящий из целевой температуры и скорости перемешивания. Для задач обучения с подкреплением типична ситуация, когда состояния и действия представлены в структурированной форме. С другой стороны, вознаграждение – всегда одиночное число. ■

Пример 3.2. Перегрузочный робот. Рассмотрим применение обучения с подкреплением к управлению движением роботизированной руки, выполняющей повторяющуюся задачу «взять и положить». Если мы хотим обучить робота быстрым и плавным движениям, то обучаемый агент должен напрямую управлять моторами и с небольшой задержкой получать информацию о текущих позициях и скоростях механических сочленений. Действиями в этом случае могут быть напряжения, подаваемые на каждый мотор в каждом сочленении, а состояниями – последние данные об углах и скоростях сочленений. Вознаграждение может быть равно +1 для каждого успешно перемещенного объекта. Чтобы поощрить плавные движения, на каждом шаге можно выдавать небольшое отрицательное вознаграждение, вычисляемое как функция от неравномерности движения в соседние моменты времени. ■

Упражнение 3.1. Придумайте три примера задач, укладывающихся в формализм МППР. Определите для каждого состояния, действия и вознаграждения. Поста-

райтесь, чтобы примеры как можно сильнее различались. Формализм абстрактный и гибкий, он может применяться разными способами. Хотя бы в одном примере попытайтесь подобраться к его пределам. □

Упражнение 3.2. Достаточно ли формализма МППР для полезного представления всех задач обучения целеустремленного агента? Можете ли вы придумать очевидные исключения? □

Упражнение 3.3. Рассмотрим задачу о вождении автомобиля. Можно было бы определить действия в терминах акселератора, руля и тормозов, т. е. тех мест, где ваше тело входит в контакт с машиной. А можно было бы поместить их дальше снаружи – например, там, где шина контактирует с дорогой, считая вашими действиями крутящий момент на колесе. Или поместить их дальше внутри – скажем, там, где ваш мозг вступает в контакт с телом, и считать действиями сокращения мускулов, управляющих вашими конечностями. Или можно перейти на совсем уж высокий уровень и сказать, что ваше действие – это выбор места назначения. Какой уровень правильный, т. е. где нужно было бы провести границу между агентом и окружающей средой? На каком основании следует предпочесть одно положение границы другому? Существует ли фундаментальная причина предпочесть одно положение другому или это свободный выбор? □

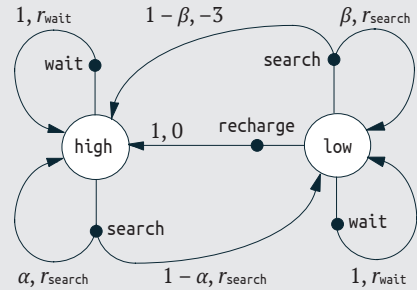
Пример 3.3. Робот-уборщик

Подвижный робот собирает пустые банки из-под газировки, оставленные в офисе. У него имеются датчики для обнаружения банок, а также рука с захватом, которая может поднимать их и складывать в находящуюся внутри корпуса урну. Робот питается от перезаряжаемого аккумулятора. В системе управления роботом имеются компоненты для интерпретации получаемой от датчиков информации, для навигации и для управления рукой и захватом. Решения верхнего уровня о том, как искать банки, принимаются обучающимся с подкреплением агентом на основе информации о текущем заряде аккумулятора. Для простоты предположим, что различается всего два уровня, составляющих небольшое множество состояний $S = \{\text{high}, \text{low}\}$. В каждом состоянии агент может принять следующие решения: (1) продолжать активный поиск банок в течение некоторого периода времени (**search**); (2) оставаться на месте и ждать, пока кто-нибудь принесет банку (**wait**); (3) направиться к станции зарядки, чтобы подзарядить аккумулятор (**recharge**). Если уровень заряда высокий (**high**), то подзарядиться бессмысленно, поэтому действие **recharge** не включено в множество действий для этого состояния. Таким образом, имеем два множества действий: $\mathcal{A}(\text{high}) = \{\text{search}, \text{wait}\}$ и $\mathcal{A}(\text{low}) = \{\text{search}, \text{wait}, \text{recharge}\}$.

Вознаграждение в большинстве случаев равно нулю, но становится положительным, когда робот находит пустую банку, и большим отрицательным, если аккумулятор полностью разряжается. Лучший способ находить банки – активно искать их, но при этом расходуется заряд аккумулятора, чего не происходит во время ожидания. Если робот ведет поиск, то существует возможность разряда аккумулятора. В таком случае робот должен остановиться и ждать, пока кто-нибудь придет на помощь (получая при этом низкое вознаграждение). Если уровень заряда **high**, то период активного поиска можно завершить без риска разрядить батарею. Период поиска, который начинается при уровне заряда **high**, оставит уровень заряда в состоянии **high** с вероятностью α и уменьшит его до уровня **low** с вероятностью $1 - \alpha$. С другой стороны, период поиска, начатый при уровне заряда **low**, оставит заряд на уровне **low** с вероятностью β и закончится полным разрядом аккумулятора с вероятностью $1 - \beta$. В последнем случае робота при-

дется выручать, после чего заряд вернется на уровень **high**. Каждая банка, подобранная роботом, засчитывается как вознаграждение 1, а если роботу пришлось выручать, то он получает вознаграждение -3 . Обозначим r_{search} и r_{wait} , где $r_{\text{search}} > r_{\text{wait}}$, ожидаемое число банок, собранных роботом во время поиска и во время ожидания соответственно. Наконец, предположим, что когда робот направляется к станции зарядки или стоит с разряженным аккумулятором, он не собирает банки. Эта система представляет собой конечный МППР, в таблице ниже выписаны вероятности переходов, ожидаемые вознаграждения и динамика:

s	a	s'	$p(s' s, a)$	$p(s, a, s')$
high	search	high	α	r_{search}
high	search	low	$1 - \alpha$	r_{search}
low	search	high	$1 - \beta$	-3
low	search	low	β	r_{search}
high	wait	high	1	r_{wait}
high	wait	low	0	r_{wait}
low	wait	high	0	r_{wait}
low	wait	low	1	r_{wait}
low	recharge	high	1	0
low	recharge	low	0	0



Отметим, что в таблице присутствуют строки для всех возможных комбинаций текущего состояния s , действия $a \in \mathcal{A}(s)$ и следующего состояния s' . Еще один полезный способ визуализации динамики конечного МППР дает граф переходов, показанный выше. В нем имеются вершины двух видов: *вершины состояний* и *вершины действий*. Для каждого возможного состояния присутствует вершина состояния (большой белый кружок, помеченный именем состояния), а для каждой пары состояние–действие – вершина действия (маленький черный кружок, помеченный именем действия и соединенный линией с вершиной состояния). Если начать в состоянии s и предпринять действие a , то мы будем двигаться вдоль линии, ведущей из вершины состояния s в вершину действия (s, a) . Тогда окружающая среда ответит переходом в следующую вершину состояния по одной из стрелок, исходящих из вершины действия (s, a) . Каждая стрелка соответствует тройке (s, s', a) , где s' – следующее состояние, и мы помечаем такую стрелку вероятностью перехода $p(s'|s, a)$ и ожидаемым вознаграждением за переход $r(s, a, s')$. Отметим, что сумма вероятностей перехода на стрелках, исходящих из любой вершины действия, всегда равна 1.

Упражнение 3.4. Постройте такую же таблицу, как в примере 3.3, но для $p(s', r|s, a)$. В ней должны быть столбцы s, a, s', r и $p(s', r|s, a)$ и по одной строке для каждой четверки такой, что $p(s', r|s, a) > 0$. □

3.2. Цели и вознаграждения

В обучении с подкреплением цель агента формализована в терминах специального сигнала, называемого вознаграждением, который окружающая среда передает агенту. На каждом временном шаге вознаграждение является числом $R_t \in \mathbb{R}$. Не-

формально говоря, цель агента заключается в том, чтобы максимизировать общее полученное вознаграждение – не получаемое немедленно, а именно суммарное при длительной работе. Эту неформальную идею можно выразить в виде *гипотезы о вознаграждении*:

Все то, что мы понимаем под целями, можно рассматривать как максимизацию математического ожидания суммарно полученного скалярного сигнала (называемого вознаграждением).

Применение сигнала вознаграждения для формализации идеи цели – одна из отличительных черт обучения с подкреплением.

На первый взгляд, постановка целей в терминах сигналов вознаграждения может показаться ограничительной, но на практике эта идея доказала гибкость и широко применяется. Убедиться в этом проще всего, рассмотрев примеры ее реального или потенциального использования. Например, чтобы научить робота ходить, исследователи предлагали на каждом временном шаге вознаграждение, пропорциональное продвижению вперед. Чтобы научить робота находить выход из лабиринта, часто предлагают вознаграждение -1 за каждый шаг до выхода; это побуждает агента находить выход как можно быстрее. Чтобы научить робота искать и собирать пустые банки, можно было бы на большинстве шагов предлагать нулевое вознаграждение, но увеличивать его до $+1$ за каждую найденную банку. Можно было бы также выдавать отрицательное вознаграждение всякий раз, как робот натывается на предметы или когда кто-нибудь кричит на него. Чтобы научить агента играть в шашки или шахматы, естественно предлагать вознаграждение $+1$ за выигрыш, -1 за проигрыш и 0 за ничью и все позиции, кроме конечной.

Можно проанализировать, что происходит во всех этих примерах. Агент всегда обучается максимизировать свое вознаграждение. Если мы хотим, чтобы он что-то сделал для нас, то должны предлагать вознаграждение, максимизируя которое, агент заодно достигает и наших целей. Поэтому так важно, чтобы вознаграждение правильно отражало, чего мы хотим добиться. В частности, сигнал вознаграждения – не место для размещения априорной информации о том, как агент может добиться того, чего мы от него хотим¹. Например, агента, играющего в шахматы, следует вознаграждать только за фактический выигрыш, а не за достижение промежуточных целей, например взятие фигур противника или захват центра доски. Если бы такие промежуточные цели вознаграждались, то агент мог бы найти способ их достижения без достижения настоящей цели. Например, он мог бы обучиться брать фигуры противника даже ценой проигрыша игры. Сигнал вознаграждения – это способ сообщить роботу, чего он должен достичь, а не как этого достичь².

¹ Для такой априорной информации больше подходит начальная стратегия, или начальная функция ценности, или факторы, оказывающие на них влияние.

² В разделе 17.4 мы глубже рассмотрим вопрос о проектировании эффективных сигналов вознаграждения.

3.3. Доход и эпизоды

До сих пор мы обсуждали назначение обучения неформально. Мы сказали, что цель агента – максимизировать полное вознаграждение, полученное за длительное время. Но как определить это формально? Если обозначить последовательность вознаграждений, полученных после шага t , $R_{t+1}, R_{t+2}, R_{t+3}, \dots$, то какой именно аспект этой последовательности мы хотим максимизировать? Вообще говоря, мы стремимся максимизировать *ожидаемый доход*, где доход, обозначаемый G_t , определен как некоторая функция от последовательности вознаграждений. В простейшем случае доход равен сумме вознаграждений:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \dots + R_T, \quad (3.7)$$

где T – последний временной шаг. Этот подход имеет смысл в приложениях, где существует естественное понятие последнего временного шага, т. е. когда взаимодействие агента с окружающей средой естественным образом распадается на подпоследовательности, называемые *эпизодами*¹, например партии в игре, обходы лабиринта или любое другое повторяющееся взаимодействие. Каждый эпизод заканчивается специальным состоянием, которое называется *заключительным* и за которым следует возврат в стандартное начальное состояние или выборка из стандартного распределения начальных состояний. Даже если эпизоды могут заканчиваться по-разному, например выигрышем или проигрышем игры, следующий эпизод начинается независимо от того, как закончился предыдущий. Поэтому можно считать, что все эпизоды завершаются в одном и том же заключительном состоянии, но с разными вознаграждениями за разные результаты. Задачи с эпизодами такого рода называются *эпизодическими*. В эпизодических задачах иногда необходимо отличать множество всех незаключительных состояний, обозначаемое $\mathcal{S}$, от множества всех состояний плюс заключительное состояние, обозначаемого $\mathcal{S}^+$. Время завершения эпизода T – случайная величина, которая в разных эпизодах, как правило, различна.

С другой стороны, во многих случаях взаимодействие агента со средой не распадается естественным образом на легко различимые эпизоды, но продолжается безгранично. Например, так было бы естественно поставить задачу управления процессом или применения робота, который делает одно и то же все время, пока существует. Такие задачи называются *непрерывными*. Определить для непрерывных задач доход, как в формуле (3.7), проблематично, потому что последним временным шагом был бы $T = \infty$, и доход, который мы пытаемся максимизировать, также мог бы оказаться бесконечным. (Например, если агент получает вознаграждение +1 на каждом временном шаге.) Поэтому в этой книге мы обычно пользуемся определением дохода, которое несколько сложнее концептуально, но гораздо проще математически.

Для этого нам понадобится еще одно понятие – *обесценивание*. При таком подходе агент пытается выбирать действия, так чтобы сумма обесцененных вознаграждений, которые он получит в будущем, принимала максимальное значение. В частности, действие A_t выбирается так, чтобы максимизировать ожидаемый обесцененный доход:

¹ В литературе эпизоды иногда называют «испытаниями».

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \quad (3.8)$$

где параметр γ , $0 \leq \gamma \leq 1$, называется *коэффициентом обесценивания*.

Коэффициент обесценивания определяет ценность будущих вознаграждений с точки зрения настоящего: вознаграждение, которое будет получено спустя k временных шагов, стоит в γ^{k-1} раз меньше, чем если бы оно было получено прямо сейчас. Если $\gamma < 1$, то бесконечная сумма в формуле (3.8) имеет конечное значение, если последовательность вознаграждений $\{R_k\}$ ограничена. Если $\gamma = 0$, то агент «близорукий», т. е. озабочен лишь максимизацией немедленного вознаграждения: в таком случае его цель – обучиться выбирать A_t , так чтобы максимизировать только R_{t+1} . Если каждое действие агента влияет лишь на немедленное вознаграждение, но не на будущие, то близорукий агент мог бы максимизировать (3.8), по отдельности максимизируя каждое немедленное вознаграждение. Но в общем случае действия, направленные на максимизацию немедленного вознаграждения, могут ограничить доступ к будущим вознаграждениям, так что доход уменьшится. Когда γ стремится к 1, вклад будущих вознаграждений в целевую функцию доходности увеличивается, т. е. агент становится более дальнзорким.

Доходы на соседних временных шагах связаны друг с другом соотношением, важным как теоретически, так и для алгоритмов обучения с подкреплением:

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \\ &\doteq R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots) \\ &\doteq R_{t+1} + \gamma G_{t+1}. \end{aligned} \quad (3.9)$$

Чтобы это равенство было справедливо для всех временных шагов $t < T$, даже если завершение имеет место в момент $t + 1$, следует положить $G_T = 0$. Часто это упрощает вычисление дохода, соответствующего последовательности вознаграждений.

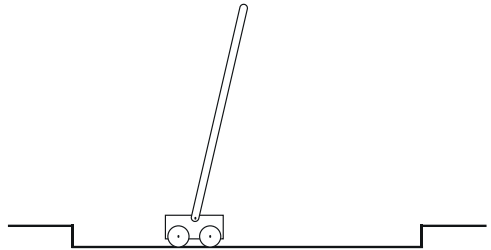
Отметим, что хотя доход (3.8) – сумма бесконечного числа слагаемых, он все же конечен, если вознаграждение ненулевое и постоянное – при условии, что $\gamma < 1$. Например, если вознаграждение всегда равно +1, то доход равен

$$G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1 - \gamma}. \quad (3.10)$$

Упражнение 3.5. Равенства в разделе 3.1 справедливы для непрерывного случая, а для эпизодических задач их необходимо модифицировать (совсем чуть-чуть). Покажите, что вы знаете, как именно, для чего приведите модифицированный вариант формулы (3.3). □

Пример 3.4: балансирование стержня.

В этой задаче требуется прикладывать силу к тележке, катящейся по дорожке, так чтобы шарнирно прикрепленный к ней стержень не падал. Неудачей считается ситуация, когда стержень отклонился более чем на заданный угол от вертикали или тележка докатилась до конца



дорожки. После каждой неудачи стержень снова устанавливается вертикально. Эту задачу можно было бы рассматривать как эпизодическую, в которой попытки уравновесить стержень – естественные эпизоды. Вознаграждение может быть равно $+1$ за каждый временной шаг, на котором не случилась неудача, тогда доход равен количеству шагов до наступления неудачи. В таком случае вечное успешное балансирование принесло бы бесконечный доход. Можно вместо этого рассматривать балансирование стержня как непрерывную задачу, применив обесценивание. Тогда вознаграждение могло бы быть равно -1 за каждую неудачу и 0 в остальных случаях. Доход в каждый момент времени был бы связан с $-\gamma^K$, где K – количество временных шагов до наступления неудачи. В любом случае для максимизации дохода следует удерживать шест в равновесии как можно дольше. ■

Упражнение 3.6. Предположим, что балансирование стержня рассматривается как эпизодическая задача, но с использованием обесценивания, так что все вознаграждения равны 0 , кроме вознаграждения за неудачу, равного -1 . Каким тогда будет доход в каждый момент времени? Чем этот доход отличается от дохода в непрерывной задаче с обесцениванием? □

Упражнение 3.7. Допустим, вы проектируете робота для прохождения лабиринта. Вы решили давать вознаграждение $+1$ за выход из лабиринта и 0 в остальных случаях. Похоже, задача естественно распадается на эпизоды – последовательные прохождения лабиринта, поэтому вы решили рассматривать ее как эпизодическую, в которой цель – максимизировать ожидаемое полное вознаграждение (3.7). Погоняв некоторое время обучающегося агента, вы обнаружили полное отсутствие прогресса. Что не так? Достаточно ли доходчиво вы сообщили агенту, чего именно он должен достичь? □

Упражнение 3.8. Предположим, что $\gamma = 0.5$, последовательность полученных вознаграждений имеет вид: $R_1 = -1, R_2 = 2, R_3 = 6, R_4 = 3, R_5 = 2$ и $T = 5$. Чему равны $G_0, G_1, \dots, G_5$? Указание: идите от конца к началу. □

Упражнение 3.9. Предположим, что $\gamma = 0.9$ и последовательность вознаграждений начинается с $R_1 = 2$, за которым следует бесконечная последовательность чисел 7 . Чему равны G_0 и G_1 ? □

Упражнение 3.10. Докажите равенство (3.10). □

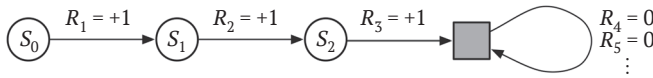
3.4. Унифицированная нотация для эпизодических и непрерывных задач

В предыдущем разделе мы описали два вида задач обучения с подкреплением: в одном взаимодействие между агентом и окружающей средой естественно разбивается на последовательность отдельных эпизодов (эпизодические задачи), а в другом – нет (непрерывные задачи). Первый случай проще с математической точки зрения, потому что каждое действие влияет только на конечное число вознаграждений, которые агент последовательно получает в течение эпизода. В этой

книге мы иногда рассматриваем один вид задач, иногда – другой, но чаще оба. Поэтому полезно выработать единую нотацию, которая позволит рассуждать об обоих случаях одновременно.

Для точности рассуждений об эпизодических задачах необходимы дополнительные обозначения. Вместо одной длинной последовательности временных шагов мы должны рассматривать ряд эпизодов, каждый из которых состоит из конечной последовательности шагов. Мы будем нумеровать шаги каждого эпизода, начиная заново с нуля. Поэтому нужно рассматривать не просто S_t – представление состояния в момент t , – а $S_{t,i}$, представление состояния в момент t эпизода i (и аналогично для $A_{t,i}$, $R_{t,i}$, $\pi_{t,i}$, T_i и т. д.). Оказывается, однако, что при обсуждении эпизодических задач нам почти никогда не приходится различать отдельные эпизоды. Почти всегда мы рассматриваем один конкретный эпизод или высказываем утверждение, справедливое для всех эпизодов. Соответственно, на практике мы почти всегда будем, хотя это и не вполне правильно, упрощать нотацию, опуская явное упоминание номера эпизода. То есть будем писать S_t вместо $S_{t,i}$ и т. д.

Нам нужно еще одно соглашение, чтобы получить единую нотацию для эпизодических и непрерывных задач. Мы определили доход как сумму конечного числа членов в случае (3.7) и бесконечного числа членов в случае (3.8). Эти два случая можно объединить, если рассматривать завершение эпизода как вход в специальное поглощающее состояние, из которого есть переход только в себя же и в котором всегда генерируется вознаграждение 0. Например, рассмотрим такую диаграмму переходов состояний:



Здесь серый квадратик представляет специальное поглощающее состояние, соответствующее концу эпизода. Начав в состоянии S_0 , мы получаем последовательность вознаграждений $+1, +1, +1, 0, 0, 0, \dots$. Ее суммирование дает такой же доход, как если бы суммировали только первые T вознаграждений (здесь $T = 3$). И это справедливо, даже если ввести обесценивание. Поэтому мы можем определить доход в общем случае по формуле (3.8), применяя соглашение об опускании номеров эпизодов, если они не нужны, и допустив возможность $\gamma = 1$, если сумма при этом остается корректно определенной (например, потому что все эпизоды завершаются). Альтернативно можно написать

$$G_t \doteq \sum_{k=t+1}^T \gamma^{k-t-1} R_k, \quad (3.11)$$

допустив возможность, что $T = \infty$ или $\gamma = 1$ (но не то и другое вместе). Мы будем придерживаться этих соглашений во всей книге, чтобы упростить обозначения и подчеркнуть параллели между эпизодическими и непрерывными задачами. (Позже, в главе 10, мы рассмотрим постановку задачи, являющейся одновременно непрерывной и необесцененной.)

3.5. СТРАТЕГИИ И ФУНКЦИИ ЦЕННОСТИ

Почти все алгоритмы обучения с подкреплением включают *функции ценности* – функции состояний (или пар состояние–действие), которые оценивают, насколько хорошо для агента оказаться в данном состоянии (или насколько хорошо предпринять данное действие в данном состоянии). Здесь понятие «хорошо» определено в терминах будущих ожидаемых вознаграждений или, точнее говоря, будущего дохода. Разумеется, вознаграждения, которых агент может ожидать в будущем, зависят от предпринимаемых им действий. Поэтому и функции ценности определены по отношению к конкретным способам действий – стратегиям.

Формально *стратегией* называется отображение состояний на вероятности выбора каждого возможного действия. Если агент следует стратегии π в момент t , то $\pi(a|s)$ – вероятность того, что $A_t = a$, если $S_t = s$. Как и p , π – обыкновенная функция; знак $|$ в середине выражения $\pi(a|s)$ просто напоминает, что она определяет распределение вероятностей $a \in \mathcal{A}(s)$ для каждого $s \in \mathcal{S}$. Методы обучения с подкреплением описывают, как изменяется стратегия агента в результате приобретения им опыта.

Упражнение 3.11. Если текущее состояние равно S_t и действия выбираются согласно стохастической стратегии π , то каково математическое ожидание R_{t+1} в терминах π и функции p с четырьмя аргументами (3.2)? $\square$

Функция ценности состояния s при стратегии π , обозначаемая $v_\pi(s)$, – это ожидаемый доход, когда агент начинает работу в состоянии s и в дальнейшем следует стратегии π . Для МППР мы можем формально определить v_* следующим образом:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right] \text{ для всех } s \in \mathcal{S}, \quad (3.12)$$

где $\mathbb{E}_\pi[\cdot]$ – математическое ожидание случайной величины, при условии что агент следует стратегии π , а t – произвольный временной шаг. Отметим, что ценность заключительного состояния, если оно существует, всегда равна 0. Мы называем функцию v_* *функцией ценности состояний при стратегии π* .

Аналогично ценность выполнения действия a в состоянии s при стратегии π , обозначаемая $q_\pi(s, a)$, определяется как ожидаемый доход, когда агент начинает работу в состоянии s , предпринимает действие a и затем следует стратегии π :

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a \right]. \quad (3.13)$$

Мы называем q_π *функцией ценности действий при стратегии π* .

Упражнение 3.12. Выразите v_π через q_π и π . $\square$

Упражнение 3.13. Выразите q_π через v_π и функцию p с четырьмя аргументами. $\square$

Функции ценности v_π и q_π можно оценить из опыта. Например, если агент следует стратегии π и хранит для каждого встретившегося состояния среднее фактических доходов, последовавших за этим состоянием, то это среднее будет схо-

даться к ценности состояния $v_\pi(s)$, когда количество появлений этого состояния стремится к бесконечности. Если для каждого действия, предпринятого в этом состоянии, хранятся отдельные средние, то эти средние точно так же будут сходиться к ценностям действий, $q_\pi(s, a)$. Мы называем такие методы оценивания *методами Монте-Карло*, потому что они сводятся к усреднению по большой случайной выборке фактических доходов. Методы этого вида представлены в главе 5. Конечно, если состояний очень много, то практически может оказаться невозможно хранить средние для каждого состояния отдельно. Вместо этого агент должен был бы представить v_π и q_π в виде параметрических функций (с количеством параметров меньшим, чем количество состояний) и подбирать параметры, так чтобы они лучше соответствовали наблюдаемым доходам. При этом также могут получиться верные оценки, хотя многое зависит от природы аппроксимирующей параметрической функции. Эти возможности обсуждаются во второй части книги.

Фундаментальное свойство функций ценности, используемых в обучении с подкреплением и динамическом программировании, состоит в том, что они удовлетворяют рекуррентным соотношениям, подобным тому, которое мы уже установили для дохода (3.9). Для любой стратегии π и любого состояния s имеет место следующее условие согласованности между ценностью s и ценностями возможных следующих за ним состояний:

$$\begin{aligned}
 v_\pi(s) &\doteq \mathbb{E}_\pi[G_t | S_t = s] \\
 &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] && \text{(из (3.9))} \\
 &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']] \\
 &= \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) [r + \gamma v_\pi(s')] \quad \text{для всех } s \in \mathcal{S}, && (3.14)
 \end{aligned}$$

где неявно подразумевается, что действия a берутся из множества $\mathcal{A}(s)$, что следующие состояния s' берутся из множества $\mathcal{S}$ (или из $\mathcal{S}^+$ в случае эпизодической задачи) и что вознаграждения r берутся из множества $\mathcal{R}$. Отметим также, что в последнем равенстве мы объединили две суммы, одна – по всем значениям s' , другая – по всем значениям r , в одну сумму по всем комбинациям того и другого. Мы часто используем такого рода двойные суммы, чтобы упростить формулы. Обратите внимание, что окончательное выражение легко интерпретируется как математическое ожидание. Действительно, это сумма по всем значениям трех величин: a , s' и r . Для каждой тройки мы вычисляем ее вероятность, $\pi(a|s)p(s', r|s, a)$, которая назначается в качестве веса выражению в квадратных скобках, а затем суммируем по всем вероятностям для получения значения математического ожидания.

Равенство (3.14) называется *уравнением Беллмана для v_π* . Оно выражает соотношение между ценностью состояния и ценностями следующих за ним состояний. Представьте себе, что смотрите вперед из некоторого состояния на его возможных последователей, как показано на рисунке. Белые кружочки представляют состояния, а черные – пару состояние–действие. Начав из состояния s , корня дерева на рисунке, агент может выбрать любое из некоторого множества действий – три из них показаны на рисунке, – сообразуясь со

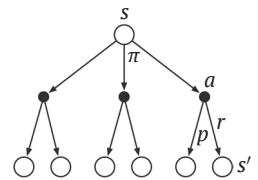


Диаграмма
предшествующих
состояний для v_π

своей стратегией π . В ответ на каждое действие окружающая среда может предложить несколько следующих состояний (на рисунке показано два) и вознаграждение r , зависящее от ее динамики, описываемой функцией p . Уравнение Беллмана (3.14) усредняет все эти возможности, назначая каждой вес, равный ее вероятности. Оно утверждает, что ценность начального состояния должна быть равна (обесцененной) ценности следующего ожидаемого состояния плюс полученное попутно вознаграждение.

Функция ценности v_π является единственным решением уравнения Беллмана. В последующих главах мы покажем, что уравнение Беллмана лежит в основе многих способов вычисления, аппроксимации и обучения v_π . Диаграммы, подобные показанной выше, мы называем *диаграммами предшествующих состояний* (back-up diagram), они иллюстрируют связи, лежащие в основе рекуррентных операций обновления, составляющих самую суть методов обучения с подкреплением. Эти операции переносят информацию о ценности последующих состояний (или пар состояние–действие) *назад* в текущее состояние (или пару состояние–действие). Мы часто будем использовать диаграммы предшествующих состояний в этой книге, чтобы наглядно проиллюстрировать обсуждаемые алгоритмы. (Отметим, что, в отличие от графов переходов, вершины состояний в диаграммах предшествующих состояний необязательно представляют разные состояния; например, состояние может быть приемником самого себя.)

Пример 3.5: сеточный мир. На рис. 3.2 (слева) показано представление простого конечного МППР на прямоугольной сетке. Ячейки сетки соответствуют состояниям окружающей среды. В каждой ячейке возможно четыре действия: север, юг, восток и запад, которые детерминированно переводят агента в соседнюю ячейку в соответствующем направлении. Действия, которые могли бы вывести агента за пределы сетки, оставляют его на месте, но при этом сопровождаются вознаграждением -1 . Все прочие действия приносят вознаграждение 0 , за исключением тех, которые выводят агента из специальных состояний A и B. Все четыре действия, выводящих из состояния A, приносят вознаграждение $+10$ и переводят агента в состояние A'. Все действия, выводящие из состояния B, приносят вознаграждение $+5$ и переводят агента в состояние B'.



Рис. 3.2. ❖ Пример с сеткой: динамика вознаграждения в исключительных состояниях (слева) и функция ценности состояний для равновероятностной случайной стратегии (справа)

Предположим, что агент выбирает любое из четырех действий в любом состоянии с равной вероятностью. На рис. 3.2 (справа) показана функция ценности v_π для этой стратегии в случае вознаграждения с коэффициентом обесценивания

$\gamma = 0.9$. Эта функция ценности была вычислена путем решения системы линейных уравнений (3.14). Обратите внимание на отрицательные значения вдоль нижнего края; это результат того, что в этой области при случайной стратегии высока вероятность наткнуться на край сетки. При данной стратегии лучше всего оказаться в состоянии А, но ожидаемый доход в нем меньше 10, немедленного вознаграждения, потому что из него агент попадает в состояние А', откуда с большой вероятностью упрется в край сетки. С другой стороны, ценность состояния В больше 5, немедленного вознаграждения в нем, потому что оттуда агент попадает в состояние В', ценность которого положительна. В В' ожидаемый штраф (отрицательное вознаграждение) за возможное упирание в край с лихвой компенсируется ожидаемым выигрышем от возможного попадания в состояние А или В. ■

Упражнение 3.14. Уравнение Беллмана (3.14) должно выполняться в каждом состоянии для функции ценности v_π , показанной на рис. 3.2 (справа) в примере 3.5. Покажите численно, что это уравнение выполняется для центрального состояния с ценностью $+0.7$ по отношению к четырем его соседним состояниям с ценностями $+2.3$, $+0.4$, -0.4 и $+0.7$. (Точность этих чисел – всего один знак после запятой.) □

Упражнение 3.15. В примере сеточного мира вознаграждения положительны для целей, отрицательны для выхода на края мира и равны нулю в остальных случаях. Что важно: знаки вознаграждений или только расстояния между ними? Докажите, пользуясь формулой (3.8), что прибавление константы c ко всем вознаграждениям прибавляет константу v_c к ценностям всех состояний и потому не влияет на относительные ценности состояний при любой стратегии. Выразите v_c через c и γ . □

Упражнение 3.16. Теперь рассмотрим прибавление константы c ко всем вознаграждениям в эпизодической задаче, например о прохождении лабиринта. Будет ли это иметь какой-то эффект или задача останется такой же, как в случае непрерывной задачи выше? Почему? Приведите пример. □

Пример 3.6: игра в гольф. Чтобы описать прохождение лунки на поле для гольфа как задачу обучения с подкреплением, мы можем начислять штраф (отрицательное вознаграждение) -1 за каждый удар, пока мяч не упадет в лунку. Состоянием будет положение мяча. Ценность состояния – количество ударов до попадания в лунку из этого положения с обратным знаком. Наши действия – это, конечно, прицеливание и удар по мячу, а также выбор клюшки. Будем считать первые два действия константами и рассмотрим только выбор клюшки. Кроме того, будем считать, что клюшек всего две: паттер и драйвер. На рис. 3.3 сверху показана возможная функция ценности состояний $v_{\text{putt}}(s)$ для стратегии, в которой всегда выбирается паттер. Ценность заключительного состояния в лунке равна 0. Предполагается, что патт (катящийся удар) можно выполнить из любой точки грина (засаженной травой части поля), эти состояния имеют ценность -1 . Находясь вне грина, мы не можем загнать мяч в лунку паттом, поэтому ценность выше. Если мы можем из некоторого состояния забросить мяч на грин паттом, то ценность этого состояния должна быть на 1 меньше ценности грина, т. е. равна -2 . Для простоты предположим, что патт можно выполнить очень точно и детерминированно, но с небольшого расстояния. Это дает нам четкий контур – линию уровня, – помеченный на рисунке числом -2 ; из всех точек между этой линией и грином для попада-

ния в лунку нужно ровно два удара. Аналогично любая точка, отстоящая от линии уровня -2 на расстояние не больше дальности патта, должна иметь ценность -3 и т. д. для всех линий уровня, показанных на рисунке. Патт не выводит за пределы песочниц, поэтому их ценность равна $-\infty$. Итак, чтобы загнать мяч в лунку с подставки только паттами, потребуется шесть ударов.

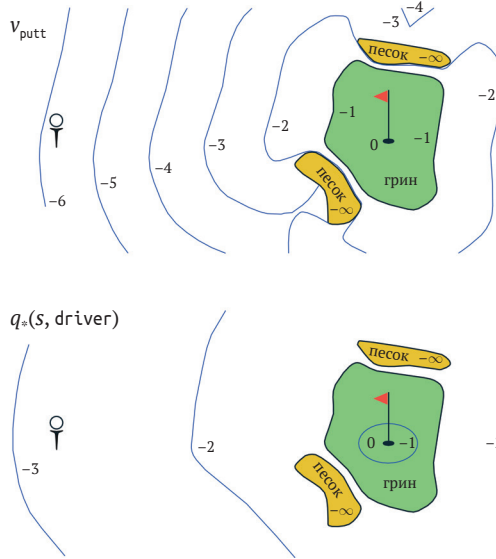
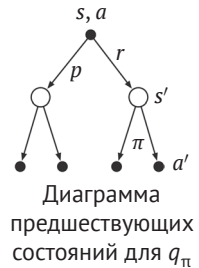
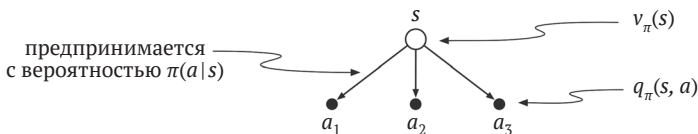


Рис. 3.3 ❖ Пример игры в гольф: функция ценности состояний для выбора паттера (сверху) и оптимальная функция ценности действий для выбора драйвера (снизу)

Упражнение 3.17. Как выглядит уравнение Беллмана для ценностей действий, т. е. для q_π ? Оно должно выражать ценность действия $q_\pi(s, a)$ в терминах ценностей действий $q_\pi(s', a')$ возможных преемников пары состояние–действие (s, a) . Указание: этому уравнению соответствует диаграмма предшествующих состояний справа. Выпишите последовательность равенств, аналогичную (3.14), но для ценностей действий.

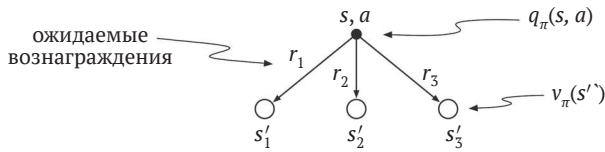


Упражнение 3.18. Ценность состояния зависит от ценностей действий, возможных в этом состоянии, и от вероятностей выбора действий при текущей стратегии. Мы можем представить это в виде небольшой диаграммы предшествующих состояний с корнем в состоянии, на которой показаны все возможные действия:



Выпишите уравнение, соответствующее этому интуитивному представлению и диаграмме для ценности в корневом узле $v_\pi(s)$, в терминах ценности в ожидаемом листовом узле $q_\pi(s, a)$, при условии что $S_t = s$. Это уравнение должно включать математическое ожидание, обусловленное следованием стратегии π . Затем выпишите второе уравнение, в котором ожидаемая ценность выражена явно через $\pi(a|s)$, так чтобы никакого математического ожидания ценности в уравнение не входило. $\square$

Упражнение 3.19. Ценность действия $q_\pi(s, a)$ зависит от следующего ожидаемого вознаграждения и математического ожидания суммы оставшихся вознаграждений. И снова мы можем рассуждать о задаче, опираясь на небольшую диаграмму предшествующих состояний, на этот раз с корнем в действии (паре состояние–действие), которое ведет в одно из следующих возможных состояний:



Выпишите уравнение, соответствующее этому интуитивному представлению и диаграмме для ценности действия $q_\pi(s, a)$, в терминах следующего ожидаемого вознаграждения R_{t+1} и ожидаемой ценности следующего состояния $v_\pi(S_{t+1})$, при условии что $S_t = s$ and $A_t = a$. Это уравнение должно включать математическое ожидание, но не обусловленное следованием стратегии. Затем выпишите второе уравнение, в котором ожидаемая ценность выражена явно через величину $p(s', r|s, a)$, определенную в (3.2), так чтобы никакого математического ожидания ценности в уравнение не входило. $\square$

3.6. ОПТИМАЛЬНЫЕ СТРАТЕГИИ И ОПТИМАЛЬНЫЕ ФУНКЦИИ ЦЕННОСТИ

Решение задачи обучения с подкреплением означает, грубо говоря, нахождение стратегии, которая дает большое вознаграждение за длительный период времени. Для конечных МППР можно точно определить оптимальную стратегию следующим образом. Функции ценности определяют отношение частичного порядка на множестве стратегий. Будем говорить, что стратегия π больше (лучше) или равна стратегии π' , если ожидаемый доход для π больше или равен ожидаемому доходу для π' для любого состояния. Иными словами, $\pi \geq \pi'$ тогда и только тогда, когда $v_\pi(s) \geq v_{\pi'}(s)$ для всех $s \in \mathcal{S}$. Всегда существует хотя бы одна стратегия, большая или равная всем остальным стратегиям. Это *оптимальная стратегия*. Хотя оптимальных стратегий может быть несколько, мы обозначаем любую из них π_* . У них у всех одна и та же функция ценности состояний, которая называется *оптимальной функцией ценности состояний*, обозначается v и определяется следующим образом:

$$v_*(s) \doteq \max_{\pi} v_\pi(s) \quad (3.15)$$

для всех $s \in \mathcal{S}$.

У оптимальных стратегий также одна и та же функция ценности действий, которая обозначается q_* и определяется следующим образом:

$$q_*(s, a) \doteq \max_{\pi} q_{\pi}(s, a) \quad (3.16)$$

для всех $s \in \mathcal{S}$ и $a \in \mathcal{A}(s)$. Для пары состояние–действие (s, a) эта функция дает ожидаемый доход от выбора действия a в состоянии s и далее следования оптимальной стратегии. Таким образом, мы можем следующим образом выразить q_* через v_* :

$$q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a]. \quad (3.17)$$

Пример 3.7: оптимальные функции ценности для игры в гольф. В нижней части рис. 3.3 показаны линии уровня возможной оптимальной функции ценности действий $q_*(s, \text{driver})$. Это ценности всех состояний в случае, когда мы наносим первый удар клюшкой-драйвером, а затем выбираем драйвер или паттер в зависимости от того, что лучше. Драйвер позволяет отправить мяч дальше, но с меньшей точностью. Загнать мяч в лунку одним ударом драйвером можно, только если мы уже очень близко, поэтому линия уровня -1 для $q_*(s, \text{driver})$ охватывает лишь малую часть грина. Но если в нашем распоряжении два удара, то мы можем достать до лунки с гораздо большего расстояния, что и показывает линия уровня -2 . В этом случае мы не обязаны попасть драйвером в точку внутри линии уровня -1 , нужно лишь добить до грина, а там уже можно будет использовать паттер. Оптимальная функция ценности действий дает ценности после фиксирования конкретного первого действия, в данном случае удара драйвером, после чего уже можно выбирать действия, которые кажутся лучше. Линия уровня -3 расположена еще дальше от лунки и включает подставку, с которой выполняется первый удар. Лучшая последовательность действий, начинающаяся с подставки, состоит из двух ударов драйвером и одного паттером, в результате чего мяч загоняется в лунку тремя ударами. ■

Поскольку v_* – функция ценности для стратегии, то она должна удовлетворять условию согласованности, которое описывается уравнением Беллмана для ценностей состояний (3.14). Но поскольку это оптимальная функция ценности, то условие согласованности можно записать в специальном виде, не ссылаясь ни на какую конкретную стратегию. Это уравнение Беллмана для v_* , или *уравнение оптимальности Беллмана*. Интуитивно уравнение оптимальности Беллмана выражает тот факт, что ценность состояния при оптимальной стратегии должна быть равна ожидаемому доходу для лучшего действия в данном состоянии:

$$\begin{aligned} v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s, a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t | S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \end{aligned} \quad (\text{в силу (3.9)})$$

$$= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \quad (3.18)$$

$$= \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')]. \quad (3.19)$$

Последние два равенства – это две формы уравнения оптимальности Беллмана для v_* . Уравнение оптимальности Беллмана для q_* имеет вид:

$$\begin{aligned} q_*(s, a) &= \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a] \\ &= \sum_{s', r} p(s', r | s, a) [r + \gamma \max_{a'} q_*(s', a')]. \end{aligned} \quad (3.20)$$

Диаграммы предшествующих состояний на рисунке ниже графически показывают области будущих состояний и действий, рассматриваемые в уравнениях оптимальности Беллмана для v_* и q_* . Это те же диаграммы предшествующих состояний для v_* и q_* , что были представлены ранее, с тем отличием, что в точках, где агент делает выбор, добавлены дуги, показывающие, что берется максимум по вариантам выбора, а не ожидаемая ценность при некоторой стратегии. Диаграмма предшествующих состояний слева графически представляет уравнение оптимальности Беллмана (3.19), а диаграмма предшествующих состояний справа – уравнение (3.20).

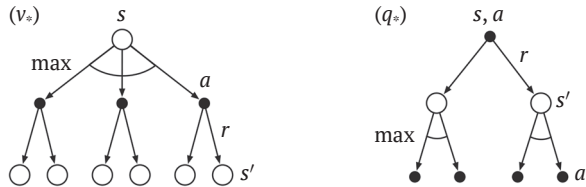


Рис. 3.4 ❖ Диаграммы предшествующих состояний для v_* и q_*

Для конечных МППР уравнение оптимальности Беллмана (3.19) имеет единственное решение v_* , не зависящее от стратегии. В действительности уравнение оптимальности Беллмана – это система уравнений, по одному для каждого состояния, так что при наличии n состояний мы имеем систему n уравнений с n неизвестными. Если динамика p окружающей среды известна, то в принципе эту систему уравнений можно решить относительно v_* , применяя любой из многочисленных методов решения систем нелинейных уравнений. Можно решить соответствующую систему уравнений для q_* .

Имея v_* , относительно легко определить оптимальную стратегию. Для каждого состояния s будет существовать одно или несколько действий, в которых достигается максимум в уравнении оптимальности Беллмана. Любая стратегия, которая назначает ненулевую вероятность только этим действиям, будет оптимальной. Мы можем считать это одношаговым поиском. Если известна оптимальная функция ценности v_* , то действия, которые выглядят лучшими после такого одношагового поиска, и будут оптимальными действиями. По-другому то же самое можно выразить, сказав, что любая стратегия, жадная относительно оптимальной функции ценности v_* , является оптимальной. Термин «жадный» употребляется в информатике для описания любой процедуры поиска или принятия решений, которая производит выбор, основываясь только на локальных, или промежуточных, сообщениях, не принимая во внимание, что такой выбор может воспрепятствовать отысканию лучших вариантов в будущем. Следовательно, он описывает стратегии,

которые выбирают действия, учитывая только краткосрочные последствия. Полезное свойство v_* состоит в том, что если использовать ее для вычисления краткосрочных последствий действий, а точнее одношаговых последствий, то жадная стратегия будет оптимальна и в интересующей нас долгосрочной перспективе, поскольку v_* уже учитывает вознаграждения, получаемые в результате всех будущих вариантов поведения. Благодаря v_* оптимальный ожидаемый долгосрочный доход превращается в локальную величину, которая непосредственно доступна для каждого состояния. Поэтому поиск с заглядыванием вперед на один шаг дает оптимальные действия в долгосрочной перспективе.

Зная q_* , выбрать оптимальные действия еще проще. В этом случае агенту даже не нужно заглядывать на один шаг вперед: для любого состояния ему достаточно просто найти какое-нибудь действие, доставляющее максимум функции $q_*(s, a)$. Функция ценности действий по существу кеширует результаты всех поисков с заглядыванием на один шаг вперед. Она возвращает оптимальный ожидаемый долгосрочный доход в качестве локального значения, которое непосредственно доступно для каждой пары состояние–действие. Поэтому ценой зависимости от пар состояние–действие, а не просто от состояний, оптимальная функция ценности действий позволяет выбирать оптимальные действия, ничего не зная о возможных последующих состояниях и их ценности, т. е. не нуждаясь в информации о динамике окружающей среды.

Пример 3.8: решение задачи о сеточном мире. Предположим, что мы решаем уравнение Беллмана относительно v_* для простой задачи на сетке из примера 3.5, которая еще раз показана на рис. 3.5 (слева). Напомним, что за выходом из состояния A следует вознаграждение +10 и переход в состояние A', а за выходом из состояния B – вознаграждение +5 и переход в состояние B'. На рис. 3.5 (в середине) показана оптимальная функция ценности, а на рис. 3.5 (справа) – соответствующие оптимальные стратегии. Если в ячейке показано несколько стрелок, значит, все соответствующие им действия оптимальны.

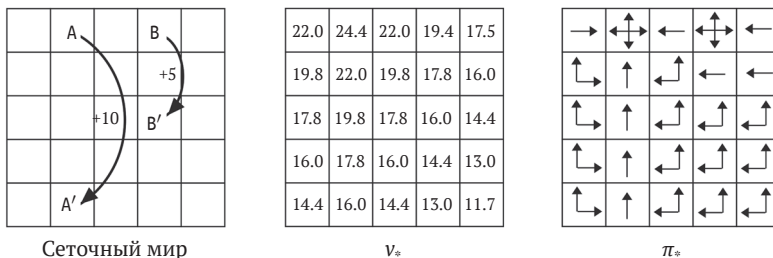


Рис. 3.5 ❖ Оптимальные решения задачи о сеточном мире

Пример 3.9: уравнения оптимальности Беллмана для робота-уборщика. Воспользовавшись формулой (3.19), мы можем явно выписать уравнение оптимальности Беллмана для примера с роботом-уборщиком. Чтобы немного сократить формулы, заменим состояния high и low, а также действия search, wait и recharge буквами h, l, s, w и re соответственно. Поскольку состояний всего два, уравнение оптимальности Беллмана сводится к системе двух уравнений. Уравнение для $v_*(h)$ записывается в виде:

$$\begin{aligned}
 v_*(h) &= \max \left\{ \begin{aligned} &p(h|h, s)[r(h, s, h) + \gamma v_*(h)] + p(l|h, s)[r(h, s, l) + \gamma v_*(l)], \\ &p(h|h, w)[r(h, w, h) + \gamma v_*(h)] + p(l|h, w)[r(h, w, l) + \gamma v_*(l)] \end{aligned} \right\} \\
 &= \max \left\{ \begin{aligned} &\alpha[r_s + \gamma v_*(h)] + (1 - \alpha)[r_s + \gamma v_*(l)], \\ &1[r_w + \gamma v_*(h)] + 0[r_w + \gamma v_*(l)] \end{aligned} \right\} \\
 &= \max \left\{ \begin{aligned} &r_s + \gamma[\alpha v_*(h) + (1 - \alpha)v_*(l)], \\ &r_w + \gamma v_*(h) \end{aligned} \right\}.
 \end{aligned}$$

Та же процедура для $v_*(l)$ дает:

$$v_*(l) = \max \left\{ \begin{aligned} &\beta r_s - 3(1 - \beta) + \gamma[(1 - \beta)v_*(h) + \beta v_*(l)], \\ &r_w + \gamma v_*(l), \\ &\gamma v_*(h) \end{aligned} \right\}.$$

При любом выборе r_s , r_w , α , β и γ таких, что $0 \leq \gamma < 1$, $0 \leq \alpha, \beta \leq 1$, существует ровно одна пара чисел $v_*(h)$ и $v_*(l)$, одновременно удовлетворяющих этим двум нелинейным уравнениям. ■

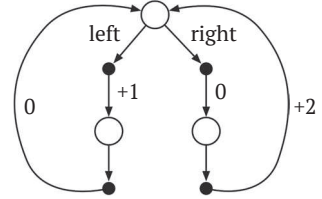
Явное решение уравнения оптимальности Беллмана – один из способов нахождения оптимальной политики, а значит, и решения задачи обучения с подкреплением. Но этот способ редко бывает полезен. Он сродни исчерпывающему поиску – необходимо просмотреть все будущие возможности, вычислить их вероятности и желательность с точки зрения ожидаемого вознаграждения. Это решение опирается по крайней мере на три предположения, которые редко выполняются на практике: (1) нам точно известна динамика окружающей среды; (2) имеется достаточно вычислительных ресурсов для полного вычисления; (3) имеет место марковское свойство. Для тех задач, которые нам интересны, реализовать такое решение в полном объеме, вообще говоря, невозможно, поскольку те или иные предположения нарушаются. Например, хотя для игры в нарды с первым и третьим предположениями проблем не возникает, второе все портит. Поскольку в игре порядка 10^{20} состояний, самым быстрым современным компьютерам понадобилось бы работать тысячи лет, чтобы решить уравнение Беллмана для v_* , и то же самое относится к нахождению q_* . В обучении с подкреплением обычно приходится соглашаться на приближенные решения.

Есть много различных методов принятия решений, которые можно рассматривать как способы приближенного решения уравнения оптимальности Беллмана. Например, эвристические методы поиска можно интерпретировать как раскрытие правой части (3.19) несколько раз, на какую-то глубину, в результате чего образуется «дерево возможностей», к которому затем применяется эвристическая функция оценивания для аппроксимации v_* в «листовых» узлах. (Эвристические методы поиска типа A^* почти всегда относятся к эпизодическому случаю.) Методы динамического программирования еще теснее связаны с уравнением оптимальности Беллмана. Многие методы обучения с подкреплением можно интерпретировать как приближенное решение этого уравнения с использованием фактически обученных переходов вместо знания ожидаемых переходов. В следующих главах мы рассмотрим несколько таких методов.

Упражнение 3.20. Нарисуйте или опишите оптимальную функцию ценности состояний для примера игры в гольф. □

Упражнение 3.21. Нарисуйте или опишите линии уровня оптимальной функции ценности действий для игры паттером, $q_*(s, \text{putter})$, в примере игры в гольф. □

Упражнение 3.22. Рассмотрим непрерывный МППР, показанный на рисунке справа. Единственное решение принимается в верхнем состоянии, в котором возможно два действия, left и right. Числа обозначают детерминированные вознаграждения за каждое действие. Имеется ровно две детерминированные стратегии, π_{left} и π_{right} . Какая стратегия оптимальна, если $\gamma = 0$? А если $\gamma = 0.9$? А если $\gamma = 0.5$? □



Упражнение 3.23. Выпишите уравнение Беллмана для q_* в задаче о роботе-уборщике. □

Упражнение 3.24. Рисунок 3.5 показывает, что оптимальная ценность лучшего состояния сеточного мира равна 24.4 с точностью до одного знака после запятой. Воспользуйтесь своим знанием оптимальной стратегии и формулой (3.8), чтобы выразить эту ценность символически, а затем вычислите ее с тремя знаками после запятой. □

Упражнение 3.25. Запишите уравнение для v_* в терминах q_* . □

Упражнение 3.26. Запишите уравнение для q_* в терминах v_* и функции p с четырьмя аргументами. □

Упражнение 3.27. Запишите уравнение для π_* в терминах q_* . □

Упражнение 3.28. Запишите уравнение для π_* в терминах v_* и функции p с четырьмя аргументами. □

Упражнение 3.29. Перепишите все четыре уравнения Беллмана для четырех функций ценности (v_π , v_* , q_π и q_*) в терминах функции p с тремя аргументами (3.4) и функции r с двумя аргументами (3.5). □

3.7. Оптимальность и аппроксимация

Мы определили оптимальные функции ценности и оптимальные стратегии. Ясно, что агент, обучившийся оптимальной стратегии, отлично справился с задачей, но на практике такое бывает редко. В задачах, которые нам интересны, оптимальные стратегии можно сгенерировать только ценой очень большого объема вычислений. Корректно определенное понятие оптимальности определяет подход к обучению, описываемый в этой книге, и дает ключ к пониманию теоретических свойств различных алгоритмов обучения, но это идеал, к которому агенты могут лишь в какой-то мере приблизиться. Как уже было сказано, даже если у нас имеется полная и точная модель динамики окружающей среды, обычно невозможно вычислить оптимальную политику путем решения уравнения оптимальности Беллмана. Например, игры на доске, скажем шахматы, составляют лишь крохот-

ную долю человеческого опыта, но все равно большие, специально спроектированные компьютеры не могут вычислить оптимальные ходы. Для решения задачи, стоящей перед агентом, критически важен объем доступных вычислительных ресурсов и, в частности, объем вычислений на одном временном шаге.

Доступная память – также важное ограничение. Большой объем памяти часто необходим для аппроксимации функций ценности, стратегий и моделей. В задачах с небольшим конечным множеством состояний такие аппроксимации можно построить в виде массивов или таблиц, в которых имеется одна запись для каждого состояния (или пары состояние–действие). Мы называем это *табличным* случаем, а соответствующие методы решения – табличными методами. Но во многих случаях, представляющих практический интерес, состояний так много, что никакой таблицы не хватит. Тогда функции необходимо аппроксимировать с помощью какого-то более компактного представления в виде параметрической функции.

Наша постановка задачи обучения с подкреплением вынуждает соглашаться на аппроксимацию. Но она также открывает уникальные возможности для нахождения полезных аппроксимаций. Например, при аппроксимации оптимального поведения может существовать много состояний с такой низкой вероятностью, что выбор неоптимальных действий в них почти не оказывает влияния на размер полученного агентом вознаграждения. Например, программа игры в нарды Тезауро демонстрирует замечательную игру, хотя может принимать очень плохие решения в позициях, которые никогда не возникают в игре с умелым противником. На самом деле вполне может статься, что программа TD-Gammon принимает плохие решения для очень большой доли множества состояний игры. Благодаря онлайн-природе обучения с подкреплением можно аппроксимировать оптимальные стратегии, так чтобы больше усилий тратилось на обучение умению принимать хорошие решения в часто встречающихся состояниях ценой меньшего внимания к редко встречающимся. Это то ключевое свойство, которое отличает обучение с подкреплением от других подходов к приближенному решению МППР.

3.8. РЕЗЮМЕ

Подведем итог тем элементам задачи обучения с подкреплением, о которых мы узнали в этой главе. Под обучением с подкреплением понимается обучение в процессе взаимодействия с целью научиться правильно вести себя для достижения цели. Обучающийся с подкреплением *агент* и *окружающая среда* взаимодействуют на протяжении последовательности дискретных временных шагов. Описание их интерфейса определяет конкретную задачу: *действия* – это варианты, выбираемые агентом, *состояния* – основа для принятия решений о выборе, а *вознаграждения* – основа для оценивания выбора. Все, что находится внутри агента, полностью известно и контролируется агентом, а все, что вне, абсолютно не контролируется, но может быть или не быть полностью известно. *Стратегия* – это стохастическое правило, руководствуясь которым, агент выбирает действия в зависимости от состояния. Цель агента – максимизировать сумму вознаграждения, полученного на протяжении периода времени.

Если описанная выше задача обучения с подкреплением сформулирована с помощью корректно определенных вероятностей переходов, то она образует *марковский процесс принятия решений* (МППР). *Конечный МППР* – это МППР с конечными множествами состояний, действий и вознаграждений (в том виде, в каком мы определили их здесь). Современная теория обучения с подкреплением по большей части ограничена конечными МППР, но методы и идеи имеют более широкое применение.

Доход – это функция будущих вознаграждений, которую агент стремится максимизировать (точнее, ее математическое ожидание). У него имеется несколько определений, зависящих от природы задачи и того, хотим ли мы *обесценивать* отложенное вознаграждение. Формулировка без обесценивания годится для *эпизодических задач*, когда взаимодействие между агентом и средой естественно распадается на *эпизоды*, а формулировка с обесцениванием – для *непрерывных задач*, когда взаимодействие не распадается на эпизоды, а продолжается безгранично. Мы хотим определить доход в задачах обоих видов, так чтобы к эпизодическому и непрерывному случаям был применим один и тот же набор уравнений.

Функции ценности стратегии сопоставляют каждому состоянию или паре состояние–действие ожидаемый доход, который агент может получить, применяя данную стратегию. *Оптимальные функции ценности* сопоставляют каждому состоянию или паре состояние–действие наибольший ожидаемый доход, достижимый при использовании любой стратегии. Стратегия, для которой функции ценности оптимальны, называется *оптимальной стратегией*. Оптимальные функции ценности для состояний и пар состояние–действие однозначно определены для данного МППР, но оптимальных стратегий может быть много. Любая стратегия, *жадная* относительно оптимальных функций ценности, должна быть оптимальной стратегией. *Уравнения оптимальности Беллмана* – это специальные условия согласованности, которым должны удовлетворять оптимальные функции ценности и которые в принципе можно решить и тем самым найти оптимальные функции ценности, после чего можно сравнительно просто вычислить оптимальную стратегию.

Задачу обучения с подкреплением можно поставить разными способами в зависимости от предположений об уровне знаний, доступных агенту в начальный момент. В задачах с *полным знанием* агент располагает полной и точной моделью динамики окружающей среды. Если окружающая среда представляет собой МППР, то такая модель состоит из функции динамики p с четырьмя аргументами (3.2). В задачах с *неполным знанием* полная и точная модель среды отсутствует.

Даже если у агента имеется полная и точная модель окружающей среды, он, как правило, все равно не может произвести на одном шаге достаточный объем вычислений, чтобы воспользоваться ей в полной мере. Другим важным ограничением является доступная память. Память необходима для построения хороших аппроксимаций функций ценности, стратегий и моделей. В большинстве практически интересных случаев состояний гораздо больше, чем можно представить с помощью записей таблицы, поэтому необходима аппроксимация. Корректно определенное понятие оптимальности определяет подход к обучению, описываемый в этой книге, и дает ключ к пониманию теоретических свойств различных алгоритмов обучения, но это идеал, к которому агенты могут лишь в какой-то мере приблизиться. В обучении с подкреплением нас очень интересуют случаи,

в которых оптимальное решение найти невозможно, но его можно каким-то способом аппроксимировать.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

Задача обучения с подкреплением многим обязана марковским процессам принятия решений (МППР), применяемым в оптимальном управлении. Это, а также влияние психологии описано в краткой истории в главе 1. Обучение с подкреплением вносит свой вклад в МППР в виде повышенного внимания к аппроксимации и неполной информации для больших реалистичных задач. МППР и задача обучения с подкреплением лишь слабо связаны с традиционными задачами обучения и принятия решений в искусственном интеллекте. Однако теперь специалисты по искусственному интеллекту живо интересуются МППР-постановками задач планирования и принятия решений с самых разных точек зрения. МППР являются более общими, чем прежние формулировки, использовавшиеся в искусственном интеллекте, в том плане, что допускают более общие виды целей и неопределенность.

Теория МППР рассматривается, например, в работах Bertsekas (2005), White (1969), Whittle (1982, 1983) и Puterman (1994). Особенно компактное изложение конечного случая имеется в работе Ross (1983). МППР изучались также в контексте стохастического оптимального управления, где методы *адаптивного* оптимального управления наиболее тесно связаны с обучением с подкреплением (см., например, Kumar, 1985; Kumar and Varaiya, 1986).

Теория МППР развивалась из-за стремления разобраться в задаче принятия последовательности решений в условиях неопределенности, когда каждое решение может зависеть от предыдущих и их результатов. Иногда МППР называют многошаговыми процессами принятия решений или последовательными процессами принятия решений, эти названия уходят корнями в статистическую теорию последовательной выборки, начало которой положили статьи Thompson (1933, 1934) и Robbins (1952), которые мы цитировали в главе 2 в связи с задачами о бандитах (это прототипы МППР, сформулированные как задачи с несколькими ситуациями).

Самый ранний известный нам пример обсуждения обучения с подкреплением с применением формализма МППР – данное в работе Andreea (1969b) описание унифицированного взгляда на самообучающиеся машины. В работе Witten and Corbin (1973) описаны эксперименты с системой обучения с подкреплением, позднее проанализированные в работе Witten (1977, 1976a) с использованием формализма МППР. В работе Werbos (1977) МППР явно не упоминаются, но предложены методы приближенного решения задач стохастического оптимального управления, связанные с современными методами обучения с подкреплением (см. также Werbos, 1982, 1987, 1988, 1989, 1992). Хотя идеи Вербоса не нашли тогда широкого признания, они оказались провидческими, поскольку придавали большую важность приближенному решению задач оптимального управления в различных областях, включая искусственный интеллект. Наиболее влиятельной работой по интеграции обучения с подкреплением и МППР стала статья Watkins (1989).

3.1 Наша характеристика динамики МППР в терминах функции $p(s', r | s, a)$ не вполне обычна. В литературе по МППР динамика чаще описывается в тер-

минах вероятностей перехода состояний $p(s'|s, a)$ и ожидаемых следующих вознаграждений $r(s, a)$. Однако в обучении с подкреплением нам чаще приходится обращаться к индивидуальным фактическим или выборочным вознаграждениям (а не просто к их ожидаемым значениям). Наша нотация также делает понятнее тот факт, что S_t и R_t в общем случае определяются совместно и потому должны иметь общий временной индекс. Преподавая обучение с подкреплением, мы пришли к выводу, что наша нотация концептуально проще и легче для понимания.

Хорошее обсуждение теоретико-системной концепции состояния на интуитивном уровне см. в работе Minsky (1967).

Пример биореактора основан на работах Ungar (1990) и Miller and Williams (1992). Пример робота-уборщика навеян роботом для сбора банок, описанным в работе Jonathan Connell (1989). В работе Kober and Peters (2012) приведена подборка приложений обучения с подкреплением к робототехнике.

3.2 Гипотеза о вознаграждении была сформулирована Майклом Литтманом (частное сообщение).

3.3–4 Терминология *эпизодических* и *непрерывных* задач отличается от той, что обычно употребляется в литературе по МППР, где принято различать три типа задач: (1) задачи с конечным горизонтом, в которых взаимодействие прекращается после *фиксированного* количества временных шагов; (2) задачи с неопределенным горизонтом, в которых взаимодействие может продолжаться сколь угодно долго, но в конечном итоге должно прекратиться; (3) задачи с бесконечным горизонтом, в которых взаимодействие не прекращается никогда. Наши эпизодические и непрерывные задачи аналогичны задачам с неопределенным и бесконечным горизонтом соответственно, но мы предпочитаем подчеркнуть различие в природе взаимодействия. Это различие кажется более фундаментальным, чем разница в целевых функциях, подразумеваемая традиционной терминологией. Часто в эпизодических задачах используется целевая функция с неопределенным горизонтом, а в непрерывных – целевая функция с бесконечным горизонтом, но мы считаем, что это скорее случайное совпадение, чем фундаментальное различие.

Пример балансирования стержня взят из работ Michie and Chambers (1968) и Barto, Sutton, and Anderson (1983).

3.5–6 Назначение ценности в зависимости от того, хорошим является действие или плохим в долгосрочной перспективе, имеет давнюю историю. Отображение состояний на числовые значения, представляющие долгосрочные последствия управляющих решений, – ключевая часть теории оптимального управления, которая была разработана в 1950-х годах как обобщение построенных в XIX веке теорий классических механизмов, основанных на функциях состояния (см., например, Schultz and Melsa, 1967). При описании того, как можно запрограммировать компьютер для игры в шахматы, Шеннон (Shannon, 1950) предложил использовать функцию оценивания, которая учитывала бы долгосрочные достоинства и недостатки позиций.

Алгоритм Q-обучения Уоткинса (Watkins, 1989) для оценивания q_* (глава 6) сделал функции ценности действий важной частью обучения с подкреплением, поэтому эти функции часто называют Q-функциями. Но сама идея функции ценности действий гораздо старше. В работе Shannon (1950) было высказано предположение, что функцию $h(P, M)$ можно использовать в программе игры в шахматы, чтобы решить, имеет ли смысл исследовать ход M в позиции P . Систему MENACE (Michies, 1961, 1963) и систему BOXES (Michie and Chambers, 1968) можно интерпретировать как оценку функций ценности действий. В классической физике функция Гамильтона является функцией ценности действий; ньютонова динамика является жадной относительно этой функции (см. Goldstein, 1957). Функции ценности действий сыграли также главную роль в теоретическом рассмотрении динамического программирования (Denardo, 1967) в терминах сжимающих отображений.

Уравнение оптимальности Беллмана (для v_*) получило известность после работы Ричарда Беллмана (Bellman, 1957a), который называл его «базовым функциональным уравнением». Аналогом уравнения оптимальности Беллмана для задач с непрерывным временем и состоянием является уравнение Гамильтона–Якоби–Беллмана (часто его называют просто уравнением Гамильтона–Якоби), уходящее корнями в классическую физику (см. Schultz and Melsa, 1967).

Пример игры в гольф предложил Крис Уоткинс.

Глава 4

Динамическое программирование

Под динамическим программированием (ДП) понимают семейство алгоритмов, которые используются для вычисления оптимальных стратегий, при условии что имеется идеальная модель окружающей среды в виде марковского процесса принятия решений (МППР). От классических алгоритмов ДП в обучении с подкреплением мало пользы, как из-за предположения об идеальной модели, так и из-за очень больших вычислительных затрат, но все равно они важны с теоретической точки зрения. ДП закладывает прочный фундамент для понимания методов, представленных в этой книге. На самом деле все эти методы можно рассматривать как попытки достичь того же эффекта, что ДП, только при меньшем объеме вычислений и без предположения о наличии идеальной модели среды.

Начиная с этой главы мы обычно будем предполагать, что окружающая среда – это конечный МППР, т. е. множества состояний, действий и вознаграждений, $\mathcal{S}$, $\mathcal{A}$ и $\mathcal{R}$, конечны, а динамика задается множеством вероятностей $p(s', r | s, a)$ для всех $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$, $r \in \mathcal{R}$ и $s' \in \mathcal{S}^+$ ($\mathcal{S}^+$ – это $\mathcal{S}$ плюс заключительное состояние, если задача эпизодическая). Хотя идеи ДП применимы и к задачам с непрерывными пространствами состояний и действий, точное решение возможно только в частных случаях. Типичный способ получить приближенное решение для задач с непрерывными состояниями и действиями заключается в том, чтобы дискретизировать пространства состояний и действий, а затем применить методы ДП для конечного числа состояний. Методы, изучаемые в главе 9, применимы к непрерывным задачам и являются важным обобщением данного подхода.

Ключевая идея ДП и обучения с подкреплением вообще заключается в использовании функций ценности для организации и структурирования поиска хороших стратегий. В этой главе мы покажем, как можно использовать ДП для вычисления функций ценности, определенных в главе 3. Там было сказано, что можно без труда получить оптимальные стратегии, коль скоро уже известны оптимальные функции ценности v_* или q_* , удовлетворяющие уравнениям оптимальности Беллмана:

$$\begin{aligned} v_*(s) &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \\ &= \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')] \end{aligned} \quad (4.1)$$

или

$$\begin{aligned} q_*(s, a) &= \mathbb{E}[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a] \\ &= \sum_{s', r} p(s', r | s, a) [r + \gamma \max_{a'} q_*(s', a')] \end{aligned} \quad (4.2)$$

для всех $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$ и $s' \in \mathcal{S}^+$. Как мы увидим, алгоритмы ДП получаются путем преобразования подобных уравнений Беллмана в присваивания, т. е. в правила обновления для улучшения приближений к искомым функциям ценности.

4.1. ОЦЕНИВАНИЕ СТРАТЕГИИ (ПРЕДСКАЗАНИЕ)

Сначала рассмотрим, как вычислить функцию ценности состояния v_π для произвольной стратегии π . В литературе по ДП это называется *оцениванием стратегии*. Мы также будем использовать термин *задача предсказания*. Напомним (см. главу 3), что для всех $s \in \mathcal{S}$

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t | S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \end{aligned} \quad (\text{из (3.9)})$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \quad (4.3)$$

$$= \sum_a \pi(a | s) \sum_{s', r} p(s', r | s, a) [r + \gamma v_\pi(s')], \quad (4.4)$$

где $\pi(a | s)$ – вероятность предпринять действие a в состоянии s при стратегии π , а индекс π после знака математического ожидания означает, что оно обусловлено следованием стратегии π . Существование и единственность v_π гарантированы при условии, что либо $\gamma < 1$, либо стратегия π гарантирует завершение при старте из любого состояния.

Если динамика окружающей среды полностью известна, то (4.4) – система $|\mathcal{S}|$ линейных уравнений с $|\mathcal{S}|$ неизвестными ($v_\pi(s)$, $s \in \mathcal{S}$). В принципе, для ее решения требуется прямолинейное, хотя и длительное, вычисление. Для наших целей больше подходят итеративные методы. Рассмотрим последовательность приближенных функций ценности $v_0, v_1, v_2, \dots$, каждая из которых отображает $\mathcal{S}^+$ в $\mathbb{R}$ (множество вещественных чисел). Начальное приближение v_0 выбирается произвольно (но заключительному состоянию, если оно существует, должна быть сопоставлена ценность 0), а каждое следующее приближение получается применением уравнения Беллмана для v_* (4.4) в качестве правила обновления:

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s] \\ &= \sum_a \pi(a | s) \sum_{s', r} p(s', r | s, a) [r + \gamma v_k(s')] \end{aligned} \quad (4.5)$$

для всех $s \in \mathcal{S}$. Очевидно, что $v_k = v_\pi$ – неподвижная точка этого правила обновления, поскольку уравнение Беллмана для v_* гарантирует нам равенство в этом случае. В действительности можно показать, что последовательность $\{v_k\}$ в общем случае сходится к v_π , когда $k \rightarrow \infty$, при тех же условиях, которые гарантируют существование v_π . Этот алгоритм называется *итеративным оцениванием стратегии*.

Для вычисления следующего приближения v_{k+1} по v_k алгоритм итеративного оценивания стратегии применяет одну и ту же операцию к каждому состоянию s : заменяет старую ценность s новой ценностью, вычисленной по старым ценностям следующих за s состояний и ожидаемым немедленным вознаграждениям на всех одношаговых переходах при следовании оцениваемой стратегии. Мы называем такого рода операцию *полным обновлением*. На каждой итерации итеративного оценивания стратегии ценность каждого состояния обновляется один раз с целью получить новую приближенную функцию ценности v_{k+1} . Существуют разные виды полного обновления, зависящие от того, что обновляется – состояние (как здесь) или пара состояние–действие, – а также от точного способа комбинирования оценок ценностей последующих состояний. Все обновления, производимые в алгоритмах ДП, называются *полными*, потому что они основаны на математическом ожидании, взятом по *всем* возможным следующим состояниям, а не на выборочном следующем состоянии. Природу обновления можно выразить в виде уравнения, как выше, или с помощью диаграммы предшествующих состояний, с которыми мы познакомились в главе 3. Например, диаграмма предшествующих состояний, соответствующая ожидаемому обновлению в алгоритме итеративного оценивания стратегии, приведена на стр. 87.

Чтобы написать последовательную компьютерную программу, реализующую итеративное оценивание стратегии (4.5), понадобилось бы два массива: для старых ценностей $v_k(s)$ и для новых ценностей $v_{k+1}(s)$. Имея такие массивы, новые ценности можно вычислять по старым, не изменяя значения старых ценностей. Разумеется, проще было бы использовать один массив и обновлять ценности «на месте», т. е. сразу перезаписывать старое значение новым. Но тогда в зависимости от порядка обновления состояний иногда в правой части (4.5) использовались бы новые ценности вместо старых. Этот алгоритм с обновлением на месте тоже сходится к v_π ; на самом деле обычно он сходится даже быстрее, чем вариант с двумя массивами, поскольку новые данные используются сразу, как только становятся доступны. Можно рассматривать обновления как проход по пространству состояний. Для алгоритма с обновлением на месте порядок обновления ценностей состояний во время этого прохода существенно влияет на скорость сходимости. Обычно, говоря об алгоритмах ДП, мы имеем в виду обновление на месте.

Во врезке ниже приведена полная версия алгоритма итеративного оценивания стратегий с обновлением на месте, написанная на псевдокоде. Обратите внимание, как обрабатывается завершение. Строго говоря, этот алгоритм сходится только в пределе, но на практике его можно остановить раньше. Псевдокод проверяет величину $\max_{s \in \mathcal{S}} |v_{k+1}(s) - v_k(s)|$ после каждого прохода и останавливается, если она достаточно мала.

Алгоритм итеративного оценивания стратегии для оценивания $V \approx v_\pi$

Вход: π , стратегия, подлежащая оцениванию

Параметр алгоритма: небольшая пороговая величина $\theta > 0$, определяющая точность оценки

Инициализировать $V(s)$ для всех $s \in \mathcal{S}^+$ произвольным образом с той оговоркой, что $V(\text{terminal}) = 0$

Повторять:

$\Delta \leftarrow 0$

Повторять для каждого $s \in \mathcal{S}$:

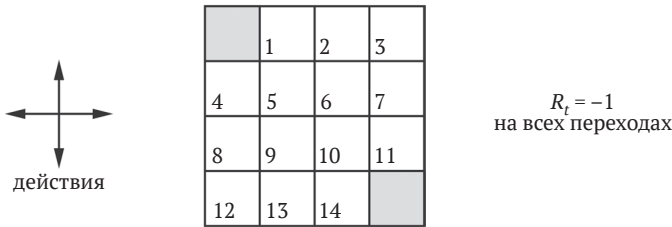
$v \leftarrow V(s)$

$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

пока не окажется $\Delta < \theta$

Пример 4.1. Рассмотрим сетку 4×4 , показанную ниже.



Незаключительными являются состояния $\mathcal{S} = \{1, 2, \dots, 14\}$. В каждом состоянии возможно четыре действия $\mathcal{A} = \{\text{up}, \text{down}, \text{right}, \text{left}\}$, которые детерминированно вызывают соответствующий переход состояний. Исключение составляют действия, которые вывели бы агента за пределы сетки, в этом случае состояние не изменяется. Так, например, $p(6, -1|5, \text{right}) = 1$, $p(7, -1|7, \text{right}) = 1$ и $p(10, r|5, \text{right}) = 0$ для всех $r \in \mathcal{R}$. Это эпизодическая задача без обесценивания. Вознаграждение при любом переходе равно -1 , пока не достигнуто заключительное состояние. Заключительное состояние закрашено серым цветом (оно показано в двух местах, но формально это одно состояние). Таким образом, функция ожидаемого вознаграждения имеет вид $r(s, a, s') = -1$ для всех состояний s, s' и действий a . Предположим, что агент следует равновероятной случайной стратегии (все действия выбираются с одинаковой вероятностью). В левой части рис. 4.1 показана последовательность функций ценности $\{v_k\}$, вычисленных алгоритмом итеративного оценивания стратегии. Конечная оценка и есть v_π , которая в данном случае дает для каждого состояния ожидаемое количество шагов из него до заключительного состояния с обратным знаком. ■

Упражнение 4.1. В примере 4.1, если π – равновероятная случайная стратегия, чему равно $q_\pi(11, \text{down})$? А чему равно $q_\pi(7, \text{down})$? □

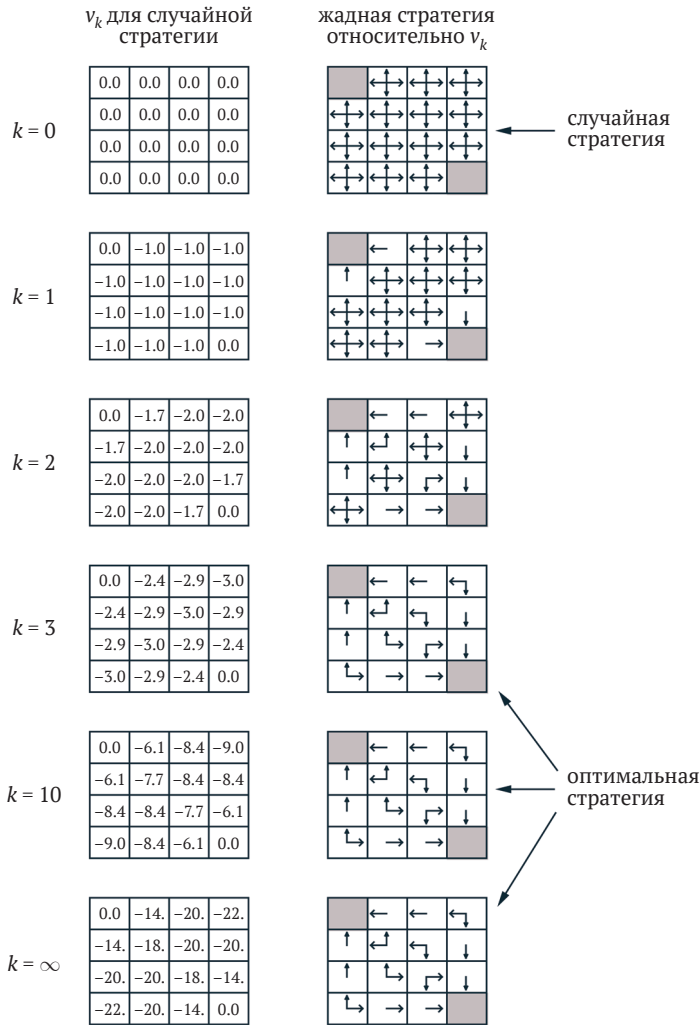


Рис. 4.1 ❖ Сходимость итеративного оценивания стратегии в маленьком сеточном мире. В левом столбце показана последовательность приближений к функции ценности состояний для случайной стратегии (все действия равновероятны). В правом столбце показана последовательность жадных стратегий, соответствующих оценкам функции ценности (стрелками показаны все действия, приводящие к достижению максимума, а числа округлены до двух значащих цифр). Гарантируется, что последняя стратегия всего лишь лучше случайной, но в данном случае она, а также все стратегии после третьей итерации оптимальны

Упражнение 4.2. В примере 4.1 предположим, что добавлено новое состояние 15 под состоянием 13, а действия left, up, right, down в нем переводят агента в состояния 12, 13, 14 и 15 соответственно. Переходы из старых состояний не изменились. Чему тогда равно $v_{\pi}(15)$ при следовании равновероятной случайной стратегии? Теперь предположим, что динамика состояния 13 тоже изменилась, так что дей-

ствие down переводит агента в новое состояние 15. Чему в этом случае равно $v_\pi(15)$ при следовании равновероятной случайной стратегии? $\square$

Упражнение 4.3. Как выглядят уравнения, аналогичные (4.3), (4.4) и (4.5), для функции ценности действий q_π и ее последовательных приближений функциями $q_0, q_1, q_2, \dots$? $\square$

4.2. УЛУЧШЕНИЕ СТРАТЕГИИ

Причина, по которой мы вычисляем функцию ценности для стратегии, – помочь в нахождении лучших стратегий. Предположим, что мы определили функцию ценности v_π для произвольной детерминированной стратегии π . Для некоторого состояния s мы хотели бы знать, стоит или не стоит изменять стратегию, так чтобы она детерминированно выбирала действие $a \neq \pi(s)$. Мы знаем ценность следования текущей стратегии из s , то есть $v_\pi(s)$, но будет ли лучше или хуже перейти на новую стратегию? Один из способов ответить на этот вопрос – рассмотреть, что будет, если выбрать a в состоянии s , а затем следовать существующей стратегии π . Ценность при таком поведении равна

$$\begin{aligned} q_\pi(s, a) &\doteq \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a] \\ &= \sum_{s', r} p(s', r | s, a) [r + \gamma v_\pi(s')]. \end{aligned} \quad (4.6)$$

Основной критерий – больше она $v_\pi(s)$ или меньше. Если больше – т. е. лучше, находясь в s , выбирать a , а затем следовать π , чем следовать π все время, – то можно было бы ожидать, что еще лучше будет выбирать a всякий раз, как встретится s , и что новая стратегия действительно будет лучше старой в целом.

Так оно и есть, и это частный случай общего результата, называемого *теоремой об улучшении стратегии*. Пусть π и π' – любая пара детерминированных стратегий такая, что для всех $s \in \mathcal{S}$

$$q_\pi(s, \pi'(s)) \geq v_\pi(s). \quad (4.7)$$

Тогда стратегия π' должна быть не хуже, чем π . Иначе говоря, она должна приносить не меньший ожидаемый доход для всех состояний $s \in \mathcal{S}$:

$$v_{\pi'}(s) \geq v_\pi(s). \quad (4.8)$$

Более того, если неравенство (4.7) строгое в любом состоянии, то неравенство (4.8) должно быть строгим по крайней мере в одном состоянии. Этот результат, в частности, применим к двум стратегиям, рассмотренным в предыдущем абзаце, исходной детерминированной стратегии π и измененной стратегии π' , которая совпадает с π всюду, за исключением того, что $\pi'(s) = a \neq \pi(s)$. Очевидно, что (4.7) имеет место во всех состояниях, кроме s . Таким образом, если $q_\pi(s, a) > v_\pi(s)$, то измененная стратегия действительно лучше, чем π .

Идея доказательства теоремы об улучшении стратегии проста. Начиная с (4.7), мы продолжаем раскрывать q_π , применяя (4.6), а затем снова (4.7), пока не дойдем до $v_{\pi'}(s)$:

$$\begin{aligned}
v_\pi(s) &\leq q_\pi(s, \pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = \pi'(s)] && (\text{из (4.6)}) \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
&\leq \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}; \pi'(S_{t+1})) | S_t = s] && (\text{из (4.7)}) \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma \mathbb{E}_\pi[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] | S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) | S_t = s] \\
&\leq \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) | S_t = s] \\
&\vdots \\
&\leq \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots | S_t = s] \\
&= v_{\pi'}(s).
\end{aligned}$$

До сих пор мы видели, как, зная стратегию и ее функцию ценности, мы легко можем оценить изменение стратегии, состоящее в замене одного действия в одном состоянии. Напрашивается обобщение – рассмотреть изменение всех возможных действий во всех состояниях, выбирая в каждом состоянии то действие, которое кажется наилучшим согласно функции $q_\pi(s, a)$. Иначе говоря, рассмотреть новую жадную стратегию π' , определенную следующим образом:

$$\begin{aligned}
\pi'(s) &\doteq \operatorname{argmax}_a q_\pi(s, a) \\
&= \operatorname{argmax}_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a] && (4.9) \\
&= \operatorname{argmax}_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_\pi(s')],
\end{aligned}$$

где argmax_a обозначает значение a , при котором следующее далее выражение достигает максимума (возможные неоднозначности разрешаются произвольным образом). Жадная стратегия выбирает действие, которое кажется наилучшим в краткосрочной перспективе – после заглядывания вперед на один шаг – согласно функции v_π . По построению, жадная стратегия удовлетворяет условиям теоремы об улучшении стратегии (4.7), поэтому мы знаем, что она не хуже исходной стратегии. Процесс конструирования новой стратегии, улучшающей исходную путем жадного выбора относительно функции ценности исходной стратегии, называется *улучшением стратегии*.

Предположим, что новая жадная стратегия π' столь же хороша, но не лучше старой стратегии π . Тогда $v_\pi = v_{\pi'}$, и из (4.9) следует, что для всех $s \in \mathcal{S}$

$$\begin{aligned}
v_{\pi'}(s) &\doteq \max_a \mathbb{E}[R_{t+1} + \gamma v_{\pi'}(S_{t+1}) | S_t = s, A_t = a] \\
&= \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_{\pi'}(s')].
\end{aligned}$$

Но это то же самое, что уравнение оптимальности Беллмана (4.1), и потому $v_{\pi'}$ должна совпадать с v_* , а обе стратегии π и π' должны быть оптимальными. Таким образом, улучшение стратегии обязано давать строго лучшую стратегию всегда, кроме случая, когда исходная стратегия уже оптимальна.

До сих пор в этом разделе мы рассматривали частный случай детерминированных стратегий. В общем случае стохастическая стратегия π задает вероятности

$\pi(a|s)$ выбора каждого действия a в каждом состоянии s . Мы не будем вдаваться в детали, скажем лишь, что все идеи этого раздела легко обобщаются на стохастические стратегии. В частности, теорема об улучшении стратегии переносится в неизменном виде на стохастический случай. Кроме того, если на шагах улучшения стратегии типа (4.9) возникают неоднозначности – т. е. имеется несколько действий, для которых достигается максимум, – то в стохастическом случае мы не обязаны выбрать из них какое-то одно действие. Вместо этого каждому максимизирующему действию может быть выделена доля вероятности быть выбранным в новой жадной стратегии. Допустима любая схема распределения долей, лишь бы все субмаксимальные действия получили нулевую вероятность.

В последней строке на рис. 4.1 приведен пример улучшения стратегии для стохастических стратегий. Здесь исходная стратегия π является случайной и равновероятной, а новая стратегия π' – жадной относительно v_π . Функция ценности v_π показана на диаграмме в левой нижней части, а множество возможных π' – на диаграмме в правой нижней части. Состояния с несколькими стрелками на диаграмме π' – это те состояния, в которых имеется несколько действий, доставляющих максимум в (4.9); разрешено любое распределение долей вероятности между этими действиями. Можно видеть, что функция ценности любой такой стратегии $v_{\pi'}(s)$ равна -1 , -2 или -3 во всех состояниях $s \in \mathcal{S}$, тогда как $v_\pi(s)$ никогда не превосходит -14 . Таким образом, $v_{\pi'}(s) \geq v_\pi(s)$ для всех $s \in \mathcal{S}$, чем иллюстрируется улучшение стратегии. В этом случае новая стратегия π' оказалась оптимальной, но, вообще говоря, гарантируется лишь улучшение.

4.3. ИТЕРАЦИЯ ПО СТРАТЕГИЯМ

После того как стратегия π была улучшена с помощью функции v_π до лучшей стратегии π' , мы можем вычислить функцию $v_{\pi'}$ и снова улучшить стратегию до π'' . Таким путем мы получаем последовательность монотонно улучшающихся стратегий и функций ценности:

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} v_*,$$

где $\xrightarrow{E}$ обозначает *оценивание* стратегии, а $\xrightarrow{I}$ – *улучшение* стратегии. Гарантируется, что каждая следующая стратегия строго лучше предыдущей (если только предыдущая уже не является оптимальной). Поскольку для конечного МППР существует лишь конечное число стратегий, этот процесс должен сойтись к оптимальной стратегии и оптимальной функции ценности за конечное число итераций.

Данный способ нахождения оптимальной стратегии называется *итерацией по стратегиям*. Полный алгоритм приведен во врезке ниже. Заметим, что каждое оценивание стратегии, само являющееся итеративным вычислением, начинается с функции ценности предыдущей стратегии. Как правило, это приводит к существенному увеличению скорости сходимости оценивания (предположительно потому, что функция ценности слабо изменяется при переходе от одной стратегии к следующей).

Алгоритм итерации по стратегиям (с использованием итеративного оценивания стратегии) для оценивания $\pi \approx \pi_*$.

1. Инициализация
 $V(s) \in \mathbb{R}$ и $\pi(s) \in \mathcal{A}(s)$ выбираются произвольно для всех $s \in \mathcal{S}$
2. Оценивание стратегии
 Повторять:
 $\Delta \leftarrow 0$
 Повторять для каждого $s \in \mathcal{S}$:
 $v \leftarrow V(s)$
 $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s)) [r + \gamma V(s')]$
 $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
 пока не окажется $\Delta < \theta$ (небольшое положительное число, определяющее точность оценки)
3. Улучшение стратегии
 $policy-stable \leftarrow true$
 Для каждого $s \in \mathcal{S}$:
 $old-action \leftarrow \pi(s)$
 $\pi(s) \leftarrow \arg \max_a \sum_{s',r} p(s', r|s, a) [r + \gamma V(s')]$
 Если $old-action \neq \pi(s)$, то $policy-stable \leftarrow false$
 Если $policy-stable$, то остановиться и вернуть $V \approx v_*$ и $\pi \approx \pi'$; иначе перейти к 2

Пример 4.2. Аренда автомобилей. Джек управляет двумя офисами национальной компании по аренде автомобилей. Каждый день в каждый офис приходят клиенты, чтобы арендовать автомобиль. Если у Джека есть свободная машина, то он сдает ее в аренду и получает от компании 10 долларов. Если машин не окажется, то он упустит выгоду. Машины становятся доступны для аренды на следующий день после возврата. Чтобы всегда иметь свободные машины, Джек может ночью перегонять их из одного офиса в другой, каждый перегон обходится в 2 доллара. Будем предполагать, что количество запрошенных и возвращенных машин в каждом офисе – пуассоновские случайные величины, т. е. вероятность, что число равно n , составляет $(\lambda^n/n!) e^{-\lambda}$, где λ – математическое ожидание. Предположим, что для количества запросов в первом и втором офисах λ равно 3 и 4 соответственно, а для количества возвратов – 3 и 2. Чтобы немного упростить задачу, предположим, что в каждом офисе может быть не более 20 машин (все лишние машины возвращаются компании и в задаче не фигурируют), а за ночь из одного офиса в другой можно перегнать не более пяти машин. Примем, что коэффициент обесценивания $\gamma = 0.9$, и будем рассматривать эту задачу как непрерывный конечный МППР, где временной шаг равен одному дню, состоянием является количество машин в каждом офисе в конце дня, а действием – количество машин, перегнанных из одного офиса в другой за ночь. На рис. 4.2 показана последовательность стратегий, найденная алгоритмом итерации по стратегиям, который начался со стратегии, при которой машины вообще не перегоняются.

Для сходимости алгоритма требуется на удивление мало итераций, что иллюстрирует пример Джека, а также пример на рис. 4.1. На левой нижней диаграмме на рис. 4.1 показана функция ценности для равновероятной случайной стратегии, а на правой нижней – жадная стратегия для этой функции ценности. Теорема об улучшении стратегии говорит, что эти стратегии лучше исходной случайной стратегии. Однако в данном случае они не просто лучше, а оптимальны, т. е. доходят до заключительных состояний за минимальное количество шагов. В этом примере алгоритм итерации по стратегиям нашел бы оптимальную стратегию после всего одной итерации.

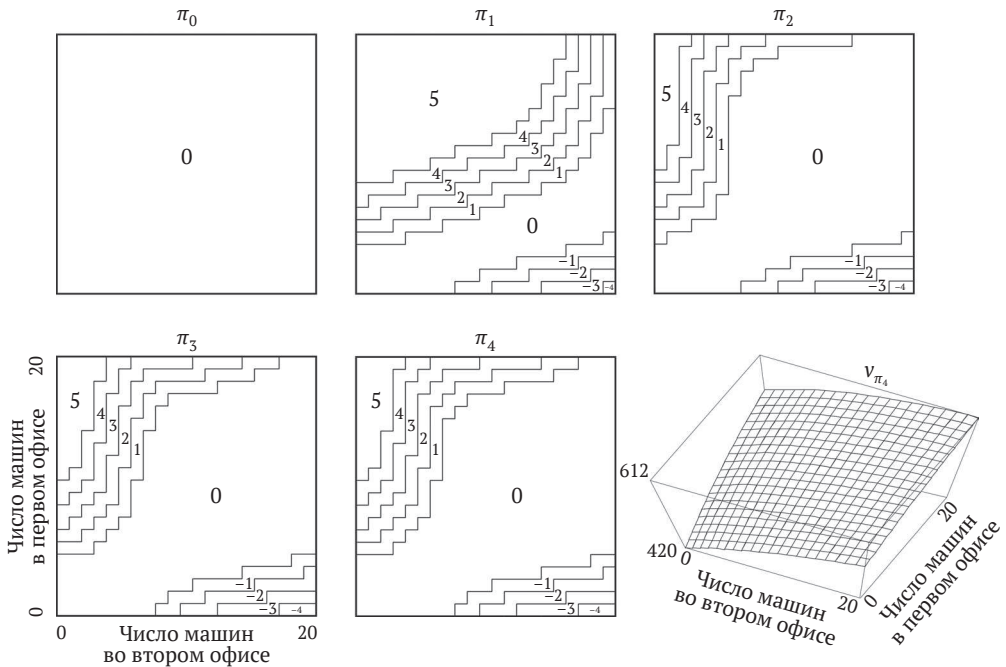


Рис. 4.2 ❖ Последовательность стратегий, найденная алгоритмом итерации по стратегиям в задаче об аренде машин, и окончательная функция ценности состояний. На первых пяти диаграммах для каждого количества машин в каждом офисе в конце дня показано количество машин, которые нужно перегнать из первого офиса во второй (отрицательные числа означают, что перегонять нужно из второго офиса в первый). Каждая последующая стратегия строго лучше предыдущей, а последняя оптимальна

Упражнение 4.4. В алгоритме итерации по стратегиям на стр. 110 есть тонкая ошибка, состоящая в том, что он может никогда не завершиться, если стратегия постоянно переключается между двумя или более стратегиями, которые одинаково хороши. В педагогических целях это, может, и неплохо, но на практике не годится. Модифицируйте псевдокод, так чтобы сходимость была гарантирована. □

Упражнение 4.5. Как следовало бы определить алгоритм итерации по стратегиям для ценностей действий? Приведите полный алгоритм вычисления q_* , аналогич-

ный показанному на стр. 110 для вычисления v_* . Обратите особое внимание на это упражнение, потому что развиваемые в нем идеи будут использоваться в книге далее. $\square$

Упражнение 4.6. Предположим, что мы ограничены только ε -мягкими стратегиями, для которых вероятность выбора любого действия в каждом состоянии s не меньше $\varepsilon/|\mathcal{A}(s)|$. Качественно опишите изменения, которые нужно было бы внести на шагах 3, 2 и 1 (именно в таком порядке) приведенного на стр. 110 алгоритма итерации по стратегиям для вычисления v_* . $\square$

Упражнение 4.7 (требуется программирование). Напишите программу, реализующую итерацию по стратегиям, и решите задачу об аренде машин со следующими изменениями. Один из служащих Джека в первом офисе каждую ночь ездит на автобусе домой, а живет он рядом со вторым офисом. Он с удовольствием готов перегонять одну машину во второй офис бесплатно. Но каждая дополнительная машина все равно стоит 2 доллара, равно как и перегон всех машин в обратном направлении. Кроме того, в каждом офисе у Джека ограниченное количество парковочных мест. Если на ночь остается больше 10 машин (после всех перегонов), то за пользование другой парковкой взимается дополнительно 4 доллара (вне зависимости от того, сколько машин там стоит). Такого рода нелинейности и произвольная динамика часто встречаются в реальных задачах, и все методы оптимизации, кроме динамического программирования, к ним применимы с трудом. Для проверки своей программы сначала воспроизведите результаты для исходной задачи. $\square$

4.4. ИТЕРАЦИЯ ПО ЦЕННОСТИ

Недостаток итерации по стратегиям заключается в том, что на каждой итерации нужно оценивать стратегию, а это само по себе длительное итеративное вычисление, требующее нескольких проходов по множеству состояний. Если оценивание стратегии производится итеративно, то сходимость точно к v_* возможна только в пределе. Должны ли мы дожидаться точной сходимости или можем остановиться раньше? Пример на рис. 4.1 наводит на мысль, что оценивание стратегии можно прервать досрочно. В этом примере все итерации, начиная с четвертой, никак не влияют на соответствующую жадную стратегию.

На самом деле шаг оценивания стратегии в алгоритме итерации по стратегиям можно усечь несколькими способами, не жертвуя гарантиями сходимости. Важный частный случай – остановка оценивания после всего одного прохода (одного обновления каждого состояния). Этот алгоритм называется *итерацией по ценности*. Его можно записать в виде особенно простой операции обновления, которая объединяет шаги улучшения стратегии и усеченного оценивания стратегии:

$$\begin{aligned} v_{k+1}(s) &\doteq \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s, A_t = a] \\ &= \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_k(s')] \end{aligned} \quad (4.10)$$

для всех $s \in \mathcal{S}$. Можно показать, что для произвольного v_0 последовательность $\{v_k\}$ сходится к v_* при тех же условиях, которые гарантируют существование v_* .

Другой способ интерпретировать итерацию по ценности – сослаться на уравнение оптимальности Беллмана (4.1). Заметим, что алгоритм итерации по ценности получается просто преобразованием уравнения оптимальности в правило обновления. Также заметим, что обновление в алгоритме итерации по ценности идентично обновлению в алгоритме оценивания стратегии (4.5) с тем отличием, что нужно взять максимум по всем действиям. По-другому эту тесную связь можно увидеть, сравнив диаграммы предшествующих состояний для этих алгоритмов на стр. 87 (оценивание стратегии) и в левой части рис. 3.4 (итерация по ценности). Это две естественные операции обновления при вычислении v_π и v_* .

Наконец, рассмотрим, как завершается итерация по ценности. Как и при оценивании стратегии, в алгоритме итерации по ценности для точной сходимости к v_* формально требуется бесконечное количество итераций. Но на практике мы останавливаемся, когда функция ценности мало изменяется после прохода. Во врезке ниже приведен полный алгоритм с таким условием остановки.

Алгоритм итерации по ценности для оценивания $\pi \approx \pi_*$

Параметр алгоритма: небольшая пороговая величина $\theta > 0$, определяющая точность оценки

Инициализировать $V(s)$ для всех $s \in \mathcal{S}^+$ произвольным образом с той оговоркой, что $V(\text{terminal}) = 0$

Повторять:

$\Delta \leftarrow 0$

Повторять для каждого $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

пока не окажется $\Delta < \theta$

Вывести детерминированную стратегию $\pi \approx \pi_*$ такую, что

$\pi(s) \leftarrow \arg \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$

Алгоритм итерации по ценности на каждом проходе объединяет один проход алгоритма оценивания стратегии с одним проходом алгоритма улучшения стратегии. Зачастую удается достичь более быстрой сходимости, вставляя несколько проходов оценивания между каждыми двумя проходами улучшения. В общем случае весь класс усеченных алгоритмов итерации по стратегиям можно рассматривать как последовательности проходов, на одних из которых производится шаг обновления алгоритма оценивания стратегии, а на других – шаг обновления алгоритма итерации по ценности. Поскольку операция взятия максимума в (4.10) – единственное различие между этими шагами обновления, это означает, что на некоторых проходах алгоритма оценивания стратегии добавляется операция $\max$. Все эти алгоритмы сходятся к оптимальной политике для конечных МППР с обесцениванием.

Пример 4.3: задача игрока. Игрок может ставить на исходы подбрасывания монеты. Если выпадает орел, то он выигрывает столько долларов, сколько поставил,

а если решка, то теряет свою ставку. Игра заканчивается, когда игрок либо выигрывает, достигнув поставленной цели 100 долларов, либо проигрывает все свои деньги. При каждом подбрасывании игрок должен решить, какую часть своего капитала (целое количество долларов) поставить. Эту задачу можно рассматривать как эпизодический конечный МППР без обесценивания. Состоянием является капитал игрока, $s \in \{1, 2, \dots, 99\}$, а действиями – ставки, $a \in \{0, 1, \dots, \min(s, 100 - s)\}$. Вознаграждение равно нулю для всех переходов, кроме тех, на которых игрок достигает своей цели, а в этом случае оно равно $+1$. Тогда функция ценности состояний дает вероятность выигрыша при старте из каждого состояния. Стратегия отображает величины капитала на ставки. Оптимальная стратегия максимизирует вероятность достижения цели. Обозначим p_h вероятность выпадения орла. Если p_h известна, то определена и вся задача, и ее можно решить, например, итерацией по ценности. На рис. 4.3 показано изменение функции ценности на последовательных проходах алгоритма итерации по ценности, а также конечная стратегия, найденная, например, для $p_h = 0.4$. Эта стратегия оптимальна, но она не единственная. На самом деле существует целое семейство оптимальных стратегий, все они соответствуют различным вариантам разрешения неоднозначности выбора стратегии с помощью функции $\arg\max$ относительно оптимальной функции ценности. Понимаете ли вы, как выглядит все семейство?

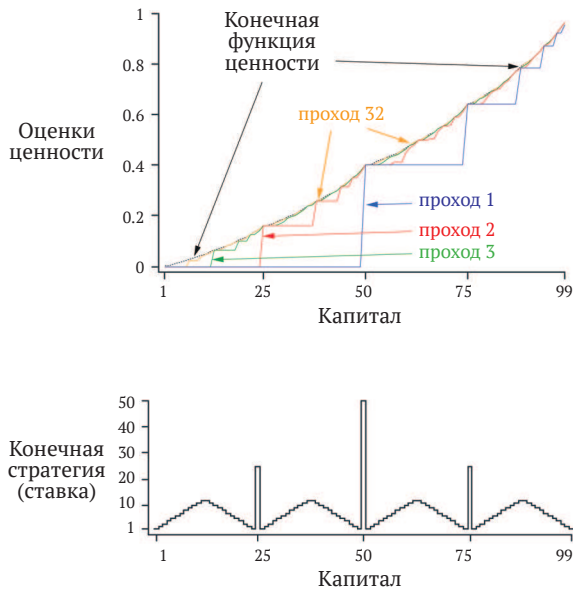


Рис. 4.3 ❖ Решение задачи игрока для $p_h = 0.4$. На верхнем графике показана функция ценности, найденная с помощью последовательных проходов алгоритма итерации по ценности. На нижнем графике показана конечная стратегия

Упражнение 4.8. Почему у оптимальной политики в задаче игрока такая странная форма? В частности, для капитала 50 предлагается поставить все на одно подбрасывание, а для капитала 51 – нет. Почему такая стратегия хороша? □

Упражнение 4.9 (требуется программирование). Реализуйте алгоритм итерации по ценности для задачи игрока и решите ее для $p_h = 0.25$ и $p_h = 0.55$. Возможно, вы сочтете удобным ввести два фиктивных состояния, соответствующих завершению с капиталом 0 и 100, сопоставив им ценности 0 и 1. Представьте свои результаты графически, как на рис. 4.3. Является ли полученное решение устойчивым при $\theta \rightarrow 0$? □

Упражнение 4.10. Каков аналог обновления в алгоритме итерации по ценности (4.10) для ценностей действий $q_{k+1}(s, a)$? □

4.5. АСИНХРОННОЕ ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Основной недостаток рассмотренных выше алгоритмов ДП состоит в том, что они включают операции над всем множеством состояний, т. е. требуют прохода по множеству состояний. Если множество состояний очень велико, то даже один проход может стоить недопустимо дорого. Например, в игре в нарды 10^{20} состояний. Даже если бы мы могли выполнять миллион обновлений состояний в секунду, то только один полный проход занял бы больше тысячи лет.

Асинхронные алгоритмы ДП – это итеративные алгоритмы ДП с обновлением на месте, которые организованы не как систематические проходы по множеству состояний. Эти алгоритмы обновляют ценности состояний в произвольном порядке, используя те ценности других состояний, которые существуют на данный момент времени. Но для сходимости асинхронный алгоритм должен продолжать обновлять ценности всех состояний, он не может игнорировать какие-то состояния, достигнув в вычислениях определенной точки. Асинхронные алгоритмы ДП демонстрируют большую гибкость при выборе состояния, подлежащего обновлению.

Например, в одном варианте асинхронного алгоритма итерации по ценности на месте обновляется ценность только одного состояния s_k на каждом шаге k , при этом применяется формула (4.10). Если $0 \leq \gamma < 1$, то асимптотическая сходимость к v_* гарантируется при условии, что каждое состояние встречается в последовательности $\{s_k\}$ бесконечное число раз (последовательность может быть даже стохастической). (В эпизодическом случае без обесценивания может случиться, что при некоторых порядках обновления сходимость отсутствует, но таких порядков сравнительно нетрудно избежать.) Аналогично мы можем чередовать обновления в алгоритмах оценивания стратегии и итерации по ценности, получая в результате вариант асинхронной усеченной итерации по стратегиям. Хотя детали этого и еще более экзотических алгоритмов ДП выходят за рамки данной книги, ясно, что имеется возможность гибкого использования различных форм обновления в широком спектре алгоритмов ДП без прохода по множеству состояний.

Разумеется, вовсе необязательно, что, обойдясь без прохода по множеству состояний, мы сумеем уменьшить объем вычислений. Это просто означает, что алгоритм не будет топтаться на месте, совершая безнадежно долгий проход, прежде чем сможет улучшить стратегию. Мы можем попытаться воспользоваться этим, выбирая, к каким состояниям применить обновления, чтобы улучшить скорость продвижения алгоритма вперед. Можно попробовать упорядочить обновления так, чтобы информация эффективно распространялась из одного состояния в дру-

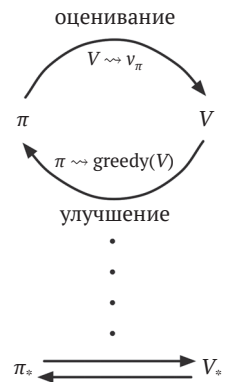
гое. Ценность некоторых состояний, быть может, не нужно обновлять так часто, как других. Можно даже попробовать вообще пропускать обновление некоторых состояний, если они никак не связаны с оптимальным поведением. Кое-какие идеи о том, как это сделать, обсуждаются в главе 8.

Асинхронные алгоритмы также упрощают чередование вычислений с взаимодействием в реальном времени. Чтобы решить заданный МППР, мы можем прогонять итеративный алгоритм ДП *одновременно с тем, как агент фактически взаимодействует с МППР*. Опыт, приобретенный агентом, можно использовать для определения состояний, к которым алгоритм ДП должен применить свои обновления. И одновременно последняя вычисленная алгоритмом ДП ценность и информация о стратегии могут служить основной для принятия решений агентом. Например, можно применять обновления к состояниям по мере посещения их агентом. Это позволяет *сфокусировать* обновления алгоритма ДП на тех частях множества состояний, которые наиболее релевантны агенту. Такого рода фокусировка – тема, проходящая через все обучение с подкреплением.

4.6. ОБОБЩЕННАЯ ИТЕРАЦИЯ ПО СТРАТЕГИЯМ

Итерация по стратегиям состоит из двух одновременных взаимодействующих процессов, один из которых делает функцию ценности согласованной с текущей стратегией (оценивание стратегии), а другой делает стратегию жадной относительно текущей функции ценности (улучшение стратегии). В ходе итерации по стратегиям эти процессы чередуются, при этом каждый завершается до того, как начнется другой, хотя это необязательно. Например, в алгоритме итерации по ценности между каждыми двумя улучшениями стратегии выполняется только одна итерация оценивания стратегии. В асинхронных методах ДП процессы оценивания и улучшения перемежаются даже чаще. В некоторых случаях в одном процессе обновляется одно состояние, а затем происходит возврат к другому процессу. Коль скоро оба процесса продолжают обновлять все состояния, конечный результат обычно одинаковый – сходимости к оптимальной функции ценности и оптимальной стратегии.

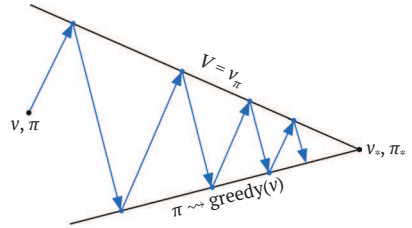
Мы будем использовать термин *обобщенная итерация по стратегиям* (ОИС) для обозначения общей идеи – разрешить процессам оценивания и улучшения стратегии взаимодействовать безотносительно к гранулярности и другим деталям процессов. Почти все методы обучения с подкреплением хорошо описываются как ОИС. То есть во всех них можно выделить стратегии и функции ценности, причем стратегия всегда улучшается относительно функции ценности, а функция ценности всегда стремится к функции ценности для этой стратегии, как показано на диаграмме справа. Легко видеть, что если и процесс оценивания, и процесс улучшения стабилизировались, т. е. ни один не порождает изменений, то функция ценности и стратегия должны быть оптимальными. Функция ценности стабилизируется, только если она согласована с текущей стратегией, а стратегия стабилизируется, только если является жадной относительно текущей функции



ценности. Таким образом, оба процесса стабилизированы, только если найдена стратегия, жадная относительно своей собственной функции ценности. Отсюда следует, что удовлетворяется уравнение оптимальности Беллмана (4.1) и что эта стратегия и эта функция ценности оптимальны.

Процессы оценивания и улучшения в ОИС можно рассматривать одновременно как конкурирующие и кооперативные. Они конкурируют в том смысле, что тянут в разные стороны. Стремление сделать стратегию жадной относительно функции ценности обычно делает эту функцию некорректной для изменившейся стратегии, а стремление сделать функцию ценности согласованной со стратегией обычно приводит к тому, что стратегия перестает быть жадной. Однако в долгосрочной перспективе эти процессы совместно находят единственное общее решение: оптимальную функцию ценности и оптимальную стратегию.

Можно также интерпретировать взаимодействие между процессами оценивания и улучшения в ОИС в терминах двух ограничений, или целей, например в виде двух прямых в двумерном пространстве, показанных на диаграмме справа. Хотя реальная геометрия намного сложнее, эта диаграмма дает представление о том, что происходит. Каждый процесс направляет функцию ценности или стратегию к одной из двух прямых, соответствующих решению для одной из двух целей. Цели взаимодействуют, потому что прямые не перпендикулярны. Приближение к одной цели вызывает отдаление от другой. Однако процесс в целом неуклонно продвигается в направлении к оптимальности. Стрелки на диаграмме соответствуют поведению алгоритма итерации по стратегиям в том смысле, что каждая направляет систему к полному достижению одной из двух целей. В ОИС можно также совершать более мелкие, неполные шаги к целям. В любом случае оба процесса вместе достигают общей цели оптимальности, хотя ни один из них не пытается достичь ее напрямую.



4.7. ЭФФЕКТИВНОСТЬ ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ

ДП не всегда практически применимо для очень больших задач, но по сравнению с другими методами решения МППР методы динамического программирования весьма эффективны. Если оставить в стороне несколько технических деталей, то время (в худшем случае), необходимое методу ДП для нахождения оптимальной стратегии, полиномиально зависит от количества состояний и действий. Если обозначить n и k количество состояний и действий соответственно, то это означает, что количество вычислительных операций, выполняемых методом ДП, меньше некоторой полиномиальной функции от n и k . Метод ДП гарантированно находит оптимальную стратегию за полиномиальное время, хотя общее количество (детерминированных) стратегий равно k^n . В этом смысле ДП экспоненциально быстрее любого прямого поиска в пространстве стратегий, поскольку для предоставления такой же гарантии прямому методу пришлось бы исследовать все без исключения стратегии. К решению МППР применимы также методы линейного программирования, и в некоторых случаях их гарантии сходимости в худшем

случае лучше тех, что дают методы ДП. Но методы линейного программирования становятся практически неприменимыми при гораздо меньшем количестве состояний, чем методы ДП (примерно в 100 раз). Для самых больших задач работают только методы ДП.

Иногда считают, что применимость ДП ограничена из-за проклятия размерности – того факта, что количество состояний растет экспоненциально с увеличением числа переменных состояния. Большие множества состояний действительно создают трудности, но это трудности, присущие самой задаче, а не ДП как методу ее решения. На самом деле ДП больше пригодно для обработки больших пространств состояний, чем такие альтернативные методы, как прямой поиск и линейное программирование.

На практике методы ДП в сочетании с современными компьютерами позволяют решать МППР с миллионами состояний. Широко используются оба алгоритма – итерации по стратегиям и итерации по ценности, – и не ясно, какой из них лучше в общем случае. На практике эти методы обычно сходятся гораздо быстрее, чем предсказывает теория для худшего случая, особенно если задать хорошие начальные функции ценности или стратегии.

В задачах с большими пространствами состояний часто предпочтительнее асинхронные методы ДП. Для завершения всего одного прохода синхронный метод должен выполнить вычисления и получить память для каждого состояния. В некоторых задачах даже это практически невозможно, хотя потенциально задача разрешима, потому что количество состояний вдоль оптимальных траекторий сравнительно невелико. В таких случаях применение асинхронных методов и других вариантов ОИС, возможно, позволит найти хорошие или оптимальные стратегии гораздо быстрее.

4.8. РЕЗЮМЕ

В этой главе мы познакомились с основными идеями и алгоритмами динамического программирования в применении к решению конечных МППР. Под *оцениванием стратегии* понимается (обычно) итеративное вычисление функций ценности для заданной стратегии. Под *улучшением стратегии* понимается вычисление улучшенной стратегии при заданной функции ценности для этой стратегии. Объединяя оба вычисления, мы получаем алгоритм *итерации по стратегиям* и алгоритм *итерации по ценности* – два наиболее популярных метода ДП. Любой из них можно использовать для надежного вычисления оптимальных стратегий и функций ценности для конечных МППР, если вся информация о МППР известна.

Классические методы ДП выполняют несколько проходов по множеству состояний, производя операцию *полного обновления* каждого состояния. Каждая такая операция обновляет ценность одного состояния на основе ценности всех возможных последующих состояний и их вероятностей. Полные обновления тесно связаны с уравнениями Беллмана: они лишь немногим больше, чем эти уравнения, преобразованные в операторы присваивания. Если в результате обновления ценность перестает изменяться, значит, алгоритм сошелся к ценностям, удовлетворяющим соответствующему уравнению Беллмана. Четырем основным функциям ценности (v_π , v_* , q_π и q_*) соответствуют четыре уравнения Беллмана и четыре пол-

ных обновления. Интуитивное представление о работе обновлений в алгоритмах ДП дают их *диаграммы предшествующих состояний*.

Лучше понять методы ДП, да и почти все методы обучения с подкреплением, можно, если рассматривать их как *обобщенную итерацию по стратегиям* (ОИС). Это общая идея двух взаимодействующих процессов, аппроксимирующих стратегию и функцию ценности. Один процесс принимает на входе стратегию и занимается ее оценением, изменяя функцию ценности, так чтобы она больше походила на истинную функцию ценности для этой стратегии. Другой процесс принимает на входе функцию ценности и производит некое улучшение стратегии, стремясь сделать ее лучше в предположении, что заданная функция ценности совпадает с функцией ценности стратегии. Хотя каждый процесс изменяет исходные данные для другого, вместе они находят общее решение: стратегию и функцию ценности, которые не изменяются обоими процессами и, следовательно, оптимальны. В некоторых случаях можно доказать, что ОИС сходится, и прежде всего это справедливо для классических методов ДП, описанных в этой главе. В других случаях сходимость не доказана, но все равно идея ОИС позволяет лучше понять методы.

Методы ДП необязательно должны выполнять полные проходы по множеству состояний. *Асинхронные методы ДП* – это итеративные методы, обновляющие состояние на месте в произвольном порядке, быть может, стохастическом и с использованием неактуальной информации. Многие такого рода методы можно рассматривать как мелкоструктурные формы ОИС.

Наконец, отметим одно специальное свойство методов ДП. Все они обновляют оценки ценности состояний на основе оценок ценности последующих состояний. То есть оценки обновляются на основе других оценок. Эта общая идея называется *бутстрэппингом*. Многие методы обучения с подкреплением выполняют бутстрэппинг, даже те, которым, в отличие от ДП, не требуется полная и точная модель окружающей среды. В следующей главе мы будем изучать методы обучения с подкреплением, которым не нужна модель и которые обходятся без бутстрэппинга. А затем – методы, в которых модель не нужна, но бутстрэппинг присутствует. Эти важнейшие свойства в принципе разделимы, но их можно сочетать в различных интересных комбинациях.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

Термин «динамическое программирование» ввел в употребление Беллман (Bellman, 1957a), который показал, как эти методы применяются к широкому кругу задач. Подробное рассмотрение ДП можно найти во многих учебниках, включая Bertsekas (2005, 2012), Bertsekas and Tsitsiklis (1996), Dreyfus and Law (1977), Ross (1983), White (1969) и Whittle (1982, 1983). Наш интерес к ДП ограничен его использованием для решения МППР, но ДП применимо и к задачам других типов. В работе Kumar and Kanal (1988) приводится более общий взгляд на ДП.

Насколько мы знаем, впервые связь между ДП и обучением с подкреплением была установлена в работе Minsky (1961), где обсуждалась программа Сэмюэла для игры в шашки. В сноске Мински упомянул, что ДП применимо к задачам, в которых процесс обновления Сэмюэла можно разрешить в замкнутой аналити-

ческой форме. Это замечание, возможно, заставило исследователей искусственного интеллекта ошибочно заключить, что ДП ограничено только аналитически решаемыми задачами, а потому не имеет отношения к искусственному интеллекту. В работе Andreae (1969b) ДП, а именно итерация по стратегиям, упомянуто в контексте обучения с подкреплением, хотя не прослежены какие-то конкретные связи между ДП и алгоритмами обучения. В работе Werbos (1977) предложен подход к приближенному ДП, которое автор назвал «эвристическим динамическим программированием» и в котором заметную роль играют методы градиентного спуска для задач с непрерывным пространством состояний (Werbos, 1982, 1987, 1988, 1989, 1992). Эти методы тесно связаны с алгоритмами обучения с подкреплением, обсуждаемыми в настоящей книге. В работе Watkins (1989) явно установлена связь между обучением с подкреплением и ДП и охарактеризован класс методов обучения с подкреплением, названный «инкрементным динамическим программированием».

4.1–4 В этих разделах описываются хорошо изученные и зарекомендовавшие себя алгоритмы ДП, которые рассматриваются в любом из вышеупомянутых общих учебников по ДП. Теорема об улучшении стратегии и алгоритм итерации по стратегиям предложены в работах Bellman (1957a) и Howard (1960). На наше изложение оказала влияние локальная интерпретация улучшения стратегии, приведенная в работе Watkins (1989). Наше обсуждение итерации по ценности как формы усеченного алгоритма итерации по стратегиям основано на подходе работы Puterman and Shin (1978), где представлен класс алгоритмов, названный *модифицированной итерацией по стратегиям*, в который итерация по стратегиям и итерация по ценности входят как частные случаи. Анализ, показывающий, как сделать так, чтобы алгоритм итерации по ценности находил оптимальную стратегию за конечное время, имеется в работе Bertsekas (1987).

Итеративное оценивание стратегии – пример классического алгоритма последовательного приближения для решения системы линейных уравнений. Вариант алгоритма с двумя массивами, в одном из которых хранятся старые, а в другом обновляемые значения, часто называют алгоритмом *в духе Якоби*, поскольку именно Якоби первым использовал этот метод. Иногда его называют также *синхронным* алгоритмом, поскольку выглядит это так, будто все значения обновляются одновременно. Второй массив необходим, чтобы смоделировать параллельное вычисление последовательным. Вариант алгоритма с обновлением на месте часто называют алгоритмом *в духе Гаусса–Зейделя*, потому что так выглядит классический алгоритм Гаусса–Зейделя решения систем линейных уравнений. Такие два варианта существуют и у других алгоритмов ДП, помимо итеративного оценивания стратегий. В работе Bertsekas and Tsitsiklis (1989) приведен прекрасный обзор этих вариантов и их различий с точки зрения производительности.

4.5 Асинхронные алгоритмы впервые появились в работах Bertsekas (1982, 1983), который называл их распределенными. Идея первоначально зародилась в процессе реализации алгоритмов ДП в многопроцессорной системе, характеризуемой задержками передачи данных между процессорами и отсутствием глобального синхрогенератора. Эти алгоритмы подробно обсуж-

дались в работе Bertsekas and Tsitsiklis (1989). Алгоритмы ДП в духе Якоби и Гаусса–Зейделя – частные случаи асинхронной версии. В работе Williams and Baird (1990) представлены асинхронные алгоритмы ДП, не столь крупноструктурные, как в тексте: сами операции обновления разбиты на шаги, которые можно выполнять асинхронно.

- 4.7 Этот раздел, в написании которого помогал Майкл Литтман, основан на работе Littman, Dean, and Kaelbling (1995). Фразу «проклятие размерности» ввел в обиход Беллман (Bellman, 1957a).

Работа, заложившая основы применения линейного программирования к обучению с подкреплением, проделана Даниэлой де Фариас (de Farias, 2002; de Farias and Van Roy, 2003).

Глава 5

Методы Монте-Карло

В этой главе мы рассмотрим наши первые методы обучения для оценивания функций ценности и нахождения оптимальных стратегий. В отличие от предыдущей главы, мы теперь не будем предполагать наличия полной информации об окружающей среде. Обучение на *реальном* опыте удивительно тем, что позволяет достичь оптимального поведения без каких-либо априорных знаний о динамике среды. Обучение на *имитированном* опыте также приносит весомые плоды. Хотя модель требуется, она должна лишь генерировать выборочные переходы, а не полное распределение вероятностей всех возможных переходов, необходимое в динамическом программировании (ДП). Есть на удивление много случаев, когда можно без особого труда сгенерировать выборку из опыта в соответствии с желательным распределением вероятностей, но невозможно получить распределение в явном виде.

Методы Монте-Карло – это способ решения задачи обучения с подкреплением на основе усреднения выборочного дохода. Чтобы гарантировать доступность корректно определенных доходов, мы здесь определим методы Монте-Карло только для эпизодических задач. То есть предполагается, что опыт разделен на эпизоды и что каждый эпизод в конечном итоге завершается вне зависимости от того, какие действия выбирались. Только по завершении эпизода производится изменение стратегий и оценок ценности.

Таким образом, методы Монте-Карло могут быть инкрементными в том смысле, что они применяются поэпизодно, но не пошагово (как в онлайн-режиме). Термин «Монте-Карло» часто употребляется более широко и обозначает любой метод оценивания, в работе которого участвует значимый случайный компонент. Здесь же мы используем его только для методов, основанных на усреднении полного дохода (в противоположность методам, которые обучаются на частичных доходах и рассматриваются в следующей главе).

Методы Монте-Карло производят выборку и усреднение *доходов* для каждой пары состояние–действие – так же, как методы бандитов, изученные в главе 2, производят выборку и усреднение *вознаграждений* за каждое действие. Основное отличие заключается в том, что состояний теперь несколько, и каждое выступает в роли отдельной задачи о бандите (например, ассоциативного поиска или кон-

текстуального бандита), и разные задачи о бандитах взаимосвязаны. Иначе говоря, доход после выбора действия в одном состоянии зависит от действий, выбранных в более поздних состояниях того же эпизода. Поскольку любой выбор действия – это замаскированное обучение, задача оказывается стационарной с точки зрения предшествующего состояния.

Чтобы справиться с нестационарностью, мы адаптируем идею обобщенной итерации по стратегиям (ОИС), изложенную в главе 4 для ДП. Если там мы *вычисляли* функции ценности, опираясь на знание МППР, то здесь *обучаем* функции ценности на выборочных доходах в МППР. Функции ценности и соответствующие стратегии взаимодействуют для достижения оптимальности, по существу, точно так же (ОИС). Как и в главе, посвященной ДП, мы сначала рассмотрим задачу предсказания (вычисление v_π и q_π для произвольной, но фиксированной стратегии π), затем задачу об улучшении стратегии и, наконец, задачу управления и ее решение посредством ОИС. Все эти идеи заимствованы из ДП и обобщены на случай методов Монте-Карло, в котором имеется только выборочный опыт.

5.1. ПРЕДСКАЗАНИЕ МЕТОДАМИ МОНТЕ-КАРЛО

Начнем с применения методов Монте-Карло к обучению функции ценности состояний для заданной стратегии. Напомним, что ценность состояния – это ожидаемый доход – ожидаемое накопленное будущее обесцененное вознаграждение – при старте из этого состояния. Тогда очевидный способ его оценки на опыте состоит в том, чтобы просто усреднить доходы, наблюдавшиеся после посещения этого состояния. По мере увеличения количества наблюдавшихся доходов среднее должно сходиться к математическому ожиданию. Эта идея лежит в основе всех методов Монте-Карло.

В частности, предположим, что требуется оценить $v_\pi(s)$, ценность состояния s при следовании стратегии π , если дано множество эпизодов, полученных в результате следования π с прохождением через s . Каждое вхождение состояния s в эпизод называется *посещением* s . Разумеется, s можно посещать несколько раз в одном эпизоде; будем называть первый случай его посещения в эпизоде *первым посещением* s . Метод МК первого посещения оценивает $v_\pi(s)$ как среднее доходов, полученных после первого посещения s , тогда как метод МК всех посещений усредняет доходы, полученные после всех посещений s . Эти два вида методов Монте-Карло (МК) очень похожи, но их теоретические свойства немного различаются. Изучение МК первого посещения началось в 1940-х годах, они изучены наиболее полно, и именно им мы уделим внимание в этой главе. МК всех посещений более естественно обобщаются на аппроксимацию функций и следы преемственности, мы будем обсуждать их в главах 9 и 12. Во врезке ниже МК первого посещения изображен в процедурной форме. В МК всех посещений отсутствовала бы только проверка того, встречалось ли состояние S_t ранее в этом эпизоде.

Предсказание методом МК первого посещения для оценивания $V \approx v_\pi$

Вход: стратегия π , подлежащая оцениванию

Инициализация:

$V(s) \in \mathbb{R}$, присваиваются произвольные значения для всех $s \in \mathcal{S}$

$Returns(s) \leftarrow$ пустой список для всех $s \in \mathcal{S}$

Повторять бесконечно (для каждого эпизода):

Сгенерировать эпизод, следуя π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Повторять для каждого шага эпизода $t = T - 1, T - 2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Если S_t не встречается в $S_0, S_1, \dots, S_{t-1}$:

Добавить G в $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

И МК первого посещения, и МК всех посещений сходятся к $v_\pi(s)$, когда количество посещений (или первых посещений) s стремится к бесконечности. Это легко видеть для МК первого посещения. В этом случае все доходы – независимые и одинаково распределенные оценки $v_\pi(s)$ с конечной дисперсией. По закону больших чисел, последовательность средних этих оценок сходится к их математическому ожиданию. Каждое среднее само по себе является несмещенной оценкой, а стандартное отклонение его ошибки убывает как $1/\sqrt{n}$, где n – количество усредненных доходов. МК всех посещений несколько сложнее, но его оценка также квадратично сходится к $v_\pi(s)$ (Singh and Sutton, 1996).

Применение методов Монте-Карло лучше проиллюстрировать на примере.

Пример 5.1: игра в блэкджек. Цель популярной в казино карточной игры блэкджек – собрать карты, сумма числовых достоинств которых как можно ближе к 21, но не превышает этого значения. Все картинки считаются как 10, а туз может считаться как 1 или 11. Мы рассматриваем вариант, в котором все игроки независимо играют против сдающего. В начале игры сдающему и игроку сдаются по две карты. Одна из карт сдающего кладется лицом вверх, другая – лицом вниз. Если у игрока сразу после сдачи на руках 21 очко (туз и карта достоинством 10), то это называется *блэкджек*. В этом случае игрок выигрывает, если только у сдающего нет на руках блэкджека, а в этом случае игра заканчивается вничью. Если у игрока нет блэкджека, то он может попросить дополнительные карты, по одной (*еще*), пока не решит, что достаточно (*хватит*), или сумма очков не превысит 21 (*перебор*). Перебор означает проигрыш; если игрок говорит «хватит», то очередь переходит к сдающему. Сдающий набирает карты или останавливается в соответствии с фиксированной стратегией, не допускающей выбора: он останавливается, набрав 17 или более очков, в противном случае берет еще. Если у сдающего перебор, выигрывает игрок, в противном случае исход – выигрыш, проигрыш или ничья – зависит от того, у кого сумма очков ближе к 21.

Игра в блэкджек естественно описывается как эпизодический конечный МППР. Каждая партия – отдельный эпизод. За выигрыш, проигрыш или ничью начисляется вознаграждение $+1$, -1 и 0 соответственно. Все вознаграждения в процессе игры нулевые, обесценивания нет ($\gamma = 1$); поэтому вознаграждение в конце игры одновременно является доходом. Действия игрока – еще или хватит. Состояния зависят от карт игрока и открытой карты сдающего. Предполагается, что карты сдаются из бесконечной колоды (т. е. имеет место выборка с возвращением), поэтому не имеет смысла следить за тем, какие карты уже сданы. Если игрок получил туза, который мог бы засчитать как 11 без перебора, то говорят, что этот туз *играющий*. В таком случае он всегда засчитывается как 11, поскольку при засчитывании его как 1 сумма оказалась бы 11 или меньше, а тогда нет никакой свободы выбора, потому что игроку, очевидно, нужно взять еще карту. Таким образом, игрок принимает решения, исходя из трех переменных: своей текущей суммы очков (12–21), открытой карты сдающего (туз–10) и наличия у себя играющего туза. Всего получается 200 состояний.

Рассмотрим стратегию, при которой игрок говорит «хватит», если набрал 20 или 21 очко, а в противном случае просит еще карту. Чтобы найти функцию ценности состояний для этой стратегии методом Монте-Карло, необходимо смоделировать много партий с такой стратегией и усреднять доходы, полученные после каждого состояния. Именно так мы получили оценки функции ценности состояний, показанные на рис. 5.1. Оценки состояний с играющим тузом менее достоверны и менее регулярны, потому что такие состояния встречаются реже. Как бы то ни было, после 500 000 партий функция ценности аппроксимирована очень хорошо.

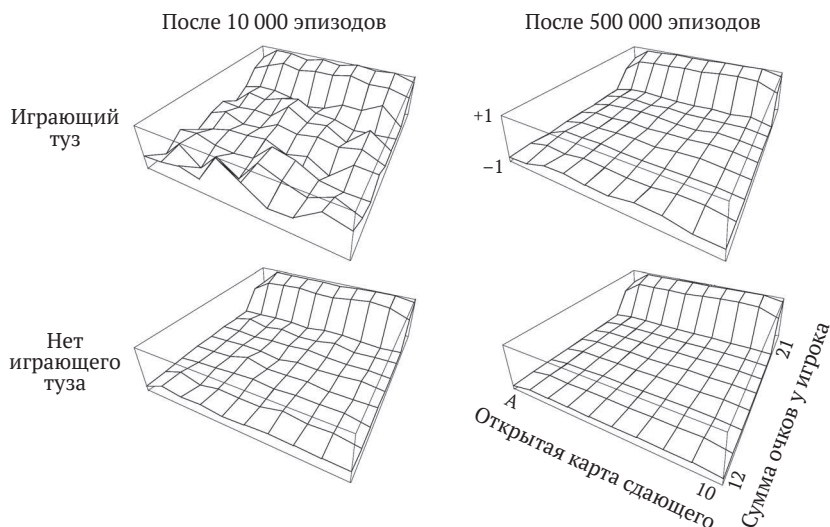


Рис. 5.1 ❖ Приближенные функции ценности состояний для стратегии игры в блэкджек, при которой игрок останавливается, набрав 20 или 21. Вычислены посредством оценивания стратегии методом Монте-Карло

Упражнение 5.1. Рассмотрим диаграммы на рис. 5.1 справа. Почему оценка функции ценности резко возрастает в последних двух строках на заднем плане? Почему она падает в последней строке на диаграмме слева? Почему значения на переднем плане на верхних диаграммах выше, чем на нижних? □

Упражнение 5.2. Предположим, что вместо МК первого посещения для моделирования игры в блэкджек использовался МК всех посещений. Как вы думаете, будет ли результат сильно отличаться? Объясните свой ответ. □

Хотя в задаче об игре в блэкджек мы располагаем полной информацией об окружающей среде, применить методы ДП к вычислению функции ценности было бы непросто. Для этого нужно знать распределение последующих событий, в частности необходима динамика среды, описываемая функцией p с четырьмя аргументами, а определить ее для блэкджека трудно. Например, предположим, что игрок набрал 14 очков и хочет остановиться. Какова вероятность, что игра завершится с вознаграждением +1 для него, выраженная в виде функции от открытой карты сдающего? *Прежде* чем применять ДП, требуется вычислить все вероятности, а зачастую это сложный и чреватый ошибками процесс. Напротив, сгенерировать выборочные партии, требуемые в методах Монте-Карло, легко. И так бывает на удивление часто; способность методов Монте-Карло работать с выборочными эпизодами может оказаться существенным преимуществом даже тогда, когда имеется полная информация о динамике окружающей среды.

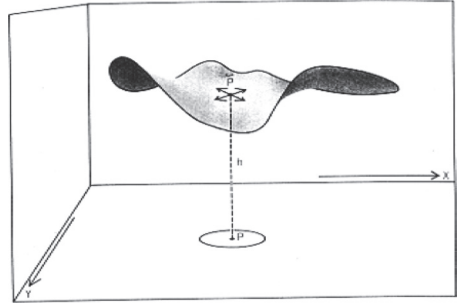
Можем ли мы обобщить идею диаграмм предшествующих состояний на алгоритмы Монте-Карло? Общая идея диаграммы предшествующих состояний – показать сверху корневой узел, подлежащий обновлению, а снизу – все переходы и листовые узлы, для которых вознаграждения и оценки ценности вносят вклад в обновление. Для оценивания v_π методами Монте-Карло корнем является узел состояния, а под ним расположена вся траектория переходов в одном конкретном эпизоде, заканчивающемся в заключительном состоянии (см. рисунок справа). Если на диаграмме ДП (стр. 87) показаны все возможные переходы, то на диаграмме Монте-Карло – только выборочные состояния в одном эпизоде. Если диаграмма ДП включает только одношаговые переходы, то диаграмма Монте-Карло продолжается до самого конца эпизода. Эти различия в диаграммах отражают фундаментальные различия между алгоритмами.

В методах Монте-Карло важно, что оценки для каждого состояния независимы. Оценка для одного состояния не строится на основе оценки для какого-либо другого состояния, как в случае ДП. Иными словами, в методах Монте-Карло нет бутстрэппинга в том смысле, который определен в предыдущей главе.

В частности, отметим, что вычислительная сложность оценивания ценности одного состояния не зависит от количества состояний. Благодаря этому методы Монте-Карло могут оказаться особенно привлекательными, когда требуется получить ценность только одного состояния или небольшого их подмножества. Можно сгенерировать большую выборку эпизодов, начинающихся в интересующих состояниях, и усреднить доходы, полученные только в этих состояниях, игнорируя все остальные. Это третье потенциальное преимущество методов Монте-Карло по сравнению с методами ДП (после способности к обучению на реальном и имитированном опыте).



Пример 5.2: мыльный пузырь. Предположим, что проволоочный каркас, образующий замкнутый контур, окунули в мыльный раствор, так что образовалась мыльная поверхность, или пузырь, натянутый на каркас. Если геометрия проволоочного каркаса нерегулярная, но известная, то как вычислить форму этой поверхности? Эта форма обладает тем свойством, что сумма сил, действующих на каждую точку со стороны ее соседей, равна нулю (иначе форма изменялась бы). Это значит, что высота поверхности в каждой точке равна средней высоте всех точек в небольшом круге с центром в данной точке. Кроме того, границы поверхности должны совпадать с каркасом. Обычный подход к решению таких задач – покрыть поверхность сеткой и находить высоту в узлах сетки с помощью итеративных вычислений. Граничные узлы сетки принудительно размещаются на проволоочном каркасе, а во всех остальных высота принимается равной средней высоте четырех ближайших соседей. Этот процесс затем подвергается итерациям, как при итеративном оценивании стратегии в ДП, и в конечном итоге сходится к хорошей аппроксимации поверхности.



Мыльный пузырь на проволоочном каркасе

Воспроизводится с разрешения авторов из работы Hersh and Griego (1969).

© 1969 Scientific American, подразделение Nature America, Inc. Все права защищены

Это похоже на задачи, для которых изначально предназначались методы Монте-Карло. Представьте, что вместо описанного выше итеративного вычисления вы стоите на поверхности и случайно блуждаете по ней, делая с равной вероятностью шаг в один из соседних узлов сетки, пока не достигнете границы. Оказывается, что математическое ожидание высоты в граничной точке может служить близкой аппроксимацией высоты поверхности в начальной точке (на самом деле оно в точности совпадает со значением, вычисленным описанным выше итеративным алгоритмом). Таким образом, можно аппроксимировать высоту поверхности в любой точке, просто усреднив высоты граничных точек, в которых завершаются начатые из нее случайные блуждания. Если нас интересует значение только в одной точке или в небольшом фиксированном множестве точек, то такой метод Монте-Карло может быть куда эффективнее итеративного метода, основанного на локальной согласованности.

5.2. ОЦЕНИВАНИЕ ЦЕННОСТИ ДЕЙСТВИЙ МЕТОДОМ МОНТЕ-КАРЛО

Если модель недоступна, то особенно полезно оценивать ценность *действий* (пар состояние–действие), а не состояний. При наличии модели одной лишь ценности состояний достаточно для определения стратегии; нужно просто заглядывать на один шаг вперед и выбирать то действие, которое приводит к наилучшей комбинации вознаграждения и следующего состояния, как мы делали в главе о ДП. Но без модели знания ценности состояния не достаточно. Необходимо явно оце-

нить ценность каждого действия, если мы хотим, чтобы ценности были полезны для выработки стратегии. Поэтому одна из главных целей методов Монте-Карло – оценить q_* . Чтобы добиться этого, мы сначала рассмотрим задачу оценивания стратегии ради ценности действий.

Задача оценивания стратегии ради ценности действий заключается в том, чтобы оценить $q_\pi(s, a)$, ожидаемый доход при старте из состояния s , выборе действия a и далее следования стратегии π . Методы Монте-Карло для этой задачи по существу такие же, как представленные выше для нахождения ценности состояний, только теперь мы говорим о посещении пары состояние–действие, а не состояния. Говорят, что пара состояние–действие посещалась в эпизоде, если посещалось состояние s и в нем выбиралось действие a . Метод МК всех посещений оценивает ценность пары состояние–действие как средний доход, полученный после всех ее посещений. Метод МК первого посещения усредняет доходы, полученные после того, как данное действие впервые было выбрано в данном состоянии. Эти методы, как и раньше, квадратично сходятся к истинным ожидаемым ценностям, когда количество посещений каждой пары состояние–действие стремится к бесконечности.

Единственная сложность состоит в том, что многие пары состояние–действие могут быть не посещены ни разу. Если π – детерминированная стратегия, то при следовании π будут наблюдаться доходы, приносимые только одним действием в каждом состоянии. Раз нет доходов для усреднения, то оценки Монте-Карло других действий не будут улучшаться с опытом. Это серьезная проблема, потому что цель обучения ценностям действий – помочь сделать выбор между действиями, доступными в каждом состоянии. Чтобы сравнить альтернативы, мы должны оценить ценности всех действий в каждом состоянии, а не только того, которому мы в данный момент отдаем предпочтение.

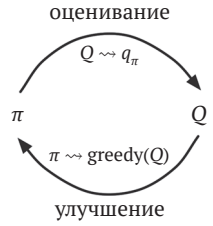
Это общая проблема *поощрения исследования*, которую мы обсуждали в контексте задачи о k -руком бандите в главе 2. Чтобы оценивание политики позволяло оценить ценности действий, мы должны гарантировать продолжение исследования. Один из способов добиться этого – сказать, что эпизоды *начинаются в паре состояние–действие* и что для любой пары вероятность быть выбранной в качестве начальной ненулевая. Это гарантирует, что все пары состояние–действие будут посещены бесконечное число раз, если количество эпизодов стремится к бесконечности. Данное предположение носит название *исследовательские старты* (exploring starts).

Предположение об исследовательских стартах иногда полезно, но, конечно, на него нельзя полагаться в общем случае, особенно при обучении на реальном взаимодействии с окружающей средой. В таком случае маловероятно, что начальные условия окажутся настолько удачными. Наиболее распространенный альтернативный подход к предположению о том, что все пары состояние–действие встречаются, – рассматривать только стохастические стратегии с ненулевой вероятностью выбора любого действия в каждом состоянии. В последующих разделах мы обсудим два важных варианта этого подхода. А пока сохраним предположение об исследовательских стартах и закончим описание полного метода управления Монте-Карло.

Упражнение 5.3. Нарисуйте диаграмму предшествующих состояний для оценивания q_π методом Монте-Карло. □

5.3. УПРАВЛЕНИЕ МЕТОДОМ МОНТЕ-КАРЛО

Теперь мы готовы рассмотреть, как оценивание методами Монте-Карло можно использовать в управлении, т. е. для аппроксимации оптимальных стратегий. Общая идея состоит в том, чтобы действовать по тому же образцу, что был описан в главе о ДП, т. е. применять обобщенную итерацию по стратегиям (ОИС). В ОИС одновременно строится приближенная стратегия и приближенная функция ценности. Функция ценности в цикле изменяется, чтобы быть ближе к функции ценности для текущей стратегии, а стратегия в цикле же улучшается относительно текущей функции ценности, как показано на диаграмме справа. Оба вида изменений в каком-то смысле антагонистичны, поскольку каждый отодвигает мишень, преследуемую другим, но совместно они приближают и стратегию, и функцию ценности к оптимальным.



Для начала рассмотрим вариант Монте-Карло классической итерации по стратегиям. В этом методе мы чередуем полные шаги оценивания и улучшения стратегии, начиная с произвольной стратегии π_0 и заканчивая оптимальной стратегией и оптимальной функцией ценности действий:

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} q_*,$$

где $\xrightarrow{E}$ обозначает полное оценивание стратегии, а $\xrightarrow{I}$ – полное улучшение стратегии. Оценивание стратегии производится точно так же, как описано в предыдущем разделе. Выполняется много эпизодов, при этом приближенная функция ценности действий асимптотически стремится к истинной. Ненадолго предположим, что мы действительно наблюдаем бесконечное число эпизодов и дополнительно что эпизоды сгенерированы в предположении об исследовательских стартах. В таком случае методы Монте-Карло будут вычислять каждую q_{π_k} точно для произвольной стратегии π_k .

Чтобы улучшить стратегию, мы делаем ее жадной относительно текущей функции ценности. В данном случае мы имеем функцию ценности действий, поэтому для построения жадной стратегии никакая модель не нужна. Для любой функции ценности действий q соответствующей жадной стратегией будет такая, которая для любого $s \in \mathcal{S}$ детерминированно выбирает действие с максимальной ценностью:

$$\pi(s) \doteq \operatorname{argmax}_a q(s, a). \quad (5.1)$$

Затем для улучшения стратегии можно построить π_{k+1} как жадную стратегию относительно q_{π_k} . После этого к π_k и π_{k+1} применима теорема об улучшении стратегии (раздел 4.2), потому что для любого $s \in \mathcal{S}$

$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \operatorname{argmax}_a q_{\pi_k}(s, a)) \\ &= \max_a q_{\pi_k}(s, a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \\ &\geq v_{\pi_k}(s). \end{aligned}$$

Как было сказано в предыдущей главе, эта теорема гарантирует, что каждая стратегия π_{k+1} равномерно лучше π_k или так же хороша, как π_k , т. е. обе стратегии оптимальны. Это, в свою очередь, гарантирует, что процесс в целом сходится к оптимальной стратегии и оптимальной функции ценности. Таким образом, методы Монте-Карло можно использовать для нахождения оптимальных стратегий, располагая только выборочными эпизодами и ничего не зная о динамике окружающей среды.

Выше мы сделали два неправдоподобных предположения, чтобы легко получить эту гарантию сходимости метода Монте-Карло. Одно заключалось в том, что в эпизодах имеются исследовательские старты, а второе – что для оценивания стратегии в нашем распоряжении имеется бесконечное число эпизодов. Чтобы получить практически применимый алгоритм, оба эти предположения необходимо снять. Рассмотрение первого мы ненадолго отложим.

А пока сосредоточимся на предположении о том, что оценивание политики производится на основании бесконечного числа эпизодов. Устранить его относительно просто. Действительно, та же самая проблема возникает и в классических методах ДП, например итеративном оценивании стратегии, которые также сходятся к истинной функции ценности только асимптотически. В обоих случаях решить эту проблему можно двумя способами. Первый – твердо придерживаться идеи аппроксимации $q_{\pi k}$ при каждом оценивании стратегии. Делаются измерения и предположения, необходимые для получения границ абсолютной величины и вероятности ошибки в оценках, а затем при оценивании каждой стратегии совершается столько шагов, чтобы гарантировать, что эти границы достаточно малы. Этот подход, вероятно, можно сделать вполне удовлетворительным в смысле гарантий аппроксимации с заданной точностью. Но, скорее всего, на практике необходимое количество эпизодов будет слишком велико даже для совсем не больших задач.

Существует другой подход, позволяющий избежать предположения о бесконечном числе эпизодов, номинально необходимом для оценивания стратегии, – мы отказываемся от попыток полностью оценить стратегию, перед тем как переходить к ее улучшению. На каждом шаге оценивания мы сдвигаем функцию ценности в направлении $q_{\pi k}$, но не ожидаем, что подойдем к ней очень близко. Этот подход мы использовали, когда только познакомились с идеей ОИС в разделе 4.6. Крайняя форма этой идеи – итерация по ценности, когда между каждыми двумя шагами улучшения стратегии выполняется только одна итерация итеративного оценивания стратегии. Еще более крайним является вариант итерации по ценности с обновлением на месте, когда мы чередуем шаги улучшения и оценивания для отдельных состояний.

Для итерации по стратегиям в методе Монте-Карло естественно чередовать оценивание и улучшение поэпизодно. После каждого эпизода наблюдаемые доходы используются для оценивания стратегии, а затем стратегия улучшается во всех состояниях, посещенных в этом эпизоде. Во врезке ниже приведен полный псевдокод алгоритма с учетом всех этих замечаний, который мы называем Монте-Карло ИС (метод Монте-Карло с исследовательскими стартами).

**Метод Монте-Карло ИС (с исследовательскими стартами)
для оценивания $\pi \approx \pi_*$**

Инициализация:

$\pi(s) \in \mathcal{A}(s)$ (произвольная) для всех $s \in \mathcal{S}$
 $Q(s, a) \in \mathbb{R}$ (произвольная) для всех $s \in \mathcal{S}, a \in \mathcal{A}(s)$
 $Returns(s, a) \leftarrow$ пустой список для всех $s \in \mathcal{S}, a \in \mathcal{A}(s)$

Повторять бесконечно (для каждого эпизода):

Выбрать $S_0 \in \mathcal{S}, A_0 \in \mathcal{A}(S_0)$ случайным образом, так что вероятность любой пары больше 0

Сгенерировать эпизод, начинающийся в S_0, A_0 и следующий π : $S_0, A_0, R_1,$

$\dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Повторять для каждого шага эпизода, $t = T - 1, T - 2, \dots, 0$:

$G \leftarrow G + \gamma R_{t+1}$

Если пара S_t, A_t не встречается в $S_0, A_0, S_1, A_1, \dots, S_{T-1}, A_{T-1}$

Добавить G в $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg \max_a Q(S_t, a)$

Упражнение 5.4. Псевдокод метода Монте-Карло ИС не эффективен, потому что для каждой пары состояние–действие он хранит список всех доходов и повторно вычисляет среднее. Было бы эффективнее использовать описанную в разделе 2.4 технику – хранить только среднее и количество элементов в списке (для каждой пары состояние–действие) и обновлять их инкрементно. Покажите, как для этого следует модифицировать псевдокод. $\square$

В методе Монте-Карло ИС все доходы для каждой пары состояние–действие накапливаются и усредняются независимо от того, какая стратегия действовала в момент их наблюдения. Легко видеть, что Монте-Карло ИС не может сойтись к неоптимальной стратегии. Если бы такое случилось, то функция ценности сходилась бы к функции ценности для данной стратегии, а это, в свою очередь, вынудило бы изменить стратегию. Устойчивость достигается, только когда и стратегия, и функция ценности оптимальны. Сходимость к этой оптимальной неподвижной точке кажется неизбежной, поскольку изменения функции ценности действий со временем становятся все меньше, тем не менее формально этот факт еще не доказан. На наш взгляд, это один из самых фундаментальных открытых теоретических вопросов в обучении с подкреплением (частичное решение см. в работе Tsitsiklis, 2002).

Пример 5.3: решение задачи об игре в блэкджек. К этой игре метод Монте-Карло ИС применяется прямолинейно. Поскольку эпизодами являются все смоделированные партии, легко организовать исследовательские старты, включающие все возможности. В данном случае просто выбираются карты сдающего, сумма очков игрока и наличие у игрока играющего туза – все случайным образом с одинаковой вероятностью. В качестве начальной стратегии мы берем стратегию, вы-

численную в предыдущем примере игры в блэкджек, при которой игрок останавливается, только набрав 20 или 21. Начальная функция ценности действий может быть равна нулю для всех пар состояние–действие. На рис. 5.2 показана оптимальная стратегия игры, найденная методом Монте-Карло ИС. Она отличается от «базовой» стратегии, описанной в работе Thorp (1966), только наличием впадины слева для играющего туза, которой в стратегии Торпа нет. Нам непонятна причина этого расхождения, но мы уверены, что на нашем рисунке показана действительно оптимальная стратегия для описанного варианта игры в блэкджек.

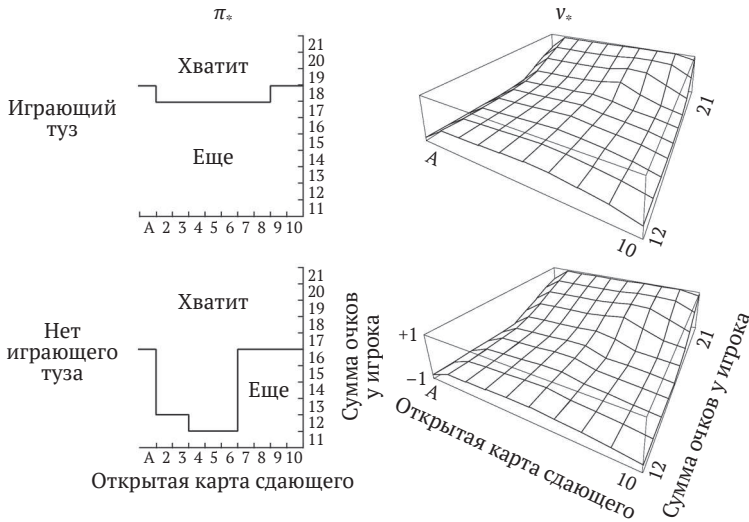


Рис. 5.2 ❖ Оптимальные стратегия и функция ценности состояний для игры в блэкджек, найденные методом Монте-Карло ИС. Эта функция ценности состояний вычислена по функции ценности действий, найденной методом Монте-Карло ИС

5.4. УПРАВЛЕНИЕ МЕТОДОМ МОНТЕ-КАРЛО БЕЗ ИССЛЕДОВАТЕЛЬСКИХ СТАРТОВ

Как избежать неправдоподобного предположения об исследовательских стартах? Единственный общий способ гарантировать, что каждое действие выбирается бесконечно часто, – заставить агента и дальше выбирать их. Таких подходов два: *методы с единой стратегией* (on-policy method) и *методы с разделенной стратегией* (off-policy method). Методы с единой стратегией пытаются оценить или улучшить стратегию, которая используется для принятия решений, а методы с разделенной стратегией – оценить или улучшить стратегию, отличную от той, что использовалась для генерации данных. Метод Монте-Карло ИС, разработанный выше, – пример метода с единой стратегией. В этом разделе мы покажем, как спроектировать метод управления Монте-Карло с единой стратегией, в котором не используется нереалистичное предположение об исследовательских стартах. Методы с разделенной стратегией рассматриваются в следующем разделе.

В методах управления с единой стратегией обычно применяется *мягкая* стратегия, т. е. $\pi(a|s) > 0$ для всех $s \in \mathcal{S}$ и всех $a \in \mathcal{A}(s)$, но постепенно приближается к детерминированной оптимальной стратегии. Многие методы, рассмотренные в главе 2, предлагают соответствующие механизмы. В представленном в этом разделе методе с единой стратегией применяются ε -жадные стратегии, т. е. большую часть времени они выбирают действие с максимальной оценкой ценности, но с вероятностью ε все же выбирают случайное действие. Таким образом, у любого нежадного действия есть ненулевая вероятность $\varepsilon/|\mathcal{A}(s)|$ быть выбранным, а вся остальная масса вероятности, $1 - \varepsilon + \varepsilon/|\mathcal{A}(s)|$, распределяется между жадными действиями. ε -жадные стратегии – пример ε -мягких стратегий, определенных как стратегии, для которых $\pi(a|s) \geq \varepsilon/|\mathcal{A}(s)|$ для всех состояний и действий при некотором $\varepsilon > 0$. Среди всех ε -мягких стратегий ε -жадные в некотором смысле наиболее близки к жадным.

Общая идея методов управления Монте-Карло с единой стратегией – по-прежнему ОИС. Как и в методе Монте-Карло ИС, мы применяем методы МК первого посещения для оценивания функции ценности действий для текущей стратегии. Однако без предположения об исследовательских стартах мы не можем просто улучшить стратегию, сделав ее жадной относительно текущей функции ценности, потому что это остановило бы дальнейшее исследование нежадных действий. По счастью, ОИС не требует, чтобы стратегия совпадала с жадной, нужно лишь, чтобы она постепенно приближалась к таковой. В нашем методе с единой стратегией мы будем двигаться в сторону всего лишь ε -жадной стратегии. Для любой ε -мягкой стратегии π гарантируется, что любая ε -жадная стратегия относительно q_π будет лучше или равна π . Полностью алгоритм приведен во врезке ниже.

Метод управления МК первого посещения (для ε -мягких стратегий) для оценивания $\pi \approx \pi_*$

Параметр алгоритма: небольшое $\varepsilon > 0$

Инициализация:

$\pi \leftarrow$ произвольная ε -мягкая стратегия

$Q(s, a) \in \mathbb{R}$ (произвольная) для всех $s \in \mathcal{S}, a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$ пустой список для всех $s \in \mathcal{S}, a \in \mathcal{A}(s)$

Повторять бесконечно (для каждого эпизода):

Сгенерировать эпизод, следующий π : $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Повторять для каждого шага эпизода, $t = T-1, T-2, \dots, 0$:

$G \leftarrow G + \gamma R_{t+1}$

Если пара S_t, A_t не встречается в $S_0, A_0, S_1, A_1, \dots, S_{T-1}, A_{T-1}$

Добавить G в $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg \max_a Q(S_t, a)$ (неоднозначности разрешаются произвольно)

Для всех $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{если } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{если } a \neq A^* \end{cases}$$

То, что любая ε -жадная стратегия относительно q_π является улучшением любой ε -мягкой стратегии π , гарантируется теоремой об улучшении стратегии. Пусть π' – ε -жадная стратегия. Условия теоремы об улучшении стратегии выполняются, потому что для любого $s \in \mathcal{S}$:

$$\begin{aligned} q_\pi(s, \pi'(s)) &= \sum_a \pi'(a|s) q_\pi(s, a) \\ &= \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) + (1 - \varepsilon) \max_a q_\pi(s, a) \\ &\geq \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) + (1 - \varepsilon) \sum_a \frac{\pi(a|s) - \frac{\varepsilon}{|\mathcal{A}(s)|}}{1 - \varepsilon} q_\pi(s, a) \end{aligned} \quad (5.2)$$

(сумма является взвешенным средним с неотрицательными весами, в сумме равными 1, а потому должна быть меньше или равна любому взвешиваемому слагаемому)

$$\begin{aligned} &= \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) - \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) + \sum_a \pi(a|s) q_\pi(s, a) \\ &= v_\pi(s). \end{aligned}$$

Таким образом, по теореме об улучшении стратегии $\pi' \geq \pi$ (т. е. $v_{\pi'}(s) \geq v_\pi$ для всех $s \in \mathcal{S}$). Теперь мы докажем, что равенство возможно, только когда π' и π оптимальны на множестве ε -мягких стратегий, т. е. когда они лучше или равны всем остальным ε -мягким стратегиям.

Рассмотрим новую окружающую среду, которая отличается от исходной только требованием, что стратегии должны быть ε -мягко «перемещены внутрь» среды. Новая среда обладает теми же множествами состояний и действий, что исходная, и ведет себя следующим образом. Если в состоянии s предпринимается действие a , то с вероятностью $1 - \varepsilon$ новая среда ведет себя точно так же, как старая. А с вероятностью ε она повторно выбирает действие случайным образом с одинаковыми вероятностями, а затем ведет себя, как старая среда с этим новым случайным действием. Лучшее, что можно сделать с общими стратегиями в этой новой среде, – то же самое, что можно было сделать в исходной среде с ε -мягкими стратегиями. Обозначим $\tilde{v}_*$ и $\tilde{q}_*$ оптимальные функции ценности для новой среды. Тогда стратегия π оптимальна среди ε -мягких стратегий тогда и только тогда, когда $v_\pi = \tilde{v}_*$. Из определения $\tilde{v}_*$ мы знаем, что это единственное решение уравнения

$$\begin{aligned} \tilde{v}_*(s) &= (1 - \varepsilon) \max_a \tilde{q}_*(s, a) + \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a \tilde{q}_*(s, a) \\ &= (1 - \varepsilon) \max_{s', r} \sum_a p(s', r|s, a) [r + \gamma \tilde{v}_*(s')] \\ &\quad + \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a \sum_{s', r} p(s', r|s, a) [r + \gamma \tilde{v}_*(s')]. \end{aligned} \quad *54*$$

Если имеет место равенство и ε -мягкая стратегия π больше не улучшается, то мы также знаем из (5.2), что

$$\begin{aligned}
v_{\pi}(s) &= (1 - \varepsilon) \max_a q_{\pi}(s, a) + \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_{\pi}(s, a) \\
&= (1 - \varepsilon) \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_{\pi}(s')] \\
&\quad + \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_{\pi}(s')].
\end{aligned}$$

Однако это уравнение отличается от предыдущего только заменой $\tilde{v}_*$ на v_{π} . Поскольку $\tilde{v}_*$ – единственное решение, то должно быть $v_{\pi} = \tilde{v}_*$.

По сути дела, на последних нескольких страницах мы показали, что итерация по стратегиям работает для ε -мягких стратегий. Применяя естественное понятие жадной стратегии к ε -мягким стратегиям, можно быть уверенным в улучшении на каждом шаге, только мы находим лучшую из ε -мягких стратегий. Этот анализ не зависит от того, как именно определяются функции ценности действий на каждом этапе, но предполагается, что они вычисляются точно. Таким образом, мы пришли примерно туда же, куда и в предыдущем разделе. Да, мы можем найти только лучшую среди ε -мягких стратегий, но зато избавились от предположения об исследовательских стартах.

5.5. ПРЕДСКАЗАНИЕ С РАЗДЕЛЕННОЙ СТРАТЕГИЕЙ ПОСРЕДСТВОМ ВЫБОРКИ ПО ЗНАЧИМОСТИ

Все методы управления обучением сталкиваются с дилеммой: требуется обучить ценности действий при условии последующего *оптимального* поведения, но чтобы исследовать все действия, приходится вести себя неоптимально (чтобы *найти* оптимальные действия). Как можно обучиться оптимальной стратегии, следуя исследовательской стратегии? Методы с единой стратегией, рассмотренные в предыдущем разделе, фактически представляют собой компромисс – они обучаются ценности действий не для оптимальной стратегии, а для почти оптимальной, которая все еще занимается исследованием. Более прямолинейный подход – использовать две стратегии: ту, которой нужно обучиться и которая станет оптимальной, и другую – носящую в большей степени исследовательский характер и служащую для генерации поведения. Стратегия, которой следует обучиться, называется *целевой*, а стратегия, генерирующая поведения, – *поведенческой*. В таком случае мы говорим, что обучение ведется на данных, «находящихся вне» целевой стратегии, а весь процесс называется *обучением с разделенной стратегией* (off-policy learning).

Далее в этой книге мы рассматриваем методы как с единой, так и с разделенной стратегией. Методы с единой стратегией, вообще говоря, проще и рассматриваются первыми. Для изучения методов с разделенной стратегией необходимы дополнительные понятия и обозначения, а поскольку данные связаны с другой стратегией, такие методы обычно обладают большей дисперсией и сходятся медленнее. С другой стороны, методы с разделенной стратегией – более мощные и общие. Они включают методы с единой стратегией в качестве частного случая,

когда целевая и поведенческая стратегии совпадают. Методы с разделенной стратегией также имеют ряд дополнительных применений. Например, они часто используются для обучения на данных, сгенерированных традиционным несамобучающимся контроллером или экспертом-человеком. Обучение с разделенной стратегией иногда рассматривается как ключ к обучению многошаговых прогнозистических моделей динамики мира (см. раздел 17.2; Sutton, 2009; Sutton et al., 2011).

В этом разделе мы начнем изучение методов с разделенной стратегией с рассмотрения задачи *предсказания*, когда целевая и поведенческая стратегии фиксированы. Предположим, что мы хотим оценить v_π или q_π , имея лишь эпизоды, следующие какой-то другой стратегии $b \neq \pi$. В таком случае π – целевая стратегия, b – поведенческая стратегия, и обе они считаются фиксированными и известными.

Чтобы использовать эпизоды из b для оценки ценностей для π , мы потребуем, чтобы всякое действие, выбранное при следовании π , выбиралось, хотя бы иногда, также и при следовании b . То есть потребуем, чтобы из $\pi(a|s) > 0$ следовало $b(a|s) > 0$. Это называется предположением о *покрытии*. Из него вытекает, что b должна быть стохастической в тех состояниях, где она не совпадает с π . С другой стороны, целевая стратегия π может быть детерминированной, и в действительности именно этот случай представляет особый интерес в приложениях к управлению. В задачах управления целевая стратегия обычно является детерминированной и жадной относительно текущей оценки функции ценности действий. Эта стратегия становится детерминированной оптимальной стратегией, тогда как поведенческая стратегия остается стохастической (и в большей мере ориентирована на исследование), например ε -жадной. Но в этом разделе мы рассматриваем задачу предсказания, в которой π задана и не изменяется.

Почти все методы с разделенной стратегией пользуются выборкой по значимости – общей техникой оценивания ожидаемых значений из одного распределения при наличии выборки из другого. Мы применим выборку по значимости к обучению с разделенной стратегией, назначив каждому доходу вес, равный относительной вероятности его траектории при следовании целевой и поведенческой стратегиям. Этот вес называется *коэффициентом выборки по значимости*. Если дано начальное состояние S_t , то вероятность последующей траектории состояний и действий $A_t, S_{t+1}, A_{t+1}, \dots, S_T$ при следовании произвольной стратегии π равна

$$\begin{aligned} \Pr\{A_t, S_{t+1}, A_{t+1}, \dots, S_T | S_t, A_{t:T-1} \sim \pi\} \\ = \pi(A_t | S_t) p(S_{t+1} | S_t, A_t) \pi(A_{t+1} | S_{t+1}) \cdots p(S_T | S_{T-1}, A_{T-1}) \\ = \prod_{k=t}^{T-1} \pi(A_k | S_k) p(S_{k+1} | S_k, A_k), \end{aligned}$$

где p – функция вероятности переходов состояний, определенная формулой (3.4). Таким образом, относительная вероятность траектории при следовании целевой и поведенческой стратегиям (коэффициент выборки по значимости) равна

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k | S_k) p(S_{k+1} | S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k | S_k) p(S_{k+1} | S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k | S_k)}{b(A_k | S_k)}. \quad (5.3)$$

Хотя вероятности траекторий зависят от вероятностей переходов в МППР, которые в общем случае неизвестны, они входят и в числитель, и в знаменатель, поэтому сокращаются. В результате коэффициент выборки по значимости зависит только от обеих стратегий и последовательности, но не от МППР.

Напомним, что мы хотим оценить ожидаемые доходы (ценности) при следовании целевой стратегии, но имеем лишь доходы G_t при следовании поведенческой стратегии. У этих доходов не то математическое ожидание $\mathbb{E}[G_t | S_t = s] = v_b(s)$, поэтому их нельзя усреднить для получения v_π . Именно здесь в игру вступает выборка по значимости. Коэффициент $\rho_{t:T-1}$ преобразует доходы так, что получается нужное математическое ожидание:

$$\mathbb{E}[\rho_{t:T-1} G_t | S_t = s] = v_\pi(s). \quad (5.4)$$

Теперь мы готовы сформулировать алгоритм Монте-Карло, который усредняет доходы, полученные в пакете наблюдавшихся эпизодов, следующих стратегии b , с целью оценить $v_\pi(s)$. Здесь удобно нумеровать временные шаги, так чтобы нумерация не прерывалась при пересечении границ эпизодов. То есть если первый эпизод пакета заканчивается в заключительном состоянии в момент 100, то следующий эпизод начинается в момент $t = 101$. Это позволяет использовать номера временных шагов для ссылки на конкретные шаги в конкретных эпизодах. В частности, мы можем определить множество всех временных шагов $\mathcal{T}(s)$, на которых посещалось состояние s . Это определение действует только для метода всех посещений; для метода первого посещения $\mathcal{T}(s)$ включало бы только временные шаги, на которых состояние s впервые посещалось в пределах одного эпизода. Введем также обозначение $T(t)$ для момента первого завершения после t , и пусть G_t – доход, полученный после t и до $T(t)$ включительно. Тогда $\{G_t\}_{t \in \mathcal{T}(s)}$ – доходы, относящиеся к состоянию s , а $\{\rho_{t:T(t)-1}\}_{t \in \mathcal{T}(s)}$ – соответствующие коэффициенты выборки по значимости. Чтобы оценить $v_\pi(s)$, мы просто умножаем доходы на коэффициенты и усредняем результаты:

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}. \quad (5.5)$$

Когда выборка по значимости производится как простое среднее (как описано выше), она называется *обыкновенной выборкой по значимости*.

Важная альтернатива – *взвешенная выборка по значимости* – определяется с помощью взвешенного среднего:

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}, \quad (5.6)$$

или равна 0, если знаменатель равен нулю. Чтобы разобраться в этих двух разновидностях выборки по значимости, рассмотрим оценки, полученные их методами первого посещения после наблюдения одного дохода при старте из состояния s . В средневзвешенной оценке коэффициент $\rho_{t:T(t)-1}$ для одного дохода присутствует в числителе и знаменателе и, следовательно, сокращается, поэтому оценка равна наблюдаемому доходу и не зависит от коэффициента (в предположении, что коэффициент не равен нулю). Учитывая, что этот доход – единственный наблю-

давшийся, это разумная оценка, но ее математическое ожидание равно $v_b(s)$, а не $v_\pi(s)$, т. е. она является смещенной в статистическом смысле. Напротив, в версии первого посещения оценки обыкновенной выборки по значимости (5.5) математическое ожидание всегда равно $v_\pi(s)$ (оценка несмещенная), но она может быть экстремальной. Предположим, что коэффициент равен 10, т. е. наблюдаемая траектория в десять раз более вероятна при следовании целевой стратегии, чем поведенческой. В таком случае оценка обыкновенной выборки по значимости была бы в *десять раз* больше наблюдаемого дохода. То есть она была бы весьма далека от наблюдаемого дохода, хотя траектория эпизода считается очень представительной для целевой стратегии.

Формально различие между методами первого посещения в двух видах выборки по значимости выражается в терминах их смещения и дисперсии. Обыкновенная выборка по значимости является несмещенной, а взвешенная – смещенной (хотя смещение асимптотически стремится к нулю). С другой стороны, дисперсия обыкновенной выборки по значимости не ограничена, поскольку дисперсия коэффициентов может быть неограниченной, тогда как во взвешенной оценке наибольший вес любого дохода равен 1. На самом деле в предположении ограниченности доходов дисперсия оценки взвешенной выборки по значимости стремится к нулю, даже если дисперсия самих коэффициентов бесконечна (Precup, Sutton, and Dasgupta 2001). На практике взвешенная оценка обычно имеет существенно меньшую дисперсию, и настоятельно рекомендуется выбирать именно ее. Тем не менее мы не станем полностью отказываться от обыкновенной выборки по значимости, поскольку ее проще обобщить на приближенные методы с использованием аппроксимации функций, изучаемые во второй части книги.

Методы первого посещения для обыкновенной и взвешенной выборок по значимости смещенные, но опять-таки смещение асимптотически стремится к нулю по мере увеличения объема выборки. На практике часто предпочитают методы всех посещений, потому что при этом отпадает необходимость следить за тем, какие состояния уже посещались, и потому что они гораздо легче обобщаются на приближенные методы. Полностью алгоритм МК всех посещений с разделенной стратегией и с применением взвешенной выборки по значимости приведен в следующем разделе на стр. 143.

Упражнение 5.5. Рассмотрим МППР с одним незаключительным состоянием и одним действием, которое ведет назад в незаключительное состояние с вероятностью p и в заключительное состояние с вероятностью $1 - p$. Пусть вознаграждение за любой переход равно $+1$ и $\gamma = 1$. Предположим, что наблюдается один эпизод, продолжающийся 10 шагов, с доходом 10. Каковы оценки ценности незаключительного состояния в случае первого посещения и всех посещений? □

Пример 5.4: Оценка с разделенной стратегией ценности состояния при игре в блэкджек. Мы применили обыкновенный и взвешенный методы выборки по значимости для оценивания ценности одного состояния блэкджека по данным, доступным в методе с разделенной стратегией. Напомним, что одно из преимуществ методов Монте-Карло состоит в том, что их можно использовать для оценивания одного состояния, не вычисляя оценки всех остальных состояний. В этом примере мы оценили состояние, в котором сдающий открыл двойку, игрок набрал

13 очков и имеет играющего туза (т. е. на руках игрока туз и двойка, что эквивалентно трем тузам). Данные были сгенерированы, начиная в этом состоянии, после чего игрок случайным образом говорил «еще» или «хватит» с одинаковой вероятностью (поведенческая стратегия). Целевой стратегией была остановка только при наборе 20 или 21 очка, как в примере 5.1. Ценность этого состояния при следовании целевой стратегии приблизительно равна -0.27726 (она была вычислена в результате отдельной генерации ста миллионов эпизодов с использованием целевой стратегии и последующего усреднения доходов). Оба метода с разделенной стратегией хорошо аппроксимировали эту ценность после 1000 эпизодов со случайной стратегией. Чтобы убедиться в надежности этой аппроксимации, мы выполнили 100 независимых прогонов, каждый из которых начинался с оценки 0 и обучался на 10 000 эпизодов. На рис. 5.3 показаны получившиеся кривые обучения – среднеквадратичная ошибка оценок каждого метода, усредненная по 100 прогонам, в виде функции от количества эпизодов. Ошибка стремится к 0 в обоих алгоритмах, но в методе взвешенной выборки по значимости ошибка вначале гораздо меньше, что типично для практических применений.

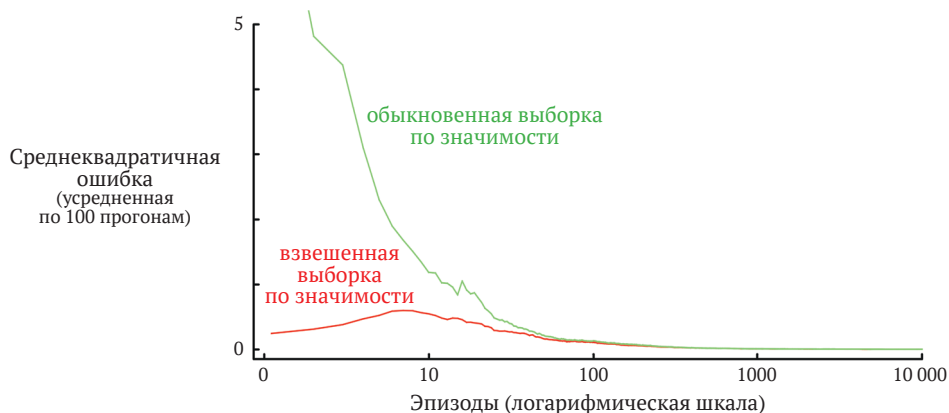


Рис. 5.3 ❖ Взвешенная выборка по значимости дает меньшую ошибку оценки ценности одного состояния в блэкджеке по эпизодам с разделенной стратегией ■

Пример 5.5: бесконечная дисперсия. Оценки обыкновенной выборки по значимости, как правило, имеют бесконечную дисперсию (а потому неудовлетворительные свойства сходимости), если дисперсия масштабированных доходов бесконечна – а это легко может случиться в обучении с разделенной стратегией, когда траектории содержат циклы. На рис. 5.4 показан простой пример. Имеется только одно незаключительное состояние s и два действия: *right* и *left*. Действие *right* вызывает детерминированный переход в заключительное состояние, а действие *left* – переход назад в s с вероятностью 0.9 или переход в заключительное состояние с вероятностью 0.1. Вознаграждение составляет +1 за последний переход и 0 в остальных случаях. Рассмотрим целевую стратегию, которая всегда выбирает действие *left*. Все эпизоды при следовании этой стратегии состоят из какого-то количества (чаще всего нуля) переходов назад в s , за которыми следует завершение с вознаграждением и доходом +1. Таким образом, ценность s при такой це-

левой стратегии равна 1 ($\gamma = 1$). Предположим, что мы оцениваем эту ценность по данным разделенной стратегии, в которой поведенческая стратегия выбирает right и left с одинаковой вероятностью.

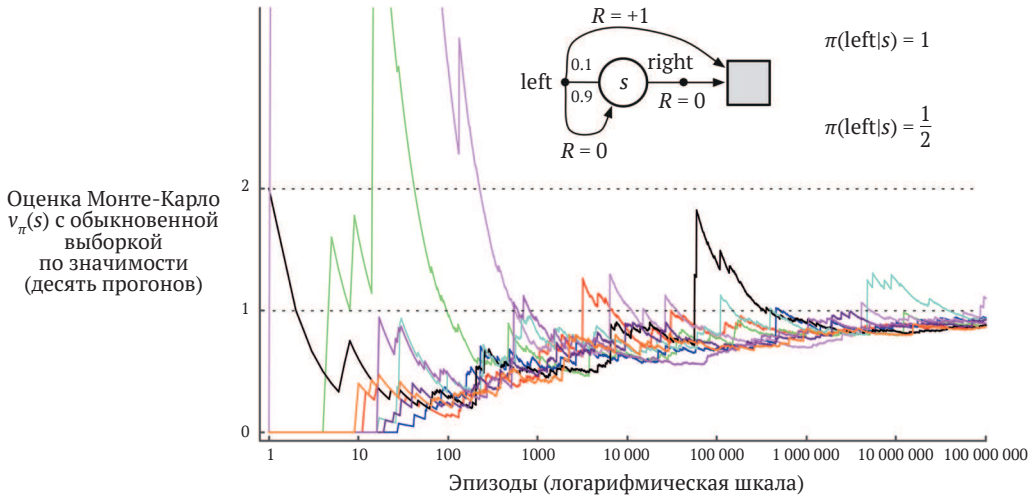


Рис. 5.4 ❖ Обыкновенная выборка по значимости порождает на удивление неустойчивые оценки для МППР с одним состоянием, показанным в верхней части (пример 5.5). В данном случае правильная оценка равна 1 ($\gamma = 1$), но, хотя это действительно математическое ожидание выборочного дохода (после выборки по значимости), дисперсия выборки бесконечна и оценки не сходятся к этому значению. Эти результаты имеют место для МК первого посещения с разделенной стратегией

В нижней части рис. 5.4 показано десять независимых прогонов алгоритма МК первого посещения с обыкновенной выборкой по значимости. Даже после миллионов эпизодов оценка не сходится к правильному значению 1. С другой стороны, алгоритм взвешенной выборки по значимости дал бы оценку, в точности равную 1 после первого эпизода, закончившегося действием left. Все доходы, не равные 1 (т. е. в эпизодах, заканчивающихся действием right), были бы не согласованы с целевой стратегией, поэтому для них коэффициент $\rho_{t:T(t)-1}$ был бы равен 0 и не давал бы никакого вклада ни в числитель, ни в знаменатель (5.6). Алгоритм взвешенной выборки по значимости порождает взвешенное среднее только для доходов, согласованных с целевой стратегией, а все такие доходы должны быть в точности равны 1.

Мы можем убедиться в том, что дисперсия доходов, масштабированных на выборку по значимости, в этом примере бесконечна, выполнив простое вычисление. Дисперсия любой случайной величины X равна математическому ожиданию отклонения от среднего $\bar{X}$, что можно записать в виде:

$$\text{Var}[X] \doteq \mathbb{E}[(X - \bar{X})^2] = \mathbb{E}[X^2 - 2X\bar{X} + \bar{X}^2] = \mathbb{E}[X^2] - \bar{X}^2.$$

Следовательно, если среднее конечно, как в нашем случае, то дисперсия бесконечна тогда и только тогда, когда математическое ожидание квадрата случайной величины бесконечно. Поэтому нужно только показать, что математическое ожидание квадрата дохода, масштабированного на выборку по значимости, бесконечно:

$$\mathbb{E}_b \left[\left(\prod_{t=0}^{T-1} \frac{\pi(A_t | S_t)}{b(A_t | S_t)} G_0 \right)^2 \right].$$

Чтобы вычислить это математическое ожидание, выделим несколько случаев в зависимости от длины эпизода и способа его завершения. Сначала заметим, что для любого эпизода, заканчивающегося действием `right`, коэффициент выборки по значимости равен нулю, потому что при целевой стратегии это действие никогда не предпринимается; таким образом, эти эпизоды ничего не вносят в математическое ожидание (выражение в скобках равно нулю), и их можно игнорировать. Нам нужно рассмотреть только эпизоды, в которых имеется обратный переход в незаключительное состояние, за которым следует действие `left`, приводящее к завершению. Во всех этих эпизодах доход равен 1, поэтому коэффициент G_0 можно проигнорировать. Чтобы получить искомое математическое ожидание, нужно для каждой длины эпизода умножить вероятность такого эпизода на квадрат его коэффициента выборки по значимости и просуммировать:

$$\begin{aligned} &= \frac{1}{2} \cdot 0.1 \left(\frac{1}{0.5} \right)^2 && \text{(эпизод длины 1)} \\ &+ \frac{1}{2} \cdot 0.9 \cdot \frac{1}{2} \cdot 0.1 \left(\frac{1}{0.5} \frac{1}{0.5} \right)^2 && \text{(эпизод длины 2)} \\ &+ \frac{1}{2} \cdot 0.9 \cdot \frac{1}{2} \cdot 0.9 \cdot \frac{1}{2} \cdot 0.1 \left(\frac{1}{0.5} \frac{1}{0.5} \frac{1}{0.5} \right)^2 && \text{(эпизод длины 3)} \\ &+ \dots \\ &= 0.1 \sum_{k=0}^{\infty} 0.9^k \cdot 2^k \cdot 2 = 0.2 \sum_{k=0}^{\infty} 1.8^k = \infty. \end{aligned}$$

Упражнение 5.6. Как выглядит уравнение, аналогичное (5.6), если вместо ценностей состояний $V(s)$ использовать ценности действий $Q(s, a)$, опять-таки при известных доходах, сгенерированных с помощью стратегии b ? □

Упражнение 5.7. На кривых обучения типа показанных на рис. 5.3 ошибка обычно убывает по мере обучения, как и случилось для метода обыкновенной выборки по значимости. Но для метода взвешенной выборки по значимости ошибка сначала возрастает, а потом убывает. Как вы думаете, почему так происходит? □

Упражнение 5.8. При получении результатов в примере 5.5, показанных на рис. 5.4, использовался метод МК первого посещения. Допустим, что для той же задачи был применен метод МК всех посещений. Будет ли в таком случае дисперсия оценки по-прежнему бесконечной? Объясните свой ответ. □

5.6. ИНКРЕМЕНТНАЯ РЕАЛИЗАЦИЯ

Методы предсказания Монте-Карло можно реализовать инкрементно, т. е. поэтапно, применяя обобщение техники, описанной в главе 2 (раздел 2.4). Если в главе 2 мы усредняли *вознаграждения*, то в методах Монте-Карло усредняются *доходы*. Если не считать этого отличия, то методы, использованные в главе 2, вполне пригодны для методов Монте-Карло с *единой стратегией*. Что касается методов Монте-Карло с *разделенной стратегией*, то необходимо по отдельности рассматривать методы, в которых применяется *обыкновенная* и *взвешенная* выборка по значимости.

В случае обыкновенной выборки по значимости доходы умножаются на коэффициент выборки по значимости $\rho_{t:T(t)-1}$ (формула 5.3), а затем просто усредняются, как в (5.5). Здесь мы снова можем использовать инкрементные методы из главы 2, только вместо вознаграждений нужно брать масштабированные доходы. Остается случай методов с разделенной стратегией с применением *взвешенной* выборки по значимости. Здесь нам понадобится взвешенное среднее доходов и несколько отличающийся инкрементный алгоритм.

Пусть имеется последовательность доходов $G_1, G_2, \dots, G_{n-1}$, полученных при старте из одного и того же состояния, и каждому назначен случайный вес W_i (на пример, $W_i = \rho_{t_i:T(t_i)-1}$). Мы хотим вычислить оценку

$$V_n \doteq \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}, \quad n \geq 2, \quad (5.7)$$

и поддерживать ее актуальность при получении дополнительного дохода G_n . Для вычисления V_n нам необходимо еще хранить для каждого состояния накопительную сумму C_n весов, при условии что известны первые n доходов. Формула обновления V_n имеет вид:

$$V_{n+1} \doteq V_n + \frac{W_n}{C_n} [G_n - V_n], \quad n \geq 1. \quad (5.8)$$

При этом

$$C_{n+1} \doteq C_n + W_{n+1},$$

где $C_0 \doteq 0$ (а V_1 произвольно, так что задавать его необязательно). Во врезке ниже полностью приведен поэтапный инкрементный алгоритм оценивания стратегии методом Монте-Карло. Формально алгоритм соответствует случаю разделенной стратегии с применением взвешенной выборки по значимости, но фактически он годится и для случая единой стратегии, нужно только, чтобы целевая и поведенческая стратегии совпадали (в этом случае ($\pi = b$), а W всегда равен 1). Аппроксимация Q сходится к q_π (для всех встретившихся пар состояние–действие), если действия выбираются, следуя потенциально другой стратегии b .

Упражнение 5.9. Модифицируйте алгоритм для оценивания стратегии методом МК первого посещения (раздел 5.1), воспользовавшись инкрементной реализацией выборочного среднего, описанной в разделе 2.4. $\square$

Упражнение 5.10. Выведите правило обновления взвешенного среднего (5.8) из (5.7). Рассуждайте так же, как при выводе правила для обычного среднего (2.3). □

Предсказание методом МК с разделенной стратегией (оценивание стратегии) для вычисления оценки $Q \approx q_\pi$

Вход: произвольная целевая стратегия π

Инициализация: для всех $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$Q(s, a) \in \mathbb{R}$ (произвольная)

$C(s, a) \leftarrow 0$

Повторять бесконечно (для каждого эпизода):

$b \leftarrow$ произвольная стратегия с покрытием π

Сгенерировать эпизод, следующий b : $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

$W \leftarrow 1$

Повторять для каждого шага эпизода, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$

$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + (W/C(S_t, A_t))[G - Q(S_t, A_t)]$

$W \leftarrow W[\pi(A_t | S_t)/b(A_t | S_t)]$

Если $W = 0$, то выйти из внутреннего цикла

5.7. УПРАВЛЕНИЕ МЕТОДОМ МОНТЕ-КАРЛО С РАЗДЕЛЕННОЙ СТРАТЕГИЕЙ

Теперь мы готовы представить пример второго класса методов обучения управлению: методов с разделенной стратегией. Напомним, что отличительной чертой методов с единой стратегией является тот факт, что они оценивают ценность той же стратегии, которая используется для управления. А в методах с разделенной стратегией эти две функции разделены. Та стратегия, которая порождает поведение и называется *поведенческой*, может быть никак не связана со стратегией, которая оценивается и улучшается (она называется *целевой*). Преимущество такого разделения в том, что целевая стратегия может быть детерминированной (например, жадной), а поведенческая при этом продолжает пробовать все возможные действия.

В методах управления Монте-Карло с разделенной стратегией используется одна из техник, описанных в двух предыдущих разделах. Они следуют поведенческой стратегии, но при этом обучаются целевой и улучшают ее. Для применения такой техники требуется, чтобы поведенческая стратегия с ненулевой вероятностью выбирала все действия, которые может выбрать целевая стратегия (покрытие). Чтобы исследовать все возможности, мы требуем, чтобы поведенческая стратегия была мягкой (т. е. выбирала все действия во всех состояниях с ненулевой вероятностью).

Во врезке ниже показан метод управления Монте-Карло с разделенной стратегией, который оценивает π_* и q_* , основываясь на ОИС и взвешенной выборке по значимости. Целевой стратегией $\pi \approx \pi_*$ является жадная стратегия относительно Q , оценки q_π . Поведенческая стратегия b может быть любой, но чтобы гарантировать сходимость π к оптимальной стратегии, для каждой пары состояние–действие необходимо получить бесконечное число доходов. Такую гарантию можно дать, если выбрать b ε -мягкой. Стратегия π сходится к оптимальной во всех встретившихся состояниях, несмотря на то что действия выбираются, следуя другой мягкой стратегии b , которая может изменяться между эпизодами или даже внутри одного эпизода.

**Управление методом МК с разделенной стратегией
для вычисления оценки $\pi \approx \pi_*$**

Инициализация: для всех $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$Q(s, a) \in \mathbb{R}$ (произвольная)

$C(s, a) \leftarrow 0$

$\pi(s) \leftarrow \operatorname{argmax}_a Q(s, a)$

Повторять бесконечно (для каждого эпизода):

$b \leftarrow$ произвольная мягкая стратегия

Сгенерировать эпизод, следующий b : $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

$W \leftarrow 1$

Повторять для каждого шага эпизода, $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$

$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + (W/C(S_t, A_t))[G - Q(S_t, A_t)]$

$\pi(S_t) \leftarrow \operatorname{argmax}_a Q(s, a)$ (неоднозначности разрешаются произвольно)

Если $A_t \neq \pi(S_t)$, то выйти из внутреннего цикла

$W \leftarrow W[1/b(A_t|S_t)]$

Потенциальная проблема заключается в том, что этот метод обучается только на хвостах эпизодов, когда все оставшиеся действия в эпизоде жадные. Если нежадные действия встречаются часто, то обучение будет происходить медленно, особенно для состояний, встречающихся в начале длинных эпизодов. Теоретически замедление обучения может быть очень сильным. Опыт применения методов Монте-Карло с разделенной стратегией недостаточен, чтобы судить о серьезности проблемы. Если она действительно серьезна, то, наверное, наиболее перспективный способ ее решения – включить обучение на основе временных различий, алгоритмическую идею, которой посвящена следующая глава. С другой стороны, если γ меньше 1, то может здорово помочь идея, описанная в следующем разделе.

Упражнение 5.11. В приведенном выше алгоритме управления методом МК с разделенной стратегией вы, возможно, ожидали, что в обновлении W будет участвовать коэффициент выборки по значимости $\pi(A_t|S_t)/b(A_t|S_t)$, а на самом деле мы видим $1/b(A_t|S_t)$. Почему же, несмотря на это, алгоритм корректен? $\square$

Упражнение 5.12: кольцевые гонки (требуется программирование). Рассмотрим управление гоночным автомобилем на повороте, показанном на рис. 5.5. Вы хотите пройти его как можно быстрее, но не настолько быстро, чтобы вылететь с трассы. В нашей упрощенной модели кольцевых гонок автомобиль может находиться в одной из дискретных ячеек сетки. Скорость также дискретна и равна количеству ячеек, на которые машина переместилась по горизонтали и по вертикали за один временной шаг. Действия – приращения составляющих скорости. Каждая может изменяться на $+1$, -1 или 0 на одном шаге, так что всего имеется девять (3×3) действий. Обе составляющие скорости должны быть неотрицательны и меньше 5, и одновременно могут быть равны 0 только на линии старта. Каждый эпизод начинается в одном из случайно выбранных начальных состояний, когда обе составляющие скорости равны нулю, а заканчивается, когда машина пересечет линию финиша. За каждый шаг до пересечения линии финиша полагается вознаграждение -1 . Если машина задела границу трассы, то она возвращается в случайную позицию на линии старта, обе составляющие скорости обнуляются, и эпизод продолжается. Прежде чем обновлять положение машины на каждом шаге, следует проверить, не пересекает ли предполагаемая траектория движения границу трассы. Если машина пересекает линию финиша, то эпизод заканчивается; если она пересекает какую-то другую границу, то считается, что машина вылетела с трассы, поэтому она возвращается на линию старта. Чтобы сделать задачу более интересной, с вероятностью 0.1 на каждом временном шаге оба приращения, составляющих скорости, равны нулю независимо от того, какими они должны были бы стать. Примените к этой задаче метод управления Монте-Карло, чтобы вычислить оптимальную стратегию при старте из каждого начального состояния. Нарисуйте несколько траекторий, следующих оптимальной стратегии (но подавите для них шум).

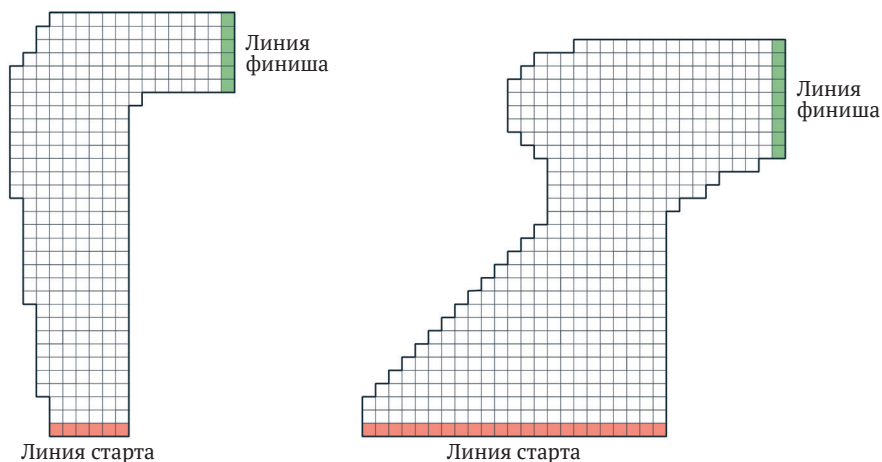


Рис. 5.5 ❖ Два правых поворота в задаче о круговых гонках



5.8. *ВЫБОРКА ПО ЗНАЧИМОСТИ С УЧЕТОМ ОБЕСЦЕНИВАНИЯ

Рассмотренные до сих пор методы с разделенной стратегией основаны на вычислении весов в выборке по значимости для доходов, рассматриваемых как единое целое, без учета внутренней структуры дохода как суммы обесцениваемых вознаграждений. Теперь мы вкратце рассмотрим передовые идеи об использовании этой структуры с целью значительно уменьшить дисперсию оценок, полученных методами с разделенной стратегией.

Например, рассмотрим случай, когда эпизоды длинные и γ намного меньше 1. Конкретно, предположим, что длина эпизода 100 шагов и что $\gamma = 0$. Тогда доход, полученный с момента 0, будет равен просто $G_0 = R_1$, но его коэффициент выборки по значимости является произведением 100 сомножителей $\frac{\pi(A_0|S_0)}{b(A_0|S_0)} \frac{\pi(A_1|S_1)}{b(A_1|S_1)} \dots \frac{\pi(A_{99}|S_{99})}{b(A_{99}|S_{99})}$. В случае обыкновенной выборки по значимости доход масштабируется путем деления на полное произведение, но в действительности необходимо делить только на первый множитель $\frac{\pi(A_0|S_0)}{b(A_0|S_0)}$. Остальные

99 множителей $\frac{\pi(A_1|S_1)}{b(A_1|S_1)} \dots \frac{\pi(A_{99}|S_{99})}{b(A_{99}|S_{99})}$ несут, потому что после получения первого вознаграждения доход уже определен. Эти последующие множители не зависят от дохода и ожидаемой ценности 1; они не сказываются на обновлении математического ожидания, но дают огромный вклад в дисперсию. В некоторых случаях они даже могут сделать дисперсию бесконечной. Перейдем теперь к идее о том, как избежать этой огромной и совершенно ненужной дисперсии.

Смысл идеи заключается в том, чтобы считать обесценивание фактором, определяющим вероятность завершения, или, эквивалентно, *степень* частичного завершения. Для любого $\gamma \in [0, 1)$ можно считать доход G_0 частично конечным после одного шага, в степени $1 - \gamma$, что порождает доход, состоящий из одного лишь первого вознаграждения R ; частично конечным после двух шагов, в степени $(1 - \gamma)\gamma$, что порождает доход $R_1 + R_2$, и т. д. Вторая из указанных выше степеней означает, что произошло завершение на втором шаге $1 - \gamma$, но не произошло на первом шаге, γ . Степень завершения на третьем шаге равна $(1 - \gamma)\gamma^2$, где γ^2 отражает тот факт, что завершения не произошло ни на одном из первых двух шагов. Эти частичные доходы называются *плоскими частичными доходами*:

$$\bar{G}_{t:h} \doteq R_{t+1} + R_{t+2} + \dots + R_h, \quad 0 \leq t < h \leq T,$$

где слово «плоский» означает отсутствие обесценивания, а слово «частичный» – что учитывается не весь доход до завершения эпизода, а только до шага h , называемого горизонтом (и T здесь – время завершения эпизода). Традиционный полный доход G_t можно рассматривать как сумму частичных доходов:

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{T-t-1} R_T \\ &= (1 - \gamma) R_{t+1} \\ &\quad + (1 - \gamma) \gamma (R_{t+1} + R_{t+2}) \\ &\quad + (1 - \gamma) \gamma^2 (R_{t+1} + R_{t+2} + R_{t+3}) \\ &\quad + \dots + (1 - \gamma) \gamma^{T-t-1} (R_{t+1} + R_{t+2} + \dots + R_T) \end{aligned}$$

$$\begin{aligned}
 & + (1 - \gamma)\gamma^2(R_{t+1} + R_{t+2} + R_{t+3}) \\
 & \vdots \\
 & + (1 - \gamma)\gamma^{T-t-2}(R_{t+1} + R_{t+2} + \dots + R_{T-1}) \\
 & + \gamma^{T-t-1}(R_{t+1} + R_{t+2} + \dots + R_T) \\
 & = (1 - \gamma) \sum_{h=t+1}^{T-1} \gamma^{h-t-1} \bar{G}_{t:h} + \gamma^{T-t-1} \bar{G}_{t:T}.
 \end{aligned}$$

Теперь мы должны масштабировать плоские частичные доходы на коэффициент выборки по значимости, который тоже усечен аналогичным образом. Поскольку $\bar{G}_{t:h}$ включает только вознаграждения до горизонта h , мы должны включить в коэффициент вероятности до h . Определим оценку обыкновенной выборки по значимости по аналогии с (5.5):

$$V(s) \doteq \frac{\sum_{t \in T(s)} \left((1 - \gamma) \sum_{h=t+1}^{T(t)-1} \gamma^{h-t-1} \rho_{t:h-1} \bar{G}_{t:h} + \gamma^{T(t)-t-1} \rho_{t:T(h)-1} \bar{G}_{t:T(t)} \right)}{|T(s)|}, \quad (5.9)$$

а оценку взвешенной выборки по значимости – по аналогии с (5.6):

$$V(s) \doteq \frac{\sum_{t \in T(s)} \left((1 - \gamma) \sum_{h=t+1}^{T(t)-1} \gamma^{h-t-1} \rho_{t:h-1} \bar{G}_{t:h} + \gamma^{T(t)-t-1} \rho_{t:T(h)-1} \bar{G}_{t:T(t)} \right)}{\sum_{t \in T(s)} \left((1 - \gamma) \sum_{h=t+1}^{T(t)-1} \gamma^{h-t-1} \rho_{t:h-1} + \gamma^{T(t)-t-1} \rho_{t:T(h)-1} \right)}. \quad (5.10)$$

Будем называть эти две оценки оценками выборки по значимости с учетом обесценивания. Они принимают во внимание коэффициент обесценивания, но не дают никакого эффекта (совпадают с оценками методов с разделенной стратегией из раздела 5.5), если $\gamma = 1$.

5.9. *ПРИВЕДЕННАЯ ВЫБОРКА ПО ЗНАЧИМОСТИ

Существует еще один способ учесть структуру дохода как суммы вознаграждений при выборке по значимости с разделенной стратегией. Этот способ может уменьшить дисперсию в отсутствие обесценивания (т. е. даже когда $\gamma = 1$). В оценках с разделенной стратегией (5.5) и (5.6) каждый член суммы в числителе сам является суммой:

$$\begin{aligned}
 \rho_{t:T-1} G_t &= \rho_{t:T-1} (R_{t+1} + R_{t+2} + \dots + \gamma^{T-t-1} R_T) \\
 &= \rho_{t:T-1} R_{t+1} + \rho_{t:T-1} R_{t+2} + \dots + \gamma^{T-t-1} \rho_{t:T-1} R_T.
 \end{aligned} \quad (5.11)$$

Оценки методов с разделенной стратегией опираются на математические ожидания этих членов, которые можно записать проще. Заметим, что каждое слагаемое в формуле (5.11) является произведением случайного вознаграждения на случайный коэффициент выборки по значимости. Например, первое слагаемое можно, в силу (5.3), записать в виде:

$$\rho_{t:T-1}R_{t+1} = \frac{\pi(A_t|S_t)}{b(A_t|S_t)} \frac{\pi(A_{t+1}|S_{t+1})}{b(A_{t+1}|S_{t+1})} \frac{\pi(A_{t+2}|S_{t+2})}{b(A_{t+2}|S_{t+2})} \dots \frac{\pi(A_{T-1}|S_{T-1})}{b(A_{T-1}|S_{T-1})} R_{t+1}. \quad (5.12)$$

Можно высказать подозрение, что из всех сомножителей только первый и последний (вознаграждение) связаны; все остальные относятся к событиям, которые произошли после получения вознаграждения. Более того, математическое ожидание всех остальных сомножителей равно 1:

$$\mathbb{E} \left[\frac{\pi(A_k|S_k)}{b(A_k|S_k)} \right] = \sum_a b(a|S_k) \frac{\pi(a|S_k)}{b(a|S_k)} = \sum_a \pi(a|S_k) = 1. \quad (5.13)$$

Еще несколько шагов – и можно показать, что, как мы и подозревали, все остальные сомножители не оказывают влияния на математическое ожидание, т. е.

$$\mathbb{E}[\rho_{t:T-1}R_{t+1}] = \mathbb{E}[\rho_{t:t}R_{t+1}]. \quad (5.14)$$

Повторив этот процесс для k -го слагаемого суммы (5.11), получим

$$\mathbb{E}[\rho_{t:T-1}R_{t+1}] = \mathbb{E}[\rho_{t:t+k-1}R_{t+1+k}].$$

Отсюда следует, что математическое ожидание исходного члена (5.11) можно записать в виде:

$$\mathbb{E}[\rho_{t:T-1}G_t] = \mathbb{E}[\tilde{G}_t],$$

где

$$\tilde{G}_t = \rho_{t:t}R_{t+1} + \gamma\rho_{t:t+1}R_{t+2} + \gamma^2\rho_{t:t+2}R_{t+3} + \dots + \gamma^{T-t-1}\rho_{t:T-1}R_T.$$

Мы называем эту идею *приведенной* (per-decision) выборкой по значимости. Отсюда сразу вытекает, что использование $\tilde{G}_t$ дает альтернативную оценку выборки по значимости, с таким же несмещенным математическим ожиданием (в случае первого посещения), как оценка обыкновенной выборки по значимости (5.5):

$$V(s) = \frac{\sum_{t \in \mathcal{T}(s)} \tilde{G}_t}{|\mathcal{T}(s)|}. \quad (5.15)$$

И можно ожидать, что ее дисперсия иногда будет меньше. Существует ли приведенный вариант взвешенной выборки по значимости? Не ясно. Все известные нам предложения такого рода дают несостоятельные оценки (т. е. они не сходятся к истинной ценности на бесконечных данных).

**Упражнение 5.13.* Восполните шаги, необходимые для вывода (5.14) из (5.12). □

**Упражнение 5.14.* Модифицируйте алгоритм управления Монте-Карло с разделенной стратегией (стр. 144), воспользовавшись идеей усеченной средневзвешенной оценки (5.10). Заметим, что сначала нужно будет преобразовать это равенство, введя ценности действий. □

5.10. РЕЗЮМЕ

Методы Монте-Карло, описанные в этой главе, обучаются функциям ценности и оптимальным стратегиям на опыте в форме выборочных эпизодов. Это дает им по меньшей мере три преимущества над методами ДП. Во-первых, их можно использовать для обучения оптимальному поведению непосредственно на взаимодействии с окружающей средой, не имея модели динамики среды. Во-вторых, они применимы совместно с имитационными или выборочными моделями. Есть на удивление много приложений, для которых легко смоделировать выборочные модели, даже если трудно построить явную модель вероятностей переходов, необходимую методам ДП. В-третьих, методы Монте-Карло легко и эффективно применяются к небольшому подмножеству состояний. Область, представляющую особый интерес, можно точно обчислить, не прибегая к точному оцениванию полного множества состояний (мы вернемся к этой теме в главе 8).

Четвертое преимущество методов Монте-Карло, обсуждаемое в этой книге, – тот факт, что они менее уязвимы к нарушениям марковского свойства. Это связано с тем, что они не обновляют оценки ценности на основе оценок ценности последующих состояний. Иными словами, в них отсутствует бутстрэппинг.

При проектировании методов управления Монте-Карло мы следовали общей схеме *обобщенной итерации по стратегиям* (ОИС), введенной в главе 4. ОИС подразумевает взаимодействие процессов оценивания и улучшения стратегии. Методы Монте-Карло предлагают альтернативный процесс оценивания стратегии. Вместо того чтобы использовать модель для вычисления ценности каждого состояния, они просто усредняют много доходов, полученных при старте из этого состояния. Поскольку ценность состояния – это ожидаемый доход, такое среднее может оказаться хорошей аппроксимацией ценности. В методах управления нас особенно интересует аппроксимация функций ценности действий, потому что их можно использовать для улучшения стратегии, не имея модели динамики переходов в окружающей среде. В методах Монте-Карло шаги оценивания и улучшения стратегии чередуются в соседних эпизодах и могут быть реализованы инкрементно на поэпизодной основе.

Обеспечение *достаточного исследования* – проблема для методов управления Монте-Карло. Недостаточно просто выбирать действия с максимальной текущей оценкой, поскольку тогда от альтернативных действий не будет получено никакого дохода, и мы можем никогда не узнать, что они на самом деле лучше. Один из возможных подходов – игнорировать эту проблему, предполагая, что эпизоды начинаются в парах состояние–действие, случайно выбранных так, что они покрывают все возможности. Такие *исследовательские старты* иногда можно организовать в приложениях, в которых эпизоды смоделированы, но маловероятно, что так случится при обучении на реальном опыте. В методах с *единой стратегией* агент обязуется не пренебрегать исследованием и пытается найти оптимальную стратегию, которая продолжает исследовать. В методах с *разделенной стратегией* агент также занимается исследованием, но обучается детерминированной оптимальной стратегии, которая может быть никак не связана со стратегией, которой он следует.

Под *предсказанием с разделенной стратегией* понимают обучение функции ценности *целевой стратегии* на данных, сгенерированных другой *поведенческой стратегией*. Такие методы обучения основаны на той или иной форме *выборки по значимости*, т. е. на сопоставлении доходов весов в виде отношения вероятностей выбрать наблюдаемые действия при следовании обеим стратегиям; тем самым их математические ожидания преобразуются от поведенческой стратегии к целевой. В *обыкновенной выборке по значимости* используется простое среднее взвешенных доходов, а во *взвешенной выборке по значимости* – взвешенное среднее. Обыкновенная выборка по значимости порождает несмещенные оценки, имеющие, однако, большую, быть может, даже бесконечную дисперсию, тогда как взвешенная выборка по значимости всегда дает оценки с конечной дисперсией и на практике более предпочтительна. Несмотря на концептуальную простоту, применение методов Монте-Карло с разделенной стратегией для предсказания и управления еще не устоялось и остается темой активных исследований.

Методы Монте-Карло, рассмотренные в этой главе, отличаются от методов ДП из предыдущей главы в двух главных отношениях. Во-первых, они работают с выборочным опытом и потому могут использоваться для прямого обучения без модели. Во-вторых, в них отсутствует бутстрэппинг, т. е. обновление оценок ценности на основе других оценок ценности. Эти два различия не являются тесно связанными и могут быть разделены. В следующей главе мы рассмотрим методы, в которых обучение происходит на опыте, как в методах Монте-Карло, но при этом присутствует бутстрэппинг, как в методах ДП.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

Термин «Монте-Карло» восходит к 1940-м годам, когда физики из Лос-Аламосской лаборатории придумали и стали изучать азартные игры, чтобы понять сложные физические явления, относящиеся к атомной бомбе. Трактовка методов Монте-Карло в этом смысле имеется в нескольких учебниках (см., например, Kalos and Whitlock, 1986; Rubinstein, 1981).

5.1–2 В работе Singh and Sutton (1996) проведено различие между методами МК первого посещения и всех посещениях и доказаны результаты, связывающие эти методы с алгоритмами обучения с подкреплением. Пример игры в блэкджек основан на примере из работы Widrow, Gupta, and Maitra (1973). Пример с мыльным пузырем – это классическая задача Дирихле, решение которой методом Монте-Карло было впервые предложено в работе Kakutani (1945; см. Hersh and Griego, 1969; Doyle and Snell, 1984).

В работе Barto and Duff (1994) обсуждается оценивание стратегии в контексте классических алгоритмов Монте-Карло для решения систем линейных уравнений. Используется анализ, проведенный в работе Curtiss (1954), чтобы подчеркнуть вычислительные преимущества оценивания стратегий методами Монте-Карло в крупномасштабных задачах.

5.3–4 Метод Монте-Карло ИС был впервые описан в издании этой книги 1998 года. Возможно, это была первая явная связь между оцениванием Монте-Карло и методами управления на основе итерации по стратегиям. Ранее приме-

нение методов Монте-Карло к оцениванию ценности действий в контексте обучения с подкреплением имеется в работе Michie and Chambers (1968). В задаче о балансировании стержня (стр. 83) они использовали среднюю продолжительность эпизодов для оценки ценности (ожидаемого «времени равновесия» стержня) каждого возможного действия в каждом состоянии, а затем использовали эти оценки для управления выбором действий. По духу их метод напоминает Монте-Карло ИС с оценками методом МК всех посещений. В работе Narendra and Wheeler (1986) изучался метод Монте-Карло для эргодических марковских цепей, в котором доход, полученный за время между посещениями одного и того же состояния, использовался в качестве вознаграждения для корректировки вероятностей действий самообучающегося автомата.

- 5.5** Эффективное обучение с разделенной стратегией теперь воспринимается как важная проблема, возникающая в нескольких областях. Например, оно тесно связано с идеей «интервенций» и «противоречий фактам» в вероятностных графических (байесовских) моделях (см., например, Pearl, 1995; Balke and Pearl, 1994). Методы с разделенной стратегией с использованием выборки по значимости имеют долгую историю и до конца еще не поняты. Взвешенная выборка по значимости, иногда называемая также нормированной выборкой по значимости (например, Koller and Friedman, 2009), обсуждается, в частности, в работах Rubinstein (1981), Hesterberg (1988), Shelton (2001) и Liu (2001).

Целевая стратегия в обучении с разделенной стратегией в литературе иногда называется стратегией «оценивания», как было в первом издании этой книги.

- 5.7** Упражнение на тему кольцевых гонок заимствовано с некоторыми изменениями из работ Barto, Bradtke, and Singh (1995) и Gardner (1973).
- 5.8** Наша трактовка идеи о выборке по значимости с учетом обесценивания основана на анализе в работе Sutton, Mahmood, Precup и van Hasselt (2014). Наиболее полное ее развитие на данный момент см. в работах Mahmood (2017) и Mahmood, van Hasselt, and Sutton (2014).
- 5.9** Приведенная выборка по значимости введена в работе Precup, Sutton, and Singh (2000). Там же осуществлено сопряжение обучения с разделенной стратегией с обучением на основе временных различий, следами приемлемости и приближенными методами, что породило тонкие вопросы, обсуждаемые в последующих главах.

Глава 6

Обучение на основе временных различий

Если бы нас попросили назвать одну идею обучения с подкреплением, которая была бы центральной и новаторской, то, без сомнения, мы выбрали бы обучение на основе *временных различий* (temporal-difference – TD). TD-обучение – это сочетание идей, заложенных в методах Монте-Карло и динамическом программировании (ДП). Как и методы Монте-Карло, методы TD позволяют обучаться непосредственно на опыте, не требуя модели динамики окружающей среды. Как и ДП, методы TD обновляют оценки, основываясь в том числе на других обученных оценках, не дожидаясь конечного результата (бутстрэппинг). Связь между TD, ДП и методами Монте-Карло – тема, пронизывающая все обучение с подкреплением; в этой главе мы начнем ее изучать. И по ходу дела увидим, что эти идеи и методы проникают друг в друга и допускают комбинирование разными способами. В частности, в главе 7 мы познакомимся с n -шаговыми алгоритмами, перебрасывающими мост между TD и методами Монте-Карло, а в главе 12 – с алгоритмом TD(λ), который органично объединяет их.

Как обычно, начнем с задачи оценивания стратегии, или предсказания, т. е. с задачи оценивания функции ценности v_π для заданной стратегии π . Для решения задачи управления (нахождения оптимальной стратегии) все методы – ДП, TD и Монте-Карло – пользуются тем или иным вариантом обобщенной итерации по стратегиям (ОИС). Различие между методами связано в основном с подходом к задаче предсказания.

6.1. Предсказание TD-методами

В TD-методах и методах Монте-Карло для решения задачи предсказания используется опыт. Если имеется некоторый опыт следования стратегии π , то те и другие методы обновляют оценку V функции v_π для незаключительных состояний S_t , встречающихся в этом опыте. Грубо говоря, методы Монте-Карло ждут, когда станет известен доход после посещения, а затем используют этот доход как цель для $V(S_t)$. Простой метод Монте-Карло всех посещений, подходящий для нестационарной среды, описывается правилом

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)], \quad (6.1)$$

где G_t – фактический доход, полученный после момента t , а α – постоянный параметр размера шага (см. уравнение 2.4). Назовем этот метод *МК с постоянным α* . Если методы Монте-Карло должны дождаться конца эпизода, чтобы определить приращение $V(S_t)$ (только тогда становится известна величина G_t), то TD-методы должны дожидаться только следующего временного шага. Уже в момент $t + 1$ они формируют цель и совершают полезное обновление, используя наблюдаемое вознаграждение R_{t+1} и оценку $V(S_{t+1})$. Простейший TD-метод выполняет обновление:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \quad (6.2)$$

сразу после перехода в состояние S_{t+1} и получения вознаграждения R_{t+1} . По сути дела, в обновлении методом Монте-Карло целью является G_t , а в обновлении TD-методом – $R_{t+1} + \gamma V(S_{t+1})$. Этот TD-метод называется *TD(0)*, или *одношаговым TD*, т. к. он является частным случаем *TD(λ)*, или *n-шаговых TD-методов*, обсуждаемых в главах 7 и 12. Во врезке ниже приведена полная процедурная форма алгоритма *TD(0)*.

Табличный TD(0) для оценивания v_π

Вход: оцениваемая стратегия π

Параметр алгоритма: размер шага $\alpha \in (0, 1]$

Инициализировать $V(s)$ для всех $s \in \mathcal{S}^+$ произвольным образом с единственным ограничением $V(\text{terminal}) = 0$

Повторять для каждого эпизода:

 Инициализировать S

 Повторять для каждого шага эпизода:

$A \leftarrow$ действие, выбранное стратегией π для S

 Предпринять действие A , наблюдать R, S'

$V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$

$S \leftarrow S'$

 пока S не является заключительным состоянием

Поскольку обновление в *TD(0)* отчасти основано на существующей оценке, говорят, что это метод с *бутстрэппингом*, как ДП. Из главы 3 мы знаем, что

$$v_\pi \doteq \mathbb{E}_\pi[G_t | S_t = s] \quad (6.3)$$

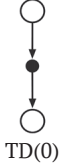
$$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \quad (\text{в силу (3.9)})$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(G_{t+1}) | S_t = s]. \quad (6.4)$$

Грубо говоря, в методах Монте-Карло в качестве цели используется оценка (6.3), тогда как в методах ДП – оценка (6.4). Цель метода Монте-Карло является оценкой, потому что математическое ожидание в формуле (6.3) неизвестно; вместо истинного ожидаемого дохода используется выборочный. Цель ДП является оценкой не из-за математических ожиданий, которые, по предположению, полностью определяются моделью окружающей среды, а потому что значение $v_\pi(S_{t+1})$ неизвестно и вместо него используется текущая оценка $V(S_{t+1})$. В TD-методе цель

является оценкой по обеим причинам: в (6.4) берутся выборочные ожидаемые значения и вместо истинного v_π используется текущая оценка V . Таким образом, TD-методы сочетают выборку методов Монте-Карло с бутстрэппингом ДП. Как мы увидим, при должной аккуратности и изобретательности это поможет в значительной мере объединить достоинства методов Монте-Карло и ДП.

Справа показана диаграмма предшествующих состояний для таблично-го TD(0). Оценка ценности верхнего узла состояния обновляется на основе одного выборочного перехода из него в состояние, непосредственно следующее за ним. Обновления в методах TD и Монте-Карло называются *выборочными*, потому что подразумевают заглядывание вперед в выборочное последующее состояние (или пару состояние–действие) и использование ценности последующего состояния и полученного по дороге дохода для вычисления приращения и соответственного обновления ценности исходного состояния (или пары состояние–действие). *Выборочные* обновления отличаются от *полных* обновлений в методах ДП тем, что основаны на одном выборочном последующем состоянии, а не на знании полного распределения всех возможных последующих состояний.



Наконец, отметим, что величина в квадратных скобках в TD(0)-обновлении – это своего рода ошибка, измеряющая разность между оценкой ценности S_t и лучшей оценкой $R_{t+1} + V(S_{t+1})$. Эта величина, называемая *TD-ошибкой*, возникает в различных формах в разных местах обучения с подкреплением:

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t). \quad (6.5)$$

Заметим, что TD-ошибка на каждом временном шаге – это ошибка оценки, сделанной на этом шаге. Поскольку TD-ошибка зависит от следующего состояния и следующего вознаграждения, она становится доступной только на следующем шаге. Следовательно, δ_t – это ошибка $V(S_t)$, доступная в момент $t + 1$. Заметим также, что если массив V не изменяется на протяжении эпизода (как в методах Монте-Карло), то ошибку Монте-Карло можно записать в виде суммы TD-ошибок:

$$\begin{aligned} G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1}) && \text{(в силу (3.9))} \\ &= \delta_t + \gamma (G_{t+1} - V(S_{t+1})) \\ &= \delta_t + \gamma \delta_{t+1} + \gamma^2 (G_{t+2} - V(S_{t+2})) \\ &= \delta_t + \gamma \delta_{t+1} + \gamma^2 \delta_{t+2} + \cdots + \gamma^{T-t-1} \delta_{T-1} + \gamma^{T-t} (G_T - V(S_T)) \\ &= \delta_t + \gamma \delta_{t+1} + \gamma^2 \delta_{t+2} + \cdots + \gamma^{T-t-1} \delta_{T-1} + \gamma^{T-t} (0 - 0) \\ &= \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k. \end{aligned} \quad (6.6)$$

Это тождество неточно, если V обновляется на протяжении эпизода (как в TD(0)), но если размер шага мал, то оно все же может иметь место приближенно. Обобщения этого тождества играют важную роль в теории и алгоритмах обучения на основе временных различий.

Упражнение 6.1. Если V изменяется на протяжении эпизода, то (6.6) справедливо только приближенно; а какова разность между левой и правой частями? Обо-

значим V_t массив ценностей состояний, используемый в выражении (6.5) для TD-ошибки в момент t и в правиле TD-обновления (6.2). Повторив приведенный выше вывод, найдите дополнительный член, который нужно прибавить к сумме TD-ошибок, чтобы получить ошибку Монте-Карло. □

Пример 6.1: поездка домой на автомобиле. Каждый день, возвращаясь домой с работы, вы пытаетесь спрогнозировать, сколько времени займет дорога. При выходе из офиса вы запоминаете время, день недели, погоду и вообще все, что может оказаться существенным. Скажем, в пятницу вы уходите с работы ровно в 6 вечера и полагаете, что дорога домой займет 30 минут. В 6:05, когда вы подходите к машине, начинается накрапывать дождь. В дождь машины движутся медленнее, поэтому вы изменяете оценку – начиная с этого момента, дорога займет 35 минут, а всего, следовательно, 40 минут. Спустя еще 15 минут вы проехали ту часть дороги до дома, которая идет по автотрассе. Свернув на местную дорогу, вы уменьшаете оценку всего времени до 35 минут. К несчастью, именно в этот момент вы уперлись в еле ползущий грузовик, а дорога слишком узкая для обгона. Приходится тащиться за ним до поворота на боковую улицу, где вы живете, – это происходит в 6:40. Еще три минуты – и вы дома. Последовательность шагов, временных интервалов и предсказаний выглядит так:

Состояние	Прошло времени (минут)	Прогноз оставшегося времени	Прогноз полного времени
Вышел из офиса, пятница, 6 вечера	0	30	30
Дошел до машины, идет дождь	5	35	40
Съехал с трассы	20	15	35
Местная дорога, за грузовиком	30	10	40
Поворот на свою улицу	40	3	43
Доехал до дома	43	0	43

В этом примере вознаграждением является прошедшее время на каждом этапе пути¹. Обесценивания нет ($\gamma = 1$), поэтому доход для каждого состояния равен фактическому времени поездки из этого состояния. Ценность каждого состояния – ожидаемое время поездки. Во втором столбце таблицы приведены текущие оценки ценности каждого встретившегося состояния.

Для иллюстрации работы методов Монте-Карло можно поступить просто – нанести на график предсказанное полное время (последний столбец), как показано на рис. 6.1 (слева). Красные стрелки показывают изменения предсказаний, рекомендованные методом МК с постоянным α (6.1) при $\alpha = 1$. Это в точности ошибки между оценкой ценности (предсказанным временем пути) в каждом состоянии и фактическим доходом (фактическим временем пути). Например, свернув с трассы, вы думали, что до дома осталось всего 15 минут, но на самом деле путь занял 23 минуты. Применив в этот момент уравнение (6.1), мы определим прира-

¹ Если бы это была задача управления, цель в которой – минимизировать время пути, то, конечно, в качестве вознаграждения мы взяли бы прошедшее время с обратным знаком. Но здесь нас интересует только предсказание (оценивание стратегии), поэтому можно не усложнять и работать с положительными числами.

щение оценки оставшегося времени после съезда с трассы. В этот момент ошибка $G_t - V(S_t)$ составляет 8 минут. Предположим, что размер шага $\alpha = 1/2$. Тогда предсказанное оставшееся время после съезда с трассы в результате этого опыта было бы пересмотрено в сторону увеличения на четыре минуты. Пожалуй, в данном случае это слишком много; медленный грузовик – всего лишь неудачное стечение обстоятельств. Как бы то ни было, изменение можно внести только в офлайн-режиме, когда вы уже приехали домой. Лишь в этот момент становится известным фактический доход.

Так ли необходимо ждать окончательного результата, прежде чем начинать обучение? Предположим, что на другой день вы снова оцениваете, что с момента выхода из офиса дорога домой займет 30 минут, но затем застреваете в огромной пробке. Спустя 25 минут после ухода с работы вы все еще ползете бампер к бамперу по трассе. Теперь вы оцениваете, что до дому ехать еще 25 минут, т. е. всего 50 минут. Стоя в пробке, вы уже знаете, что начальная оценка 30 минут была чрезмерно оптимистичной. Обязательно ли ждать до дома, чтобы увеличить оценку, сделанную в начальном состоянии? При использовании метода Монте-Карло обязательно, поскольку вы пока не знаете истинного дохода.

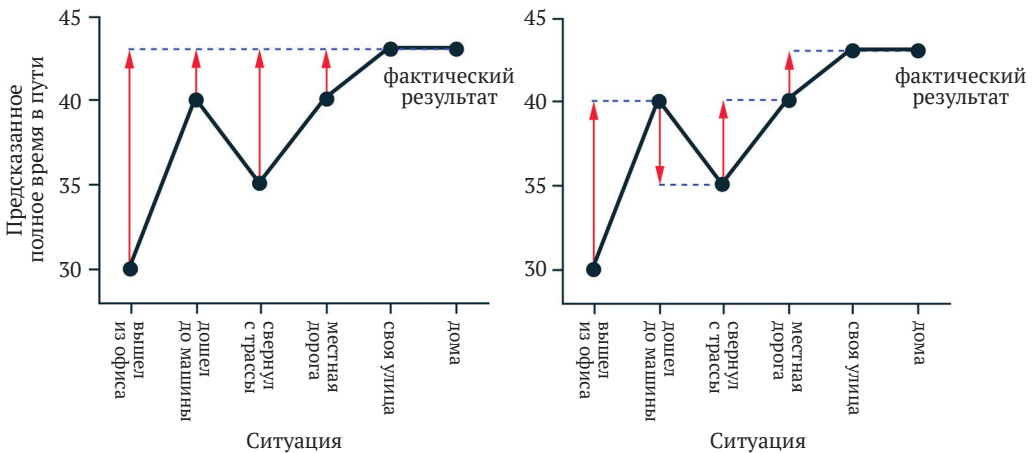


Рис. 6.1 ❖ Изменения, рекомендованные в примере поездки домой методами Монте-Карло (слева) и TD (справа)

С другой стороны, подход TD предполагает, что обучение начинается немедленно, и начальная оценка изменяется с 30 минут на 50. На самом деле любая оценка сдвинулась бы в сторону оценки, следующей сразу за ней. Вернемся к первому дню. На рис. 6.1 (справа) показаны изменения предсказаний, рекомендованные правилом TD-обновления (6.2) (в случае когда $\alpha = 1$). Каждая ошибка пропорциональна изменению предсказаний со временем, т. е. *временным различиям* предсказаний.

Помимо того что у вас появляется чем заняться, стоя в пробке, есть еще несколько вычислительных причин, почему обучаться на текущих предсказаниях выгоднее, чем ждать завершения эпизода, когда станет известен фактический доход. Некоторые из них мы обсудим в следующем разделе. ■

Упражнение 6.2. Это упражнение поможет развить интуитивное понимание того, почему TD-методы часто оказываются эффективнее методов Монте-Карло. Рассмотрим пример с поездкой домой и подход к его решению методами TD и Монте-Карло. Можете ли вы представить себе ситуацию, в которой TD-обновление в среднем было бы лучше обновления Монте-Карло? Приведите пример ситуации – описание прошлого опыта и текущего состояния, – в которой вы ожидаете, что TD-обновление будет лучше. Указание: предположим, что у вас накоплен обширный опыт поездок домой с работы. Затем ваш офис переехал в новое здание с новой парковкой (но на трассу вы въезжаете в том же месте, что и раньше). Теперь вы начинаете обучаться предсказаниям для нового здания. Понимаете ли вы, почему TD-обновления в этом случае, скорее всего, будут гораздо лучше, по крайней мере поначалу? А может ли нечто подобное произойти в исходной ситуации? □

6.2. ПРЕИМУЩЕСТВА TD-МЕТОДОВ ПРЕДСКАЗАНИЯ

TD-методы обновляют оценки отчасти на основе других оценок. Они обучаются строить гипотезы по гипотезам, т. е. *бутстрэппингу*. Хорошо ли это? Каковы преимущества TD-методов по сравнению с методами Монте-Карло и ДП? Постановка таких вопросов и ответы на них составляют содержание оставшейся части этой книги, и не только. В этом разделе мы предвосхитим некоторые ответы.

Очевидное преимущество TD-методов по сравнению с ДП – ненужность модели окружающей среды, т. е. распределений вероятностей вознаграждения и следующего состояния.

Не менее очевидное преимущество TD-методов по сравнению с методами Монте-Карло – то, что для них естественно реализуется онлайн-овый, полностью инкрементный режим. В случае методов Монте-Карло необходимо дожидаться конца эпизода, когда станет известен доход, а TD-методам достаточно подождать всего один временной шаг. Удивительно, но это соображение часто оказывается критическим. В некоторых приложениях эпизоды очень длинные, поэтому если откладывать обучение до конца эпизода, то получится слишком медленно. Другие приложения представляют собой непрерывные задачи, в которых вообще нет эпизодов. Наконец, как было отмечено в предыдущей главе, некоторые методы Монте-Карло обязаны игнорировать или обесценивать эпизоды, в которых предпринимались экспериментальные действия, что может сильно замедлить обучение. TD-методы в гораздо меньшей степени подвержены таким проблемам, потому что они обучаются на каждом переходе вне зависимости от того, какие действия последуют дальше.

Но надежны ли TD-методы? Конечно, удобно обучаться одной гипотезе по следующей за ней, не дожидаясь фактического результата, но можем ли мы при этом гарантировать сходимость к правильному ответу? По счастью, да. Доказано, что для любой фиксированной стратегии π алгоритм TD(0) сходится к v_π в среднем при постоянном размере шага, если он достаточно мал, и с вероятностью 1, если размер шага убывает в соответствии с обычными условиями стохастической аппроксимации (2.7). Большинство доказательств сходимости применимо только к табличному варианту алгоритма, представленному выше в формуле (6.2), но некоторые проходят также в случае аппроксимации линейной функцией общего вида. Эти результаты обсуждаются в более общей постановке в главе 9.

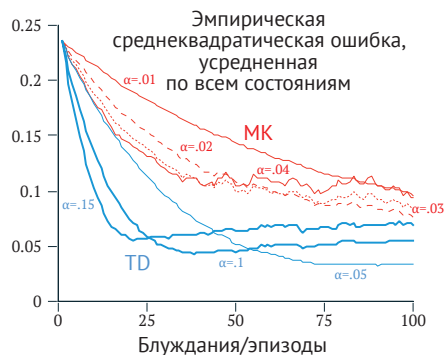
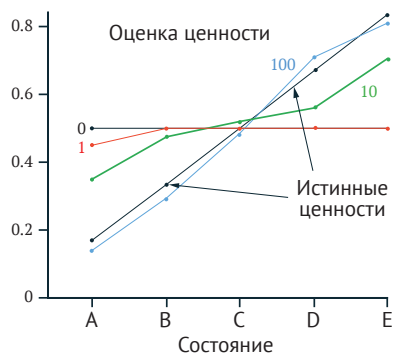
Если и методы TD, и методы Монте-Карло асимптотически сходятся к правильным предсказаниям, то возникает естественный вопрос: «Кто придет первым?» Иными словами, какой метод обучается быстрее? Какой эффективнее распоряжается ограниченными данными? В настоящее время этот вопрос остается открытым – пока не удалось математически доказать, что один метод сходится быстрее другого. На самом деле даже не ясно, как правильно поставить этот вопрос формально! Но на практике TD-методы обычно сходятся быстрее методов МК с постоянным α на стохастических задачах, что иллюстрируется в примере 6.2.

Пример 6.2. Случайное блуждание

В этом примере мы эмпирически сравним предсказательные способности методов TD(0) и МК с постоянным α в применении к следующему марковскому процессу вознаграждения:



Марковским процессом вознаграждения (МПВ) (Markov reward process, MRP) называется марковский процесс принятия решений без действий. Мы часто будем использовать МПВ, говоря о задаче предсказания, в которой нет нужды отличать динамику, связанную с окружающей средой, от динамики, связанной с агентом. В этом МПВ каждый эпизод начинается в центральном состоянии С, а затем на каждом шаге происходит смещение влево или вправо на одно состояние с одинаковой вероятностью. Эпизод заканчивается в крайнем левом или в крайнем правом состоянии. Если эпизод заканчивается в правом состоянии, то выдается вознаграждение +1; в остальных случаях вознаграждение равно 0. Например, типичный эпизод может состоять из следующей последовательности состояний и вознаграждений: С, 0, В, 0, С, 0, D, 0, Е, 1. Поскольку в этой задаче нет обесценивания, то истинная ценность каждого состояния равна вероятности завершения в крайнем правом состоянии при старте из данного состояния. Так, истинная ценность центрального состояния $v_{\pi}(C) = 0.5$. Истинные ценности состояний от А до Е равны 1/6, 2/6, 3/6, 4/6 и 5/6.



На левом графике показаны ценности, обученные после различного количества эпизодов в одном прогоне TD(0). Оценки после 100 эпизодов близки к истинным значениям – при постоянном размере шага ($\alpha = 0.1$ в данном примере) они колеблют-

ся в зависимости от результатов последних эпизодов. На правом графике показаны кривые обучения для обоих методов при различных значениях α . Как мера качества используется среднеквадратическая ошибка (СКО), измеряющая различие между обученной и истинной функциями ценности, усредненное по пяти состояниям, а затем по 100 прогонам. Во всех случаях приближенная функция ценности инициализировалась промежуточным значением $V(s) = 0.5$ для всех s . Для этой задачи TD-метод во всех случаях оказался лучше метода Монте-Карло.

Упражнение 6.3. Из результатов примера случайного блуждания, показанных на левом графике, может показаться, что первый эпизод приводит только к изменению $V(A)$. Какую информацию это несет о том, что происходило в первом эпизоде? Почему изменилась только оценка для одного этого состояния? На сколько точно она изменилась? □

Упражнение 6.4. Конкретные результаты, показанные на правом графике для примера случайного блуждания, зависят от размера шага α . Как вы думаете, повлияло бы это на выводы о том, какой алгоритм лучше, если бы значения α изменялись в более широком диапазоне? Существует ли другое фиксированное значение α , при котором любой из обсуждаемых алгоритмов работал бы значительно лучше, чем показано на рисунке? Объясните свой ответ. □

**Упражнение 6.5.* На правом графике для примера случайного блуждания ошибка СКО TD-метода сначала убывает, а затем снова возрастает, особенно при больших значениях α . Что могло бы быть причиной такого поведения? Как вы думаете, это происходит всегда или зависит от того, как инициализирована приближенная функция ценности? □

Упражнение 6.6. В примере 6.2 мы сказали, что истинные ценности состояний от A до E в примере случайного блуждания равны $1/6$, $2/6$, $3/6$, $4/6$ и $5/6$. Опишите по крайней мере два способа их вычисления. Как вы думаете, каким воспользовались мы? Почему? □

6.3. Оптимальность TD(0)

Предположим, что объем опыта конечен, скажем, 10 эпизодов или 100 временных шагов. В таком случае в инкрементных методах обучения часто применяют такой подход: предлагают этот опыт повторно, пока метод не сойдется к ответу. Если задана приближенная функция ценности V , то на каждом временном шаге t , на котором посещается незаключительное состояние, вычисляются приращения по формуле (6.1) или (6.2), но функция ценности изменяется только один раз – на сумму всех приращений. Затем весь имеющийся опыт обрабатывается заново, но уже с новой функцией ценности, при этом получается новое суммарное приращение. И так далее, пока функция ценности не сойдется. Мы называем этот способ *пакетным обновлением*, потому что обновления производятся только после обработки целого *пакета* обучающих данных.

При пакетном обновлении алгоритм TD(0) детерминированно сходится к единственному ответу, не зависящему от размера шага α , при условии что α выбран достаточно малым. Метод МК с постоянным α при тех же условиях также сходится де-

терминированно, но к другому ответу. Разобравшись с этими двумя ответами, мы лучше поймем различие между обоими методами. При нормальном обновлении методы не переходят сразу к соответствующим пакетным ответам, а в некотором смысле приближаются к ним мелкими шажками. Прежде чем пытаться понять оба ответа в общем случае, для всех возможных задач, рассмотрим несколько примеров.

Пример 6.3: случайное блуждание с пакетным обновлением. Методы TD(0) и МК с постоянным α в вариантах с пакетным обновлением следующим образом применялись к примеру случайного блуждания (пример 6.2). После каждого нового эпизода все наблюдавшиеся до сих пор эпизоды рассматривались как один пакет. Они повторно предъявлялись алгоритму, TD(0) или МК с постоянным α , причем α был достаточно малым для сходимости к функции ценности. Затем получившаяся функция ценности сравнивалась с v_π , и строился график среднеквадратической ошибки по пяти состояниям (и по 100 независимым повторениям всего эксперимента) для получения кривых обучения, изображенных на рис. 6.2. Заметим, что пакетный TD-метод во всех случаях оказался лучше пакетного метода Монте-Карло.

При пакетном обучении метод МК с постоянным α сходится к ценностям $V(s)$, являющимся выборочными средними фактических доходов, полученных экспериментально после посещения каждого состояния s . Эти оценки оптимальны в том смысле, что минимизируют среднеквадратическое отклонение от фактических доходов в обучающем наборе. Тем более удивительно, что пакетный TD-метод смог показать лучшие результаты согласно метрике СКО, как видно по рис. 6.2. Как получилось, что пакетный TD оказался лучше этого оптимального метода? Ответ в том, что метод Монте-Карло оптимален лишь в ограниченном понимании, а TD оптимален в смысле, более релевантном предсказанию доходов.

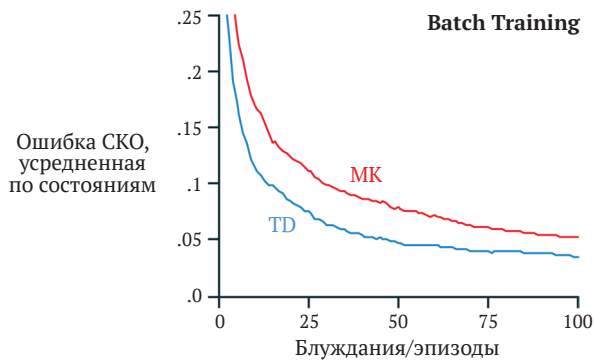


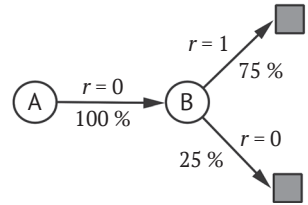
Рис. 6.2 ❖ Качество TD(0) и МК с постоянным α при пакетном обучении на задаче о случайном блуждании

Пример 6.4: вы – предсказатель. Поставьте себя на место предсказателя доходов в неизвестном процессе марковского вознаграждения. Предположим, что вы наблюдаете такие восемь эпизодов:

А, 0, В, 0	В, 1
В, 1	В, 1
В, 1	В, 1
В, 1	В, 0

Это означает, что первый эпизод начался в состоянии А, перешел в В с вознаграждением 0, а затем завершился переходом из В с вознаграждением 0. Остальные семь эпизодов еще короче, они начались в состоянии В и сразу же завершились. При таком наборе данных какие, на ваш взгляд, будут оптимальные предсказания, наилучшие значения оценок $V(A)$ и $V(B)$? Наверное, все согласятся, что оптимальным значением $V(B)$ является $\frac{3}{4}$, потому что в шести из восьми случаев нахождения в состоянии В процесс немедленно завершился с доходом 1, а в оставшихся двух случаях – с доходом 0.

Но каково оптимальное значение оценки $V(A)$ при таких данных? Есть два разумных ответа. Первый – заметить, что в 100 % случаев, когда процесс находился в состоянии А, он сразу же перешел в состояние В (с вознаграждением 0), а поскольку мы уже решили, что ценность В равна $\frac{3}{4}$, то ценность А тоже должна быть равна $\frac{3}{4}$. Этот ответ можно интерпретировать следующим образом: мы сначала смоделировали марковский процесс – он показан на рисунке справа, а затем вычислили правильные оценки при такой модели. Действительно, получается $V(A) = \frac{3}{4}$. Тот же ответ дает пакетный алгоритм TD(0).



Другой разумный ответ – просто заметить, что мы видели А один раз, и за этим последовал доход 0, поэтому мы оцениваем $V(A)$ как 0. Этот ответ дают пакетные методы Монте-Карло. Заметим, что такой же ответ дает минимизация среднеквадратической ошибки на обучающих данных. На самом деле на этих данных получается нулевая ошибка. Но тем не менее нам кажется, что первый ответ лучше. Если процесс марковский, то мы ожидаем, что первый подход даст меньшую ошибку на будущих данных, пусть даже метод Монте-Карло дает лучший ответ на существующих данных. ■

Пример 6.4 иллюстрирует общее различие между оценками, которые вычисляют пакетные методы TD(0) и Монте-Карло. Пакетные методы Монте-Карло всегда находят оценки, минимизирующие среднеквадратическую ошибку на обучающем наборе, тогда как пакетный TD(0) всегда находит оценку, которая была бы в точности правильна для модели максимального правдоподобия марковского процесса. В общем случае оценка максимального правдоподобия параметра – это такая оценка, при которой вероятность генерации имеющихся данных наибольшая. В данном случае оценкой максимального правдоподобия является модель марковского процесса, очевидным образом построенная по наблюдавшимся эпизодам: оценка вероятности перехода из i в j равна доле наблюдавшихся переходов из i в j , а ассоциированное с таким переходом ожидаемое вознаграждение равно среднему вознаграждению, наблюдавшемуся на этих переходах. При такой модели мы можем вычислить оценку функции ценности, которая была бы в точности правильна, если бы в точности правильной была модель. Это называется *стохастически эквивалентной оценкой* (certainty-equivalence estimate), поскольку она эквивалентна предположению о том, что оценка истинного протекающего процесса была достоверно известна, а не аппроксимирована. В общем случае пакетный TD(0) сходится к стохастически эквивалентной оценке.

Это помогает объяснить, почему TD-методы сходятся быстрее методов Монте-Карло. В пакетной форме TD(0) быстрее методов Монте-Карло, потому что он вы-

числяет истинную стохастически эквивалентную оценку. Это объясняет преимущество TD(0), которое мы видим для задачи о случайном блуждании на рис. 6.2. Связь со стохастически эквивалентной оценкой может также частично объяснить преимущество в скорости сходимости непакетного метода TD(0) (пример 6.2 на стр. 158, правый график). Хотя непакетные методы не достигают ни стохастически эквивалентной оценки, ни оценки среднеквадратической ошибки, их все равно можно интерпретировать как движение примерно в этом направлении. Непакетный TD(0) может быть быстрее МК с постоянным α , потому что приближается к лучшей оценке, хотя никогда не достигает ее. В настоящее время нельзя сказать ничего более определенного об относительной эффективности онлайн-методов TD-методов и методов Монте-Карло.

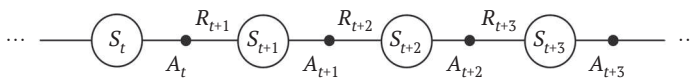
Наконец, стоит отметить, что хотя стохастически эквивалентная оценка в некотором смысле является оптимальным решением, ее почти никогда не удастся вычислить непосредственно. Если $n = |\mathcal{S}|$ – число состояний, то только для формирования оценки максимального правдоподобия процесса может потребоваться память порядка n^2 , а для вычисления соответствующей функции ценности – порядка n^3 шагов вычислений, если оно производится традиционными способами. С этой точки зрения, действительно поражает, что TD-методы способны аппроксимировать то же самое решение, потребляя объем памяти порядка n и повторно применяя вычисления к обучающему набору. На задачах с большим пространством состояний TD-методы могут оказаться единственным практически осуществимым способом аппроксимировать стохастически эквивалентную оценку.

**Упражнение 6.7.* Спроектируйте вариант обновления TD(0) с разделенной стратегией, который можно использовать с произвольной целевой стратегией π и покрывающей поведенческой стратегией b и в котором на каждом шаге t применяется коэффициент выборки по значимости $\rho_{t:t}$ (5.3). $\square$

6.4. SARSA: TD-УПРАВЛЕНИЕ С ЕДИНОЙ СТРАТЕГИЕЙ

Теперь обратимся к использованию TD-методов предсказания для задачи управления. Как обычно, мы следуем принципу обобщенной итерации по стратегиям (ОИС), но на этот раз в части оценивания, или предсказания, используем TD-методы. Как и в случае методов Монте-Карло, мы сталкиваемся с необходимостью поиска компромисса между исследованием и использованием, и снова предлагаемые подходы распадаются на два класса: с единой и с разделенной стратегиями. В этом разделе мы опишем TD-метод управления с единой стратегией.

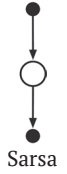
Первый шаг состоит в том, чтобы обучиться функции ценности действий, а не функции ценности состояний. В частности, для метода с единой стратегией мы должны оценить $q_\pi(s, a)$ для текущей поведенческой стратегии π и для всех состояний s и действий a . Это можно сделать, используя по существу тот же TD-метод, что был описан выше для обучения v_π . Напомним, что эпизод состоит из последовательности чередующихся состояний и пар состояние–действие:



В предыдущем разделе мы рассматривали переходы из одного состояния в другое и обучение ценностям состояний. Теперь рассмотрим переходы между парами состояние–действие и обучение ценностям таких пар. Формально оба случая одинаковы: то и другое – марковские цепи с вознаграждением. Теоремы о сходимости ценностей состояний в алгоритме TD(0) применимы также к соответствующему алгоритму для ценностей действий:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]. \quad (6.7)$$

Это обновление производится после каждого перехода из незаключительного состояния S_t . Если состояние S_{t+1} заключительное, то $Q(S_{t+1}, A_{t+1})$ по определению равно 0. В этом правиле встречаются все элементы пятерки $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$, составляющие переход из одной пары состояние–действие в следующую. Эта пятерка и дала название алгоритму – Sarsa. Диаграмма предшествующих состояний для него показана на рисунке справа.



Легко спроектировать алгоритм управления с единой стратегией на основе метода предсказания Sarsa. Как и во всех методах с единой стратегией, мы продолжаем оценивать q_π для поведенческой стратегии π и одновременно изменяем π в сторону жадности относительно q_π . В общем виде алгоритм управления Sarsa приведен во врезке ниже.

Свойства сходимости алгоритма Sarsa зависят от природы зависимости стратегии от Q . Например, можно использовать ε -жадные или ε -мягкие стратегии. Sarsa сходится с вероятностью 1 к оптимальным стратегии и функции ценности действий при условии, что все пары состояние–действие посещаются бесконечное число раз и стратегия в пределе сходится к жадной (что можно организовать, например, в случае ε -жадных стратегий, положив $\varepsilon = 1/t$).

Упражнение 6.8. Покажите, что вариант (6.6) для ценности действий имеет место, если ошибка TD для ценности действий определена как $\delta_t = R_{t+1} + Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$ и снова предполагается, что ценности не изменяются от шага к шагу. □

Sarsa (TD-управление с единой стратегией) для оценивания $Q \approx q_*$

Параметры алгоритма: размер шага $\alpha \in (0, 1]$, небольшое $\varepsilon > 0$

Инициализировать $Q(s, a)$ для всех $s \in S^+$, $a \in \mathcal{A}(s)$ произвольным образом с ограничением $Q(\text{terminal}, \cdot) = 0$

Повторять для каждого эпизода:

 Инициализировать S

 Выбрать A в состоянии S , следуя стратегии, выведенной из Q (например, ε -жадной)

 Повторять для каждого шага эпизода:

 Предпринять действие A , наблюдать R, S'

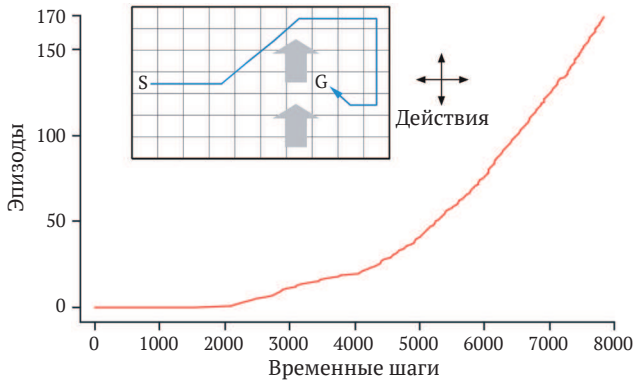
 Выбрать A' в S' , следуя стратегии, выведенной из Q (например, ε -жадной)

$Q(S, A) \leftarrow Q(S, A) + \alpha[R + Q(S', A') - Q(S, A)]$

$S \leftarrow S', A \leftarrow A'$;

 пока состояние S не является заключительным

Пример 6.5: ветреный сеточный мир. На вставке на рисунке ниже показан сеточный мир с начальным и целевым состоянием, в одном отношении отличающийся от стандартного: в середине сетки снизу вверх дует ветер. Имеется четыре стандартных действия – *up*, *down*, *right* и *left*, – но в средней области следующие состояния «сдуваются» вверх ветром, сила которого меняется от столбца к столбцу. Сила ветра показана под каждым столбцом и выражается в числе ячеек, на которые «сдувается» состояние. Например, если мы находимся на одну ячейку правее цели, то действие *left* переведет нас в ячейку, находящуюся прямо над целью. Это эпизодическая задача без обесценивания с постоянным вознаграждением -1 за все действия, не приводящие к переходу в целевое состояние.



На графике выше показаны результаты применения к этой задаче ϵ -жадного алгоритма Sarsa с параметрами $\epsilon = 0.1$, $\alpha = 0.5$ и начальными ценностями $Q(s, a) = 0$ для всех s и a . Возрастающий коэффициент наклона кривой означает, что со временем цель достигается быстрее. К 8000 временных шагов жадная стратегия уже давно оптимальна (выбранная ей траектория показана на вставке); благодаря продолжающемуся ϵ -жадному исследованию средняя длина эпизода составляет примерно 17 шагов, на два больше минимальной длины 15. Заметим, что для этой задачи использовать методы Монте-Карло было бы затруднительно, потому что не для всех стратегий гарантировано завершение. Если бы была найдена стратегия, при которой агент остается в одном и том же состоянии, то следующий эпизод никогда не закончился бы. В пошаговых методах обучения типа Sarsa этой проблемы нет, потому что они уже в процессе эпизода быстро обучаются, что такие стратегии не годятся, и переключаются на что-то другое. ■

Упражнение 6.9: ветреный сеточный мир с ходами короля. Решите задачу о ветренном сеточном мире, считая, что возможных действий не четыре, а восемь – добавляются ходы по диагонали. Насколько лучшее решение вы сможете найти, располагая дополнительными действиями? Можно ли еще улучшить его, включив девятое действие – не двигаться самому, а положиться только на силу ветра? □

Упражнение 6.10: стохастический ветер. Решите задачу о ветренном сеточном мире с ходами короля в предположении, что воздействие ветра если и есть, то стохастическое: иногда оно отклоняется на 1 от средних значений, указанных под

столбцом. Точнее, треть времени сила ветра средняя (как в предыдущем упражнении), еще треть времени вас сдувает на дополнительную ячейку вверх, а еще треть – на дополнительную ячейку вниз. Например, если вы находитесь на одну ячейку правее цели и сдвигаетесь влево, то в одной трети случаев вы окажетесь на одну ячейку выше цели, в одной трети – на две ячейки выше цели и еще в одной трети – точно в целевой ячейке. $\square$

6.5. Q-ОБУЧЕНИЕ: TD-УПРАВЛЕНИЕ С РАЗДЕЛЕННОЙ СТРАТЕГИЕЙ

Одним из ранних прорывов в обучении с подкреплением стала разработка TD-алгоритма управления с разделенной стратегией, известного под названием Q-обучение (Q-learning) (Watkins, 1989), который определяется правилом

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]. \quad (6.8)$$

В этом случае обученная функция ценности действий Q непосредственно аппроксимирует q_* , оптимальную функцию ценности действий, независимо от того, какой стратегии следует агент. Это существенно упрощает анализ алгоритма, так что доказательства сходимости появились почти сразу. Стратегия по-прежнему играет роль, поскольку определяет, какие пары состояние–действие посещаются и обновляются. Однако для сходимости требуется лишь, чтобы обновление всех пар продолжалось. Как было отмечено в главе 5, это минимальное требование в том смысле, что в общем случае таким свойством должен обладать любой метод, который гарантированно находит оптимальное поведение. В этом предположении и при соблюдении варианта обычных условий стохастической аппроксимации на последовательность размеров шагов доказано, что Q сходится к q_* с вероятностью 1. Ниже приведена процедурная форма алгоритма Q-обучения.

Q-обучение (TD-управление с разделенной стратегией) для оценивания $\pi \approx \pi_*$

Параметры алгоритма: размер шага $\alpha \in (0, 1]$, небольшое $\varepsilon > 0$
Инициализировать $Q(s, a)$ для всех $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$ произвольным образом
с ограничением $Q(\text{terminal}, \cdot) = 0$

Повторять для каждого эпизода:

Инициализировать S

Повторять для каждого шага эпизода:

Выбрать A в состоянии S , следуя стратегии, выведенной из Q (например, ε -жадной)

Предпринять действие A , наблюдать R, S'

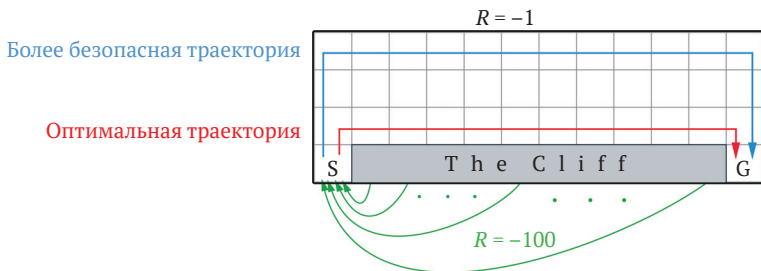
$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$

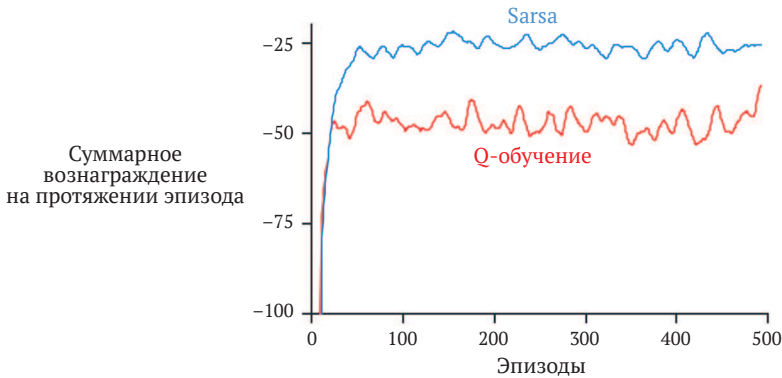
пока состояние S не является заключительным

Как выглядит диаграмма предшествующих состояний для Q-обучения? Правило (6.8) обновляет пару состояние–действие, поэтому в корне обновления должен находиться маленький черный кружок, соответствующий действию. Обновление также производится из узлов действий, при этом берется максимум по всем действиям, возможным в следующем состоянии. Таким образом, внизу диаграммы предшествующих состояний должны находиться все эти узлы действий. Наконец, вспомним, что операция взятия максимума по узлам «следующего действия» обозначается поперечной дугой (рис. 3.4 справа). Так сможете ли теперь сообразить, как выглядит диаграмма? Попробуйте, прежде чем посмотреть ответ на рис. 6.4 ниже.

Пример 6.6: блуждание на краю обрыва. В этом примере сеточного мира сравниваются алгоритмы Sarsa и Q-обучения и подчеркивается различие между методами с единой стратегией (Sarsa) и с разделенной стратегией (Q-обучение). Рассмотрим сеточный мир на рисунке справа. Это стандартная эпизодическая задача без обесценивания с начальным и целевым состояниями и обычными действиями up, down, right и left. Вознаграждение равно -1 для всех переходов, кроме ведущих в область, помеченную «The Cliff» (обрыв). За вход в эту область начисляется вознаграждение -100 , после чего агент мгновенно переносится в начальную позицию.



На графике ниже показано качество методов Sarsa и Q-обучения с ϵ -жадным выбором действия при $\epsilon = 0.1$. После начального переходного периода метод Q-обучения обучается оптимальной стратегии, которой соответствует траектория, проходящая точно вдоль края обрыва. К сожалению, при этом иногда агент срывается с обрыва из-за ϵ -жадного выбора действия. С другой стороны, Sarsa принимает во внимание выбор действия и обучается более безопасному, хотя и более длинному пути в верхней части сетки. Хотя метод Q-обучения действительно находит ценности, определяющие оптимальную стратегию, его качество в онлайн-режиме хуже, чем у Sarsa, который обучается околной стратегии. Конечно, если постепенно уменьшать ϵ , то оба метода асимптотически сойдутся к оптимальной стратегии.



Упражнение 6.11. Почему Q-обучение считается методом управления с разделенной стратегией? ☐

Упражнение 6.12. Предположим, что выбор действия жадный. Будет ли тогда алгоритм Q-обучения в точности совпадать с Sarsa? Будут ли они выбирать в точности одинаковые действия и одинаково обновлять веса? ☐

6.6. EXPECTED SARSA

Рассмотрим алгоритм обучения, который во всем похож на Q-обучение, но, вместо того чтобы брать максимум по парам следующее состояние–действие, вычисляет математическое ожидание, принимая во внимание, с какой вероятностью каждое действие выбирается при текущей стратегии. То есть правило обновления в этом алгоритме выглядит так:

$$\begin{aligned} Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \mathbb{E}_\pi [Q(S_{t+1}, A_{t+1}) | S_{t+1}] - Q(S_t, A_t)] \\ &\leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \sum_a \pi(a | S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \right], \end{aligned} \quad (6.9)$$

но в остальном он следует схеме Q-обучения. Если следующее состояние S_{t+1} , то этот алгоритм *детерминированно* движется в том же направлении, что и Sarsa *в смысле математического ожидания*, поэтому называется *Expected Sarsa*. Его диаграмма предшествующих состояний показана на рис. 6.4 справа.

Вычислительно Expected Sarsa сложнее, чем Sarsa, но зато устраняет дисперсию из-за случайного выбора A_{t+1} . На одном и том же объеме опыта можно ожидать, что он будет работать немного лучше Sarsa, и это действительно так. На рис. 6.3 показаны сводные результаты работы Expected Sarsa, Sarsa и Q-обучения в задаче о блуждании на краю обрыва. Expected Sarsa сохраняет значительное преимущество над Q-обучением, свойственное Sarsa. Кроме того, Expected Sarsa демонстрирует заметное улучшение по сравнению с Sarsa в широком диапазоне значений размера шага. В задаче о блуждании на краю обрыва все переходы состояний детерминированны, а случайность исходит только от стратегии. В таких случаях

Expected Sarsa может положить $\alpha = 1$, не опасаясь ухудшения асимптотического поведения, тогда как Sarsa может хорошо работать в долгосрочной перспективе только при малых значениях α , при которых ухудшается качество работы на коротком отрезке времени. В этом и других примерах эмпирически наблюдается устойчивое преимущество Expected Sarsa над Sarsa.

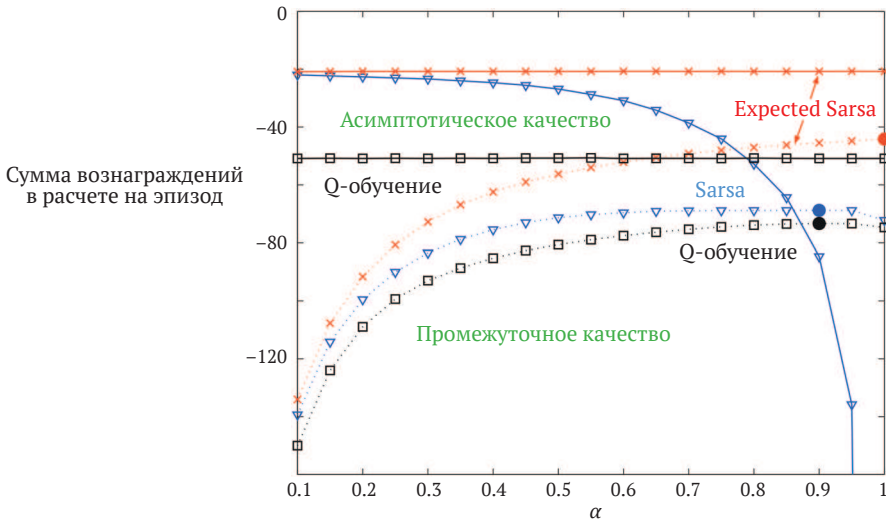


Рис. 6.3 ❖ Промежуточное и асимптотическое качество TD-методов управления в задаче о блуждании на краю обрыва в виде функции от α . Во всех алгоритмах использовалась ε -жадная стратегия с $\varepsilon = 0.1$. Асимптотическое качество получено усреднением по 100 000 эпизодов, а промежуточное – по первым 100 эпизодам. Эти данные получены усредненным по 50 000 и 10 прогонам для промежуточного и асимптотического случаев соответственно. Сплошными кружками обозначено наилучшее промежуточное качество каждого метода. Заимствовано из работы Seijen et al. (2009)

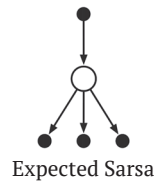
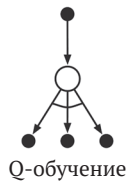


Рис. 6.4 ❖ Диаграммы предшествующих состояний для алгоритмов Q-обучения и Expected Sarsa

В этих результатах алгоритм Expected Sarsa использовался с единой стратегией, но в общем случае может использовать для порождения поведения стратегию, отличную от целевой; в таком случае он становится алгоритмом с разделенной стратегией. Например, предположим, что π – жадная политика, а поведению в большей степени присуще исследование; тогда Expected Sarsa в точности совпадает с Q-обучением. В этом случае Expected Sarsa является обобщением Q-обучения и в то же время безусловным улучшением Sarsa. Если бы не небольшие дополнительные вычислительные затраты, то Expected Sarsa мог бы полностью заместить два других, более известных TD-алгоритма управления.

6.7. СМЕЩЕНИЕ МАКСИМИЗАЦИИ И ДВОЙНОЕ ОБУЧЕНИЕ

Все рассмотренные выше алгоритмы управления включают максимизацию в построение целевых стратегий. Например, в Q-обучении целевой является жадная стратегия при известных текущих ценностях действий, а в определении жадности участвует операция $\max$. В Sarsa стратегия часто ε -жадная, т. е. взятие максимума также присутствует. В этих алгоритмах вычисление максимума по оценкам ценностей используется неявно в виде оценки максимальной ценности, что может приводить к значительному положительному смещению. Чтобы понять, почему это так, рассмотрим одно состояние s , в котором имеется много действий a , чья истинная ценность $q(s, a)$ всегда равна нулю, а оценки ценности $Q(s, a)$ недостоверны – одни больше, другие меньше нуля. Максимум по истинным ценностям равен 0, но максимум по оценкам положителен, так что налицо положительное смещение, которое называется *смещением максимизации*.

Пример 6.7: пример смещения максимизации. Небольшой МППР на вставке в рис. 6.5 дает простой пример того, как максимизация смещения может навредить качеству TD-алгоритмов управления. В этом МППР два незаклучительных состояния, А и В. Все эпизоды начинаются в А, где есть выбор из двух действий: left и right. Действие right сразу переводит в заклучительное состояние с нулевым вознаграждением и доходом. Действие left переводит (также с нулевым вознаграждением) в состояние В, из которого много возможных действий, и все они сразу же ведут в заклучительное состояние, а вознаграждение выбирается из нормального распределения со средним 0.1 и дисперсией 1.0. Таким образом, ожидаемый доход для любой траектории, начинающейся действием left, равен -0.1 , поэтому выбор left в состоянии А всегда является ошибкой. Тем не менее наши методы управления могут отдать предпочтение left из-за смещения максимизации, благодаря которому ценность В выглядит положительной. На рис. 6.5 показано, что в этом примере Q-обучение с ε -жадным выбором действия поначалу отдает явное предпочтение действию left. Даже асимптотически алгоритм Q-обучения выбирает действие left примерно на 5 % чаще, чем было бы оптимально при наших значениях параметров ($\varepsilon = 0.1$, $\alpha = 0.1$, $\gamma = 1$).

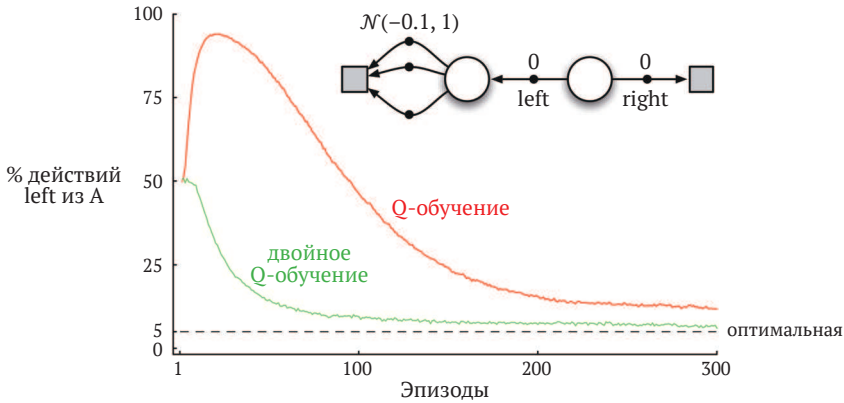


Рис. 6.5 ❖ Сравнение Q-обучения с двойным Q-обучением на примере простого эпизодического МППР (показан на вставке). Алгоритм Q-обучения сначала обучается выбирать действие left гораздо чаще, чем right, и всегда выбирает его значительно чаще, чем 5-процентная минимальная вероятность, гарантируемая ϵ -жадным выбором действий при $\epsilon = 0.1$. Напротив, на алгоритм двойного Q-обучения смещение максимизации не оказывает практически никакого влияния. Данные получены усреднением по 10 000 прогонов. Начальные оценки ценности действий равны нулю. Неоднозначности при ϵ -жадном выборе действий разрешались случайным образом

Существуют ли алгоритмы, избегающие смещения максимизации? Для начала рассмотрим алгоритм бандита, в котором мы имеем зашумленные оценки ценности каждого из многих действий, полученные усреднением вознаграждений за все игры после каждого действия. Как мы уже сказали, положительное смещение максимизации будет иметь место, если использовать максимальную оценку в качестве оценки максимума истинных значений ценности. Одно из возможных объяснений проблемы таково: мы используем одну и ту же выборку (игры) как для определения действия, доставляющего максимум, так и для оценки его ценности. Давайте разобьем все множество игр на два подмножества и будем использовать их для обучения двух независимых оценок, $Q_1(a)$ и $Q_2(a)$, истинной ценности $q(a)$ для всех $a \in \mathcal{A}$. Тогда можно было бы взять одну оценку, скажем Q_1 , для определения доставляющего максимум действия $A^* = \operatorname{argmax}_a Q_1(a)$, а другую, Q_2 , — для оценки ценности этого действия, $Q_2(A^*) = Q_2(\operatorname{argmax}_a Q_1(a))$. Тогда эта оценка будет несмещенной в том смысле, что $E[Q_2(A^*)] = q(A^*)$. Этот процесс можно повторить, поменяв обе оценки ролями, и получить тем самым вторую несмещенную оценку, $Q_1(\operatorname{argmax}_a Q_2(a))$. В этом и заключается идея двойного обучения. Заметим, что хотя мы обучаем две оценки, при каждой игре обновляется только одна; двойное обучение требует двойного объема памяти, но не увеличивает объем вычислений на каждом шаге.

Идея двойного обучения естественно обобщается на алгоритмы для полных МППР. Например, алгоритм двойного обучения, аналогичный Q-обучению и называемый двойным Q-обучением (Double Q-learning), разбивает множество временных шагов на два подмножества, например путем подбрасывания монеты. Если выпадает орел, то обновление производится по правилу

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha [R_{t+1} + \gamma Q_2(S_{t+1}, \operatorname{argmax}_a Q_1(S_{t+1}, a)) - Q_1(S_t, A_t)]. \quad (6.10)$$

Если же выпадает решка, то в этом правиле Q_1 и Q_2 меняются местами, поэтому обновляется Q_2 . Обе приближенные функции ценности обрабатываются совершенно симметрично. В поведенческой стратегии можно использовать обе оценки ценности действий. Например, ε -жадная стратегия в двойном Q-обучении может быть основана на среднем (или сумме) обеих оценок. На врезке ниже алгоритм двойного Q-обучения приведен полностью. С помощью этого алгоритма были получены результаты на рис. 6.5. В этом примере двойное обучение, похоже, устранило вред, причиненный смещением максимизации. Разумеется, существуют также двойные версии алгоритмов Sarsa и Expected Sarsa.

Двойное Q-обучение для оценивания $Q_1 \approx Q_2 \approx q^*$

Параметры алгоритма: размер шага $\alpha \in (0, 1]$, небольшое $\varepsilon > 0$

Инициализировать $Q_1(s, a)$ и $Q_2(s, a)$ для всех $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$ произвольным образом с ограничением $Q(\text{terminal}, \cdot) = 0$

Повторять для каждого эпизода:

 Инициализировать S

 Повторять для каждого шага эпизода:

 Выбрать A в состоянии S , следуя ε -жадной стратегии относительно $Q_1 + Q_2$

 Предпринять действие A , наблюдать R, S'

 С вероятностью 0.5

$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha (R + \gamma Q_2(S', \operatorname{argmax}_a Q_1(S', a)) - Q_1(S, A))$

 иначе

$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha (R + \gamma Q_1(S', \operatorname{argmax}_a Q_2(S', a)) - Q_2(S, A))$

$S \leftarrow S'$

пока состояние S не является заключительным

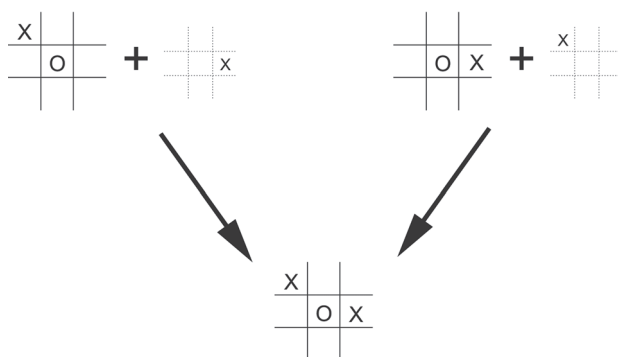
*Упражнение 6.13. Как выглядят уравнения обновления для двойной версии алгоритма Expected Sarsa с ε -жадной целевой стратегией?

6.8. Игры, послесостояния и другие специальные случаи

В этой книге мы стремимся представить единый подход к широкому классу задач, но, конечно, всегда будут особые задачи, для которых лучше подобрать специальные способы решения. Например, наш общий подход подразумевает обучение функции ценности действий, но в главе 1 мы описали TD-метод для обучения игре в крестики-нолики, который обучал нечто, гораздо больше похожее на функцию ценности состояний. Если взглянуть на пример более пристально, то станет понятно, что обученная функция не является ни функцией ценности действий, ни функцией ценности состояний в обычном смысле. Традиционная функция ценности состояний оценивает состояния, в которых агент имеет возможность выбрать действие, тогда как функция ценности состояний, используемая в крес-

тиках-ноликах, оценивает позиции доски, после того как агент сделал ход. Назовем это *последствиями*, а соответствующие функции ценности – *функциями ценности последствий*. Последствия полезны, если у нас имеется информация о начальной части динамики окружающей среды, но не обязательно о полной динамике. Например, в играх мы обычно знаем непосредственные последствия своих ходов. Для каждого возможного хода в шахматах мы знаем, какая получится позиция, но не знаем, как ответит противник. Функции ценности последствий – естественный способ воспользоваться такой информацией и предложить более эффективный метод обучения.

Причина, по которой более эффективно проектировать алгоритмы в терминах последствий, станет понятна в результате анализа игры в крестики-нолики. Традиционная функция ценности действий сопоставляла бы позициям и ходам оценки ценности. Но есть много пар позиция–ход, которые приводят к одной и той же позиции, как показано в примере ниже: в таких случаях пары позиция–ход различны, но дают одну и ту же «последпозицию», а значит, должны иметь одинаковую ценность. Традиционная функция ценности действий должна была бы раздельно оценивать обе пары, тогда как функция ценности последствий сразу оценивает их одинаково. Любая полученная при обучении информация для левой пары позиция–ход сразу же переносится на правую пару.



Последствия возникают во многих задачах, а не только в играх. Например, в задачах массового обслуживания существуют такие действия, как назначение заказчиков серверам, отклонение заказчиков или отбрасывание информации. В подобных случаях действия фактически определяются в терминах их непосредственных последствий, которые точно известны.

Невозможно описать все мыслимые виды специальных задач и соответствующих им алгоритмов обучения. Однако принципы, излагаемые в этой книге, все же имеют широкое применение. Например, методы, основанные на последствиях, как нельзя лучше описываются в терминах обобщенной итерации по стратегиям, причем стратегия и функция ценности последствий взаимодействуют, по сути, таким же образом. Во многих случаях по-прежнему стоит выбор между методами с единой и с разделенной стратегиями, если требуется управлять необходимостью постоянного исследования.

Упражнение 6.14. Опишите, как можно переформулировать задачу Джека об аренде машин (пример 4.1) в терминах послесостояний. Почему в этой конкретной задаче такая постановка, скорее всего, приведет к ускорению сходимости? □

6.9. РЕЗЮМЕ

В этой главе мы познакомились с новым видом методов обучения – обучением на основе временных различий (TD-обучением) – и показали, как они применяются к задаче обучения с подкреплением. Как обычно, мы разбили задачу на две части: предсказание и управление. TD-методы представляют альтернативу методам Монте-Карло для решения задачи предсказания. В обоих случаях в основе обобщения на задачу управления лежит идея обобщенной итерации по стратегиям (ОИС), которую мы вынесли из динамического программирования. Идея эта заключается в том, что приближенные стратегия и функции ценности должны взаимодействовать так, чтобы одновременно приближаться к своим оптимальным значениям.

Один из двух процессов, составляющих ОИС, побуждает функцию ценности точно предсказывать доход для текущей стратегии; это задача предсказания. Второй процесс побуждает стратегию к локальному улучшению (например, ценой ε -жадности) относительно текущей функции ценности. Если первый процесс основан на опыте, то возникает осложнение, связанное с необходимостью поддерживать достаточный уровень исследования. Мы можем классифицировать TD-методы управления по тому, как они справляются с этой трудностью – применяя подход с единой или с разделенной стратегией. Sarsa – метод с единой стратегией, а Q-обучение – с разделенной стратегией. Метод Expected Sarsa в том виде, в котором мы его описали, также является методом с разделенной стратегией. Существует и третий способ обобщения TD-методов на управление, который мы не включили в эту главу. Это методы типа исполнитель–критик. Они будут подробно рассмотрены в главе 13.

Представленные в данной главе методы в настоящее время являются наиболее широко распространенными методами обучения с подкреплением. Возможно, это объясняется их простотой: их можно применять в онлайн-режиме с минимальным объемом вычислений к опыту, сгенерированному на основе взаимодействия с окружающей средой; их можно почти полностью выразить с помощью одного уравнения, легко реализуемого небольшой компьютерной программой. В следующих нескольких главах мы разовьем эти алгоритмы, сделав их немного сложнее и значительно мощнее. Все новые алгоритмы будут сохранять существенные черты описанных выше: способность обрабатывать опыт в онлайн-режиме при относительно небольшом объеме вычислений и участие в алгоритме ошибок TD. Частные случаи TD-методов, описанные в этой главе, правильнее было бы назвать *одношаговыми табличными безмодельными* TD-методами. В следующих двух главах мы обобщим их, введя n -шаговые формы (связь с методами Монте-Карло) и формы, включающие модель окружающей среды (связь с планированием и динамическим программированием). Затем во второй части книги мы обобщим их на различные формы аппроксимации функций вместо таблиц (связь с глубоким обучением и искусственными нейронными сетями).

Наконец, в этой главе мы обсуждали TD-методы только в контексте задач обучения с подкреплением, но на самом деле они носят более общий характер. Это общие методы обучения, позволяющие делать долгосрочные предсказания о динамических системах. Например, TD-методы могут применяться к предсказанию финансовых показателей, продолжительности жизни, исхода выборов, погоды, поведения животных, спроса на электроэнергию или действий покупателей. Лишь после того как TD-методы начали анализировать как методы чистого предсказания, независимые от обучения с подкреплением, стали наконец понятны их теоретические свойства. Но, несмотря на это, существуют потенциальные применения методов TD-обучения, которые еще только ждут досконального исследования.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

В главе 1 мы уже отмечали, что идея TD-обучения уходит корнями в психологию обучения животных и искусственный интеллект, и в первую очередь в работы Samuel (1959) и Klopff (1972). Работа Сэмюэла описывается в примере в разделе 16.2. К TD-обучению примыкают также ранние идеи из работ Holland (1975, 1976) о согласованности предсказаний ценности. Они оказали влияние на одного из авторов, который в период с 1970 по 1975 год был аспирантом в Мичиганском университете, где преподавал Холланд. Идеи Холланда привели к созданию целого ряда TD-подобных систем, в т. ч. работы Booker (1982) и приборов с зарядовой связью типа «пожарная цепочка» из работы Holland (1986), связанной с обсуждаемым ниже методом Sarsa.

6.1–2 Большая часть конкретных материалов из этих разделов взята из статьи Sutton (1988), включая алгоритм TD(0), пример случайного блуждания и термин «обучение на основе временных различий». На способ анализа связей с методами динамического программирования и Монте-Карло оказали влияние работы Watkins (1989), Werbos (1987) и др. Диаграммы предшествующих состояний впервые были использованы в первом издании этой книги.

Сходимость табличного метода TD(0) в среднем была доказана в работе Sutton (1988), а сходимость с вероятностью 1 – в работе Dayan (1992), основанной на работе Watkins and Dayan (1992). Эти результаты были дополнены и усилены в работах Jaakkola, Jordan, and Singh (1994) и Tsitsiklis (1994) с применением обобщений существующей мощной теории стохастической аппроксимации. Другие дополнения и обобщения будут рассмотрены в последующих главах.

6.3 Оптимальность TD-алгоритма установлена в работе Sutton (1988). Истолкование этого результата можно найти в работе Barnard (1993), где TD-алгоритм строится как сочетание двух шагов: инкрементный метод для обучения модели марковской цепи и метод для вычисления предсказаний с использованием этой модели. Термин *стохастически эквивалентная оценка* заимствован из литературы по адаптивному управлению (см., например, Goodwin and Sin, 1984).

- 6.4** Алгоритм Sarsa впервые был исследован в работе Rummery and Niranjan (1994) в связи с искусственными нейронными сетями, где назван «модифицированным коннекционистским Q-обучением». Название «Sarsa» предложено в работе Sutton (1996). Сходимость одношагового табличного алгоритма Sarsa (случай, рассмотренный в этой главе) доказана в работе Singh, Jaakkola, Littman, and Szepesvári (2000). Пример «ветреного сеточного мира» предложен Томом Колтом (Tom Kalt).

Идея пожарной цепочки, высказанная в работе Holland (1986), трансформировалась в алгоритм, тесно связанный с Sarsa. Первоначально эта идея включала цепочки правил, активирующих друг друга, и в центре ее была передача данных от текущего правила обратно к активировавшим его правилам. Со временем идея пожарной цепочки стала больше походить на TD-обучение, в котором данные передаются обратно любому предшествующему по времени правилу, а не только тем, которые активировали текущее правило. Современная форма пожарной цепочки, после упрощения различными естественными способами, почти совпадает с одношаговым алгоритмом Sarsa и подробно описана в статье Wilson (1994).

- 6.5** Q-обучение впервые представлено в работе Watkins (1989); приведенный в ней набросок доказательства сходимости оформлен в виде строгого доказательства в работе Watkins and Dayan (1992). Более общие результаты о сходимости доказаны в работах Jaakkola, Jordan, and Singh (1994) и Tsitsiklis (1994).
- 6.6** Алгоритм Expected Sarsa впервые описан в работе George John (1994), где был назван «Q-обучением»; там же были подчеркнуты его преимущества по сравнению с Q-обучением, обусловленные тем, что это алгоритм с разделенной стратегией. Мы были незнакомы с работой Джона, когда излагали алгоритм Expected Sarsa в первом издании этой книги; эта работа также была неизвестна авторам статьи van Seijen, van Hasselt, Whiteson, and Weir-ing (2009), в которой установлены свойства сходимости Expected Sarsa и условия, при которых он дает лучшие результаты, чем стандартный алгоритм Sarsa и Q-обучение. Наш рис. 6.3 основан на результатах, описанных в этой статье. Ван Сейен с соавторами определили алгоритм «Expected Sarsa» исключительно как метод с единой стратегией (мы в первом издании посту-пили точно так же), но теперь мы используем это название для общего ал-горитма, в котором целевая и поведенческая стратегии могут различаться. Возможность интерпретировать Expected Sarsa как общий алгоритм с раз-деленной стратегией была подмечена в работе Hasselt (2011), где он назван «обобщенным Q-обучением».
- 6.7** Смещение максимизации и двойное обучение введены в рассмотрение и подробно изучены в работах van Hasselt (2010, 2011). Пример МППР на рис. 6.5 получен адаптацией рис. 4.1 из работы van Hasselt, 2011.
- 6.8** Понятие послесостояния – то же самое, что «состояние после принятия ре-шения», встречающееся в работах Van Roy, Bertsekas, Lee, and Tsitsiklis, 1997 и Powell, 2011.

Глава 7

n -шаговый бутстрэппинг

В этой главе мы объединим методы Монте-Карло (МК) и одношаговые методы на основе временных различий (TD-методы), представленные в двух предыдущих главах. Ни методы МК, ни TD-методы не являются однозначно лучшими. Мы опишем n -шаговые TD-методы, которые обобщают те и другие, так чтобы можно было естественно переходить от одного к другому по мере необходимости, чтобы удовлетворить требованиям конкретной задачи. На одном конце спектра n -шаговых методов находятся МК, а на другом – TD-методы. Зачастую лучший метод находится где-то между этими двумя крайностями.

Другой взгляд на преимущества n -шаговых методов заключается в том, что они избавляют нас от тирании временного шага. В случае одношаговых TD-методов один и тот же временной шаг определяет сразу две вещи: как часто можно изменять действие и временной интервал, на котором выполняется бутстрэппинг. Во многих приложениях желательно обновлять действие очень быстро, чтобы учесть, что изменилось, но бутстрэппинг работает оптимально, когда занимает время, в течение которого произошло существенное и очевидное изменение состояния. Поскольку в одношаговых TD-методах оба временных интервала совпадают, приходится искать компромисс. В случае n -шаговых методов бутстрэппинг может быть разнесен на несколько шагов, так что мы перестаем быть рабами единого временного шага.

Идея n -шаговых методов обычно используется как введение в алгоритмическую идею *следов приемлемости* (глава 12), которая допускает бутстрэппинг одновременно по нескольким временным интервалам. Но здесь мы будем рассматривать только саму идею n -шагового бутстрэппинга, отложив разговор о механизме следов приемлемости на потом. Это позволит лучше разделить проблемы и обсудить как можно большее их число в более простой n -шаговой постановке.

Как обычно, сначала рассматривается задача предсказания, а затем задача управления. То есть сначала мы рассмотрим, как n -шаговые методы можно использовать для предсказания дохода в виде функции состояния для фиксированной стратегии (т. е. для оценивания v_π). А затем обобщим эти идеи на ценности действий и методы управления.

7.1. n -шаговое TD-предсказание

Что представляет собой пространство методов между методами Монте-Карло и TD-методами? Рассмотрим оценивание v_π по выборке эпизодов, сгенерированных с помощью стратегии π . Методы Монте-Карло выполняют обновление для

каждого состояния, основываясь на всей последовательности наблюдаемых вознаграждений от данного состояния до конца эпизода. С другой стороны, обновление в одношаговых TD-методах основывается только на одном следующем вознаграждении – с помощью бутстрэппинга ценность этого состояния используется на следующем шаге как заменитель оставшихся вознаграждений. Но тогда можно представить себе один из видов промежуточных методов: обновление производится на основе промежуточного количества вознаграждений – больше одного, но меньше, чем осталось до завершения эпизода. Например, двухшаговое обновление будет основываться на первых двух вознаграждениях и оценке ценности состояния двумя шагами позже. Аналогично возможно трехшаговое обновление, четырехшаговое обновление и т. д. На рис. 7.1 показаны диаграммы предшествующих состояний для всего спектра n -шаговых методов обновления v_π от одношагового TD-обновления слева до обновления в методе Монте-Карло, основанного на вознаграждениях до конца эпизода, справа.

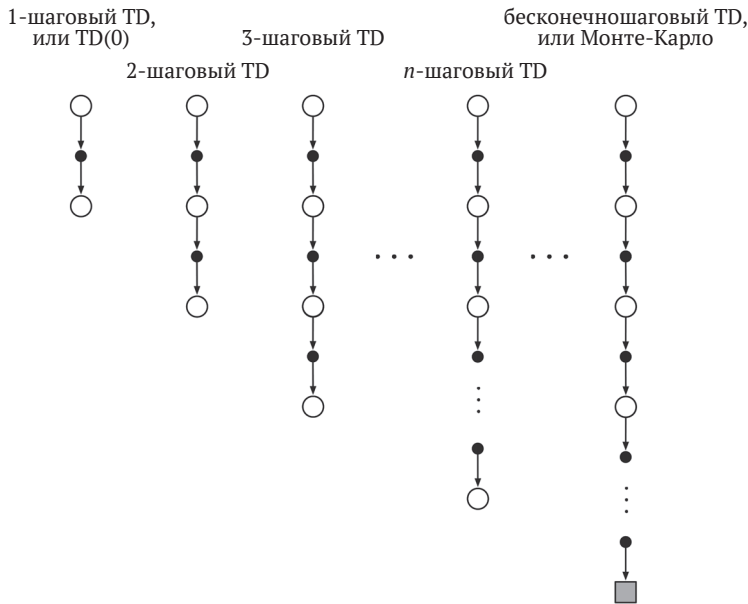


Рис. 7.1 ❖ Диаграммы предшествующих состояний для n -шаговых методов. Эти методы образуют спектр вариантов от одношаговых TD-методов до методов Монте-Карло

Методы, в которых используется n -шаговое обновление, по-прежнему являются TD-методами, поскольку они все-таки изменяют более раннюю оценку в зависимости от того, как она отличается от более поздней. Но теперь более поздняя оценка отстоит не на один, а на n шагов дальше. Методы, в которых временное различие распространяется на n шагов, называются n -шаговыми TD-методами. Во всех TD-методах, описанных в предыдущей главе, применялись одношаговые обновления, поэтому мы и называли их *одношаговыми TD-методами*.

Формально, рассмотрим обновление оценки ценности состояния S_t как результат последовательности состояние–вознаграждение $S_t, R_{t+1}, S_{t+1}, R_{t+2}, \dots, R_T, S_T$

(действия опущены). Мы знаем, что в методах Монте-Карло оценка $v_\pi(S_t)$ обновляется в направлении значения полного дохода:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{T-t-1} R_T,$$

где T – последний временной шаг эпизода. Назовем эту величину *целью* обновления. В методах Монте-Карло целью обновления является доход, а в одношаговых методах – первое вознаграждение плюс обесцененная оценка ценности следующего состояния (будем называть эту величину *одношаговым доходом*):

$$G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1}),$$

где $V_t: \mathcal{S} \rightarrow \mathbb{R}$ здесь является оценкой v_π в момент t . Нижние индексы в $G_{t:t+1}$ означают, что это усеченный доход в момент t , при вычислении которого используются только вознаграждения до момента $t + 1$ включительно, а остальные составляющие полного дохода $\gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{T-t-1} R_T$ заменены обесцененной оценкой $\gamma V_t(S_{t+1})$, как описано в предыдущей главе. Суть в том, что эта идея сохраняет смысл и после двух шагов, а не только одного. Целью двухшагового обновления является *двухшаговый доход*:

$$G_{t:t+2} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+2}),$$

где $\gamma^2 V_{t+1}(S_{t+2})$ занимает место слагаемых $\gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots + \gamma^{T-t-1} R_T$. Аналогично целью произвольного n -шагового обновления является *n -шаговый доход*:

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}) \quad (7.1)$$

для всех n, t таких, что $n \geq 1$ и $0 \leq t < T - n$. Все n -шаговые доходы можно рассматривать как аппроксимации полного дохода с усечением после n шагов и поправкой, заключающейся в замене оставшихся слагаемых величиной $V_{t+n-1}(S_{t+n})$. Если $t + n \geq T$ (n -шаговый доход простирается до завершения эпизода или дальше), то все отсутствующие члены полагаются равными нулю, а n -шаговый доход по определению считается равным обыкновенному полному доходу ($G_{t:t+n} \doteq G_t$, если $t + n \geq T$).

Заметим, что n -шаговый доход для $n > 1$ включает будущие вознаграждения и состояния, которые еще недоступны в момент перехода от t к $t + 1$. Ни в каком реальном алгоритме n -шаговый доход нельзя использовать, пока он не увидел R_{t+n} и не вычислил V_{t+n-1} . Впервые эти величины становятся доступны в момент $t + n$. Таким образом, естественный алгоритм обучения ценности состояний с использованием n -шаговых доходов выглядит так:

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha[G_{t:t+n} - V_{t+n-1}(S_t)], \quad 0 \leq t < T, \quad (7.2)$$

тогда как ценности прочих состояний не изменяются: $V_{t+n}(s) = V_{t+n-1}(s)$ для всех $s \neq S_t$. Мы можем назвать этот алгоритм *n -шаговым TD-алгоритмом*. Заметим, что на первых $n - 1$ шагах любого эпизода не производится вообще никаких изменений. Чтобы компенсировать это обстоятельство, в конце каждого эпизода выполняется равное количество дополнительных обновлений, после завершения и до начала следующего эпизода. Полный псевдокод алгоритма приведен ниже.

Упражнение 7.1. В главе 6 мы отметили, что ошибку Монте-Карло можно записать в виде суммы TD-ошибок (6.6), если оценка ценности не изменяется от шага к шагу. Покажите, что n -шаговую ошибку в формуле (7.2) также можно записать в виде суммы TD-ошибок (опять-таки, если оценки ценности не изменяются). Это обобщает ранее полученный результат. $\square$

Упражнение 7.2 (требуется программирование). В случае n -шагового метода оценки ценности изменяются от шага к шагу, поэтому алгоритм, в котором используется сумма TD-ошибок (см. предыдущее упражнение) вместо ошибки в формуле (7.2), отличается от описанного выше. Как вы думаете, лучше он или хуже? Придумайте и запрограммируйте простой эксперимент, чтобы ответить на этот вопрос эмпирически. $\square$

n -шаговый TD-алгоритм для оценивания $V \approx v_\pi$

Вход: стратегия π

Параметры алгоритма: размер шага $\alpha \in (0, 1]$, целое положительное число n

Инициализировать $V(s)$ произвольным образом для всех $s \in \mathcal{S}$

Все операции сохранения и доступа (для S_t и R_t) принимают индекс по модулю $n + 1$

Повторять для каждого эпизода:

Инициализировать и сохранить $S_0 \neq$ заключительное состояние

$T \leftarrow \infty$

Повторять для $t = 0, 1, 2, \dots$:

Если $t < T$, то:

Предпринять действие в соответствии с $\pi(\cdot | S_t)$

Наблюдать и сохранить следующее вознаграждение как R_{t+1}

и следующее состояние как S_{t+1}

Если S_{t+1} – заключительное состояние, то $T \leftarrow t + 1$

$\tau \leftarrow t - n + 1$ (τ – момент времени, для которого обновляется оценка состояния)

Если $\tau \geq 0$:

$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$

Если $\tau + n < T$, то $G \leftarrow G + \gamma^n V(S_{\tau+n})$

($G_{\tau:\tau+n}$)

$V(S_\tau) \leftarrow V(S_\tau) + \alpha[G - V(S_\tau)]$

пока не $\tau = T - 1$

При вычислении n -шагового дохода используется функция ценности V_{t+n-1} для замещения отсутствующих вознаграждений после R_{t+n} . Важное свойство n -шагового дохода заключается в том, что его математическое ожидание гарантированно дает лучшую оценку v_π , чем V_{t+n-1} в худшем случае. Это означает, что наихудшая ошибка ожидаемого n -шагового дохода будет гарантированно меньше или равна γ^n , умноженному на наихудшую ошибку при использовании V_{t+n-1} :

$$\max_s |\mathbb{E}_\pi[G_{t:t+n} | S_t = s] - v_\pi(s)| \leq \gamma^n \max_s |V_{t+n-1}(s) - v_\pi(s)| \quad (7.3)$$

для всех $n \geq 1$. Это называется *свойством уменьшения ошибки* n -шагового дохода. Благодаря свойству уменьшения ошибки можно формально показать, что все n -шаговые TD-методы сходятся к правильным предсказаниям при соответствующих технических условиях. Таким образом, n -шаговые TD-методы образуют семейство эффективных методов, крайними членами которого являются одношаговые TD-методы и методы Монте-Карло.

Пример 7.1 (n -шаговые TD-методы в задаче о случайном блуждании). Рассмотрим применение n -шаговых TD-методов в задаче о случайном блуждании с 5 состояниями, описанной в примере 6.2. Предположим, что первый эпизод развивается прямо из центрального состояния C направо, проходит через состояния D и E и завершается справа с доходом 1. Напомним, что оценки ценностей всех состояний вначале имеют промежуточное значение $V(s) = 0.5$. В результате полученного опыта одношаговый метод изменит только оценку последнего состояния $V(E)$, которая станет ближе к 1, т. е. наблюдаемому доходу. С другой стороны, двухшаговый метод увеличит ценности двух состояний, предшествующих завершению: $V(D)$ и $V(E)$ станут ближе к 1. Трехшаговый метод, как и любой n -шаговый метод при $n > 2$, увеличит ценности всех трех посещенных состояний на одну и ту же величину, приблизив их к 1.

Какое значение n лучше? На рис. 7.2 показаны результаты простого эмпирического теста процесса случайного блуждания с 19 состояниями вместо 5 (и с результатом -1 слева, когда все ценности вначале равны 0). Мы будем ссылаться на этот пример на всем протяжении данной главы. Результаты приведены для n -шаговых TD-методов с различными значениями n и α . Мерой качества различных комбинаций параметров является корень из среднеквадратической ошибки между истинными значениями ценности и предсказаниями в конце эпизода, усредненной по всем 19 состояниям и затем по первым 10 эпизодам и 100 повторениям всего эксперимента (для всех комбинаций параметров брались одни и те же наборы шагов блуждания). Эта величина отложена по вертикальной оси. Заметим, что лучше всего оказались методы с промежуточным значением n . Это показывает, что n -шаговые методы как обобщение TD-методов и методов Монте-Карло потенциально могут давать лучшие результаты, чем оба предельных варианта.

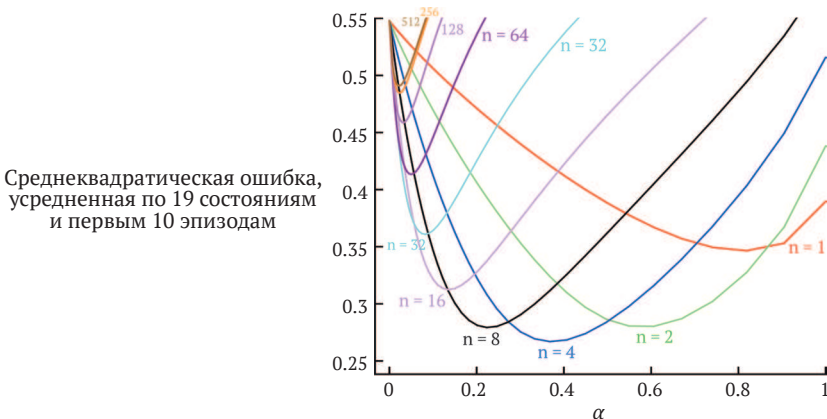


Рис. 7.2 ❖ Качество n -шаговых TD-методов в виде функции от α для различных значений n в задаче о случайном блуждании с 19 состояниями (пример 7.1)

Упражнение 7.3. Как вы думаете, почему в примерах из этой главы используется более масштабная задача о случайном блуждании (с 19, а не 5 состояниями)? Верно ли, что при меньшем блуждании более выгодным оказалось бы другое значение n ? А что, если в большем блуждании изменить результат левой части с 0 на -1 ? Повлияло бы это на выбор наилучшего значения? □

7.2. n -ШАГОВЫЙ АЛГОРИТМ SARSA

Как можно использовать n -шаговые методы не только для предсказания, но и для управления? В этом разделе мы покажем, как естественно объединить n -шаговые методы с алгоритмом Sarsa, получив тем самым TD-метод управления с единой стратегией. n -шаговый вариант Sarsa мы будем называть n -шаговым Sarsa, а первоначальный вариант, описанный в предыдущей главе, – *одношаговым Sarsa*, или *Sarsa(0)*.

Основная идея – просто заменить состояния на действия (пары состояние–действие), а затем воспользоваться ε -жадной стратегией. Диаграммы предшествующих состояний (см. рис. 7.3), как и в случае n -шагового TD (рис. 7.1), представляют собой чередующиеся состояния и действия, только для Sarsa все они начинаются и заканчиваются действием, а не состоянием. Переопределим n -шаговый доход (цели обновления) в терминах оценок ценностей действий:

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}), \quad n \geq 1, 0 \leq t < T - n, \quad (7.4)$$

где $G_{t:t+n} \doteq G_t$, если $t + n \geq T$. Тогда естественный алгоритм имеет вид

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha [G_{t:t+n} - Q_{t+n-1}(S_t, A_t)], \quad 0 \leq t < T, \quad (7.5)$$

а ценности всех прочих состояний не изменяются: $Q_{t+n}(s, a) = Q_{t+n-1}(s, a)$ для всех s, a таких, что $s \neq S_t$ или $a \neq A_t$. Этот алгоритм мы будем называть *n -шаговым Sarsa*. Его псевдокод приведен ниже, а пример, объясняющий, почему он может ускорять обучение по сравнению с одношаговыми методами, показан на рис. 7.4.

n -шаговый алгоритм Sarsa для оценивания $Q \approx q^*$ или q_π

Инициализировать $Q(s, a)$ произвольным образом для всех $s \in \mathcal{S}, a \in \mathcal{A}$
 Инициализировать π как ε -жадную стратегию относительно Q или как фиксированную заданную стратегию
 Параметры алгоритма: размер шага $\alpha \in (0, 1]$, небольшое $\varepsilon > 0$ и целое положительное число n
 Все операции сохранения и доступа (для S_t, A_t и R_t) принимают индекс по модулю $n + 1$

Повторять для каждого эпизода:

Инициализировать и сохранить $S_0 \neq \text{заключительное состояние}$
 Выбрать и сохранить действие $A_0 \sim \pi(\cdot | S_0)$
 $T \leftarrow \infty$

Повторять для $t = 0, 1, 2, \dots$:

Если $t < T$, то:

Предпринять действие A_t

Наблюдать и сохранить следующее вознаграждение как R_{t+1}

и следующее состояние как S_{t+1}

Если S_{t+1} – заключительное состояние, то:

$T \leftarrow t + 1$

иначе:

Выбрать и сохранить действие $A_{t+1} \sim \pi(\cdot | S_{t+1})$

$\tau \leftarrow t - n + 1$ (τ – момент времени, для которого обновляется оценка)

Если $\tau \geq 0$:

$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$

Если $\tau + n < T$, то $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ ($G_{\tau:\tau+n}$)

$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha[G - Q(S_\tau, A_\tau)]$

Если π обучается, то позаботиться о том, чтобы $\pi(\cdot | S_t)$ была

ε -жадной относительно Q

пока не $\tau = T - 1$

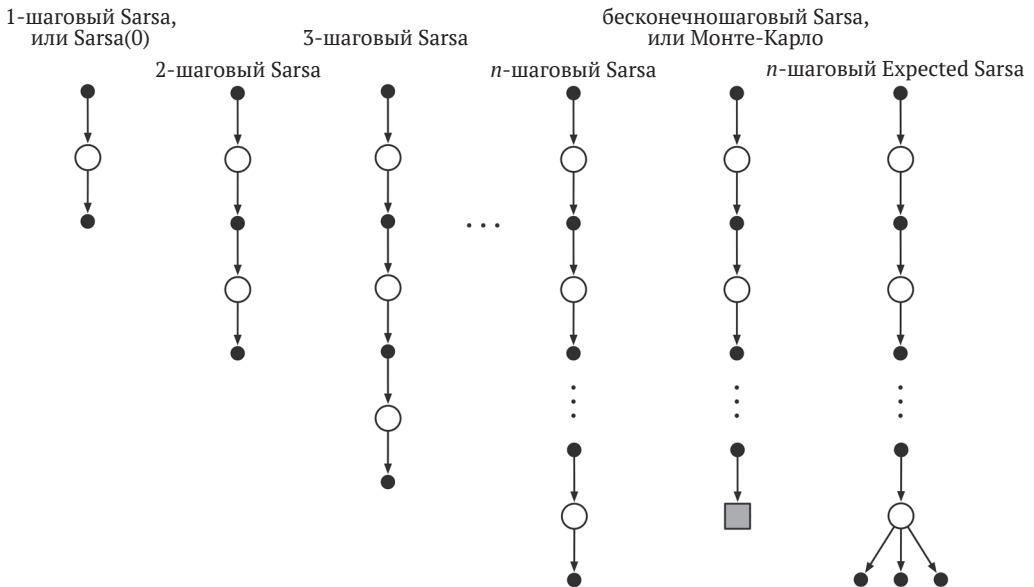


Рис. 7.3 ❖ Диаграммы предшествующих состояний для n -шаговых методов, применяемых к ценностям пар состояние–действие. Эти методы образуют спектр вариантов от одношагового обновления Sarsa(0) до методов Монте-Карло, где обновление производится до завершения эпизода. Посередине находятся n -шаговые обновления, основанные на n шагах реального вознаграждения и оценке ценности следующей пары состояние–действие, в которых к каждому слагаемому применяется подходящее обесценивание. Самая правая диаграмма соответствует n -шаговому алгоритму Expected Sarsa



Рис. 7.4 ❖ Сеточный мир, в котором применение n -шаговых методов ускоряет обучение стратегии. На первом рисунке показан путь, выбираемый агентом в одном эпизоде, который заканчивается в ячейке G с высоким вознаграждением. В этом примере ценности были инициализированы нулями, а все вознаграждения равны 0, кроме вознаграждения в ячейке G , которое положительно. Стрелки на двух других рисунках показывают, какие ценности действий были увеличены в результате выбора этого пути одношаговым и n -шаговым методами Sarsa. Одношаговый метод повышает только ценность последнего действия в последовательности, приведшей к высокому вознаграждению, а n -шаговый – ценности последних n действий, поэтому обучение в одном эпизоде оказывается гораздо эффективнее

Упражнение 7.4. Докажите, что n -шаговый доход в методе Sarsa (7.4) можно точно выразить в терминах новой TD-ошибки:

$$G_{t:t+n} = Q_{t-1}(S_t, A_t) + \sum_{k=t}^{\min(t+n, T)-1} \gamma^{k-t} [R_{k+1} + \gamma Q_k(S_{k+1}, A_{k+1}) - Q_{k-1}(S_k, A_k)]. \quad (7.6)$$

□

А как насчет Expected Sarsa? Диаграмма предшествующих состояний для n -шаговой версии Expected Sarsa показана справа на рис. 7.3. Она представляет собой линейную цепочку выборочных действий и состояний, как в n -шаговом Sarsa, но в самом конце находится ветвление по всем возможным действиям, как обычно с весами, равными их вероятностям при использовании стратегии π . Этот алгоритм можно описать тем же уравнением, что n -шаговый Sarsa (см. выше), с тем отличием, что n -шаговый доход определяется формулой

$$G_{t:t+n} \doteq R_{t+1} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n \bar{V}_{t+n-1}(S_{t+n}), \quad t+n < T \quad (7.7)$$

($G_{t:t+n} \doteq G_p$, если $t+n \geq T$), где $\bar{V}_t(s)$ – аппроксимация математического ожидания ценности состояния s , вычисленная с помощью оценок ценностей действий в момент t при целевой стратегии:

$$\bar{V}_t(s) \doteq \sum_a \pi(a|s) Q_t(s, a) \quad \text{для всех } s \in \mathcal{S}. \quad (7.8)$$

Аппроксимации математических ожиданий используются при разработке многих методов на основе ценности действий далее в этой книге. Если s – заключительное состояние, то его приближенная ожидаемая ценность по определению равна 0.

7.3. n -ШАГОВОЕ ОБУЧЕНИЕ С РАЗДЕЛЕННОЙ СТРАТЕГИЕЙ

Напомним, что обучением с разделенной стратегией называется обучение функции ценности для одной стратегии π , следуя другой стратегии b . Часто π является жадной стратегией относительно текущей оценки функции ценности действий, а b – в большей степени исследовательская стратегия, быть может, ε -жадная. Чтобы использовать данные от b , мы должны учесть различие между обеими стратегиями с помощью относительной вероятности предпринять те действия, которые были предприняты (см. раздел 5.5). В случае n -шаговых методов доход строится на протяжении n шагов, поэтому нас интересует относительная вероятность только этих n действий. Например, чтобы сконструировать версию n -шагового TD с разделенной стратегией, мы можем просто сопоставить обновлению для момента t (которое в действительности выполняется в момент $t + n$) вес $\rho_{t:t+n-1}$:

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha \rho_{t:t+n-1} [G_{t:t+n} - V_{t+n-1}(S_t)], \quad 0 \leq t < T, \quad (7.9)$$

где величина $\rho_{t:t+n-1}$, называемая *коэффициентом выборки по значимости*, – это относительная вероятность предпринять n действий от A_t до A_{t+n-1} при стратегиях π и b (см. формулу 5.3):

$$\rho_{t:h} \doteq \prod_{k=t}^{\min(h, T-1)} \frac{\pi(A_k | S_k)}{b(A_k | S_k)}. \quad (7.10)$$

Например, если какое-то действие никогда не выбирается стратегией π (т. е. $\pi(A_k | S_k) = 0$), то n -шаговому доходу следует приписать нулевой вес и полностью игнорировать его. С другой стороны, если предпринято действие, которое π выбрала бы с гораздо большей вероятностью, чем b , то это увеличит вес, который был бы приписан доходу в противном случае. И это разумно, поскольку действие – характеристика π (и, следовательно, мы хотим обучиться ему), но лишь изредка выбирается стратегией b и потому редко встречается в данных. Чтобы компенсировать это обстоятельство, мы должны придать ему дополнительный вес, когда наконец встретим его. Заметим, что если обе стратегии в действительности совпадают (случай единой стратегии), то коэффициент выборки по значимости равен 1. Поэтому формула обновления (7.9) обобщает и может полностью заменить прежнее обновление для n -шагового TD. Аналогично прежнее обновление для n -шагового Sarsa можно полностью заменить простой формулой для разделенной стратегии:

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha \rho_{t+1:t+n} [G_{t:t+n} - Q_{t+n-1}(S_t, A_t)] \quad (7.11)$$

для $0 \leq t < T$. Заметим, что здесь коэффициент выборки по значимости начинается и заканчивается на один шаг позже, чем в случае n -шагового TD (7.9). Это объясняется тем, что мы обновляем пару состояние–действие. Нам не нужно думать о том, с какой вероятностью следует выбирать действие; коль скоро оно выбрано, мы хотим максимально эффективно обучиться на том, что произошло, а по значимости выбирать только последующие действия. Полный псевдокод алгоритма приведен ниже.

***n*-шаговый алгоритм Sarsa с разделенной стратегией
для оценивания $Q \approx q_*$ или q_π**

Вход: произвольная поведенческая стратегия b такая, что $b(a|s) > 0$ для всех $s \in \mathcal{S}$, $a \in \mathcal{A}$
 Инициализировать $Q(s, a)$ произвольным образом для всех $s \in \mathcal{S}$, $a \in \mathcal{A}$
 Инициализировать π как ε -жадную стратегию относительно Q или как фиксированную заданную стратегию
 Параметры алгоритма: размер шага $\alpha \in (0, 1]$, целое положительное число n
 Все операции сохранения и доступа (для S_t , A_t и R_t) принимают индекс по модулю $n + 1$

Повторять для каждого эпизода:

Инициализировать и сохранить $S_0 \neq$ заключительное состояние

Выбрать и сохранить действие $A_0 \sim b(\cdot|S_0)$

$T \leftarrow \infty$

Повторять для $t = 0, 1, 2, \dots$:

Если $t < T$, то:

Предпринять действие A_t

Наблюдать и сохранить следующее вознаграждение как R_{t+1}

и следующее состояние как S_{t+1}

Если S_{t+1} – заключительное состояние, то:

$T \leftarrow t + 1$

иначе:

Выбрать и сохранить действие $A_{t+1} \sim b(\cdot|S_{t+1})$

$\tau \leftarrow t - n + 1$ (τ – момент времени, для которого обновляется оценка)

Если $\tau \geq 0$:

$$\rho \leftarrow \prod_{i=\tau+1}^{\min(\tau+n-1, T-1)} \frac{\pi(A_i|S_i)}{b(A_i|S_i)} \quad (\rho_{\tau+1:\tau+n-1})$$

$$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$$

$$\text{Если } \tau + n < T, \text{ то } G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n}) \quad (G_{\tau:\tau+n})$$

$$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha[G - Q(S_\tau, A_\tau)]$$

Если π обучается, то позаботиться о том, чтобы $\pi(\cdot|S_t)$ была ε -жадной относительно Q

пока не $\tau = T - 1$

В версии n -шагового Expected Sarsa с разделенной стратегией использовалось бы такое же обновление, как в n -шаговом Sarsa, с тем отличием, что в коэффициенте выборки по значимости было бы на один сомножитель меньше. То есть в приведенной выше формуле использовалось бы $\rho_{\tau+1:\tau+n-1}$ вместо $\rho_{\tau+1:\tau+n}$ и, конечно, вариант n -шагового дохода, определенный для Expected Sarsa (7.7). Это связано с тем, что в Expected Sarsa в последнем состоянии принимаются во внимание все возможные действия; какое из них действительно выбрано, ни на что не влияет, и вводить поправку не нужно.

7.4. *ПРИВЕДЕННЫЕ МЕТОДЫ С ПЕРЕМЕННЫМ УПРАВЛЕНИЕМ

Многошаговые методы с разделенной стратегией, представленные в предыдущем разделе, просты и концептуально прозрачны, но, пожалуй, они не самые эффективные. В более изощренном подходе следовало бы использовать идеи приведенной выборки по значимости, изложенные в разделе 5.9. Чтобы понять, как это делается, сначала заметим, что обыкновенный n -шаговый доход (7.1), как и любой доход, допускает рекуррентную запись. Для n шагов, заканчивающихся на горизонте h , n -шаговый доход можно записать в виде:

$$G_{t:h} = R_{t+1} + G_{t+1:h}, \quad t < h < T, \quad (7.12)$$

где $G_{h:h} \doteq V_{h-1}(S_h)$. (Напомним, что этот доход используется в момент h , который мы раньше обозначали $t + n$.) Теперь рассмотрим, что получится, если следовать поведенческой стратегии b , не совпадающей с целевой стратегией π . Весь полученный опыт, включая первое вознаграждение R_{t+1} и следующее состояние S_{t+1} , необходимо умножить на вес, равный коэффициенту выборки по значимости для момента t , $\rho_t = \pi(A_t | S_t) / b(A_t | S_t)$. Возникает соблазн просто взвесить правую часть приведенного выше уравнения, но можно поступить лучше. Предположим, что действие в момент t никогда не было бы выбрано стратегией π , так что ρ_t равно нулю. Тогда при простом взвешивании n -шаговый доход оказался бы нулевым, что могло бы привести к высокой дисперсии при использовании его в качестве цели. Поэтому в более хитроумном подходе используется альтернативное, с *разделенной стратегией* определение n -шагового дохода с горизонтом h , а именно:

$$G_{t:h} \doteq \rho_t(R_{t+1} + G_{t+1:h}) + (1 - \rho_t)V_{h-1}(S_t), \quad t < h < T, \quad (7.13)$$

где снова $G_{h:h} \doteq V_{h-1}(S_h)$. При таком подходе, если ρ_t равно нулю, цель не обращается в 0, что приводит к уменьшению оценки, а совпадает с оценкой, и никаких изменений не производится. Нулевой коэффициент выборки по значимости означает, что мы должны игнорировать выборку, поэтому решение оставить оценку без изменения выглядит разумным. Второй, дополнительный член в уравнении (7.13) по не вполне ясным причинам называется *переменным управлением* (control variate). Заметим, что переменное управление не изменяет математического ожидания обновления; ожидаемое значение коэффициента выборки по значимости равно 1 (раздел 5.9) и не коррелирует с оценкой, поэтому ожидаемое значение переменного управления равно 0. Отметим также, что определение с разделенной стратегией (7.13) является строгим обобщением данного ранее определения n -шагового дохода с единой стратегией (7.1), поскольку в случае единой стратегии, когда ρ_t равно 1, оба определения совпадают.

Для традиционного n -шагового метода правило обучения, используемое в сочетании с (7.13), – это обновление n -шагового TD-алгоритма (7.2), в котором нет других явных коэффициентов выборки по значимости, кроме тех, что включены в определение дохода.

Упражнение 7.5. Напишите псевдокод описанного выше алгоритма предсказания ценности состояния с разделенной стратегией. $\square$

Для ценностей действий определение n -шагового дохода с разделенной стратегией несколько отличается, потому что первое действие не играет никакой роли в выборке по значимости. Именно первое действие подлежит обучению; не важно, было ли оно маловероятным или вообще невозможным при следовании целевой стратегии, – оно было предпринято, и теперь последовавшим за ним вознаграждению и состоянию должен быть сопоставлен вес, равный 1. Выборка по значимости применяется только к последующим действиям.

Сначала заметим, что для ценностей действий формулу (7.7) для ожидаемого n -шагового дохода с горизонтом h и *единой стратегией* можно записать рекурсивно, как в (7.12), только для ценностей действий рекурсия завершается на $G_{h:h} \doteq \bar{V}_{h-1}(S_h)$, как в (7.8). Уравнение для разделенной стратегии с переменным управлением имеет вид

$$\begin{aligned} G_{t:h} &\doteq R_{t+1} + \gamma(\rho_{t+1}G_{t+1:h} + \bar{V}_{h-1}(S_{t+1}) - \rho_{t+1}Q_{h-1}(S_{t+1}, A_{t+1})) \\ &= R_{t+1} + \gamma\rho_{t+1}(G_{t+1:h} - Q_{h-1}(S_{t+1}, A_{t+1})) + \gamma\bar{V}_{h-1}(S_{t+1}), \quad t < h \leq T. \end{aligned} \quad (7.14)$$

Если $h \leq T$, то рекурсия завершается на $G_{h:h} \doteq Q_{h-1}(S_h, A_h)$, тогда как при $h > T$ рекурсия завершается на $G_{T-1:h} \doteq R_T$. Получающийся алгоритм предсказания (после объединения с (7.5)) аналогичен Expected Sarsa.

Упражнение 7.6. Докажите, что переменное управление в приведенных выше уравнениях не изменяет математического ожидания дохода. □

**Упражнение 7.7.* Напишите псевдокод описанного выше алгоритма предсказания ценности действий с разделенной стратегией. Особое внимание обратите на условия завершения рекурсии при достижении горизонта или конца эпизода. □

Упражнение 7.8. Покажите, что общий (с разделенной стратегией) вариант n -шагового дохода (7.13) все еще можно точно и компактно записать в виде суммы основанных на состоянии TD-ошибок (6.5), если приближенная функция ценности состояний не изменяется. □

Упражнение 7.9. Повторите предыдущее упражнение для варианта n -шагового дохода с разделенной стратегией для действий (7.14) и TD-ошибкой в смысле Expected Sarsa (величина в квадратных скобках в формуле 6.9). □

Упражнение 7.10 (требуется программирование). Придумайте не очень сложную задачу предсказания с разделенной стратегией и воспользуйтесь ей, чтобы показать, что алгоритм обучения с разделенной стратегией, в котором используются уравнения (7.13) и (7.2), эффективнее более простого алгоритма, в котором используются уравнения (7.1) и (7.9). □

Выборка по значимости, которой мы пользовались в этом и в предыдущем разделах, а также в главе 5, допускает эффективное обучение с разделенной стратегией, но в то же время приводит к обновлениям с большой дисперсией, вынуждая нас использовать небольшой шаг, что замедляет обучение. По-видимому, обучение с разделенной стратегией в принципе медленнее обучения с единой стратегией – в конце концов, данные в меньшей степени релевантны предмету обучения. Однако, по-видимому, правдой является и то, что эти методы можно улучшить. Переменное управление – один из способов уменьшения дисперсии.

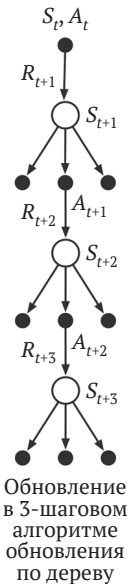
Другой способ – быстрая адаптация размера шага к наблюдаемой дисперсии, как в методе Autostep (Mahmood, Sutton, Degris and Pilarski, 2012). Еще один многообещающий подход – инвариантные обновления, описанные в работе Karampatziakis and Langford (2010) и обобщенные на TD в работе Тиана (Tian), готовящейся к публикации. Техника, предложенная Махмудом (Mahmood, 2017; Mahmood and Sutton, 2015), также может стать частью решения. В следующем разделе мы рассмотрим метод обучения с разделенной стратегией, в котором выборка по значимости не используется.

7.5. ОБУЧЕНИЕ С РАЗДЕЛЕННОЙ СТРАТЕГИЕЙ БЕЗ ВЫБОРКИ ПО ЗНАЧИМОСТИ: n -ШАГОВЫЙ АЛГОРИТМ ОБНОВЛЕНИЯ ПО ДЕРЕВУ

Возможно ли обучение с разделенной стратегией без выборки по значимости? Алгоритмы Q -обучения и Expected Sarsa из главы 6 решают эту задачу для одношагового случая, но существует ли соответствующий многошаговый алгоритм? В этом разделе мы опишем как раз такой n -шаговый метод, который называется *алгоритмом обновления по дереву* (tree-backup algorithm).

Идею подсказывает диаграмма предшествующих состояний 3-шагового алгоритма обновления по дереву, показанная справа. На центральном стволе расположены три выборочных состояния и вознаграждения и два выборочных действия. Это случайные величины, представляющие события, имевшие место после начальной пары состояние–действие S_t, A_t . По бокам от каждого состояния изображены действия, которые *не были* выбраны. (В случае последнего состояния все действия считаются (пока) невыбранными.) Поскольку мы не располагаем данными для невыбранных действий, производится бутстрэппинг, и оценки их ценностей используются для формирования цели обновления. Это небольшое обобщение идеи диаграммы предшествующих состояний. До сих пор мы всегда обновляли оценку ценности верхнего узла диаграммы в направлении цели, объединяющей полученные по пути вознаграждения (с соответствующим обесцениванием) и оценки ценностей нижних узлов. При обновлении в алгоритме обновления по дереву цель включает всё это и еще оценки ценностей боковых узлов действий на всех уровнях. Именно поэтому мы говорим об *обновлении по дереву*; это обновление на основе всего дерева оценок ценностей действий.

Если быть точным, то обновление производится на основе оценок ценностей действий в листовых узлах дерева. Внутренние узлы действий, соответствующие фактически предпринятым действиям, не принимают участия в обновлении. Каждый листовой узел входит в цель с весом, пропорциональным вероятности его выбора при целевой стратегии π . Таким образом, каждое действие a первого уровня входит с весом $\pi(a|S_{t+1})$, за исключением фактически выбранного действия A_{t+1} , которое не дает вообще никакого вклада. Его вероятность $\pi(A_{t+1}|S_{t+1})$ используется для вычисления весов всех ценностей действий второго уровня.



Следовательно, каждое невыбранное действие второго уровня a' входит с весом $\pi(A_{t+1}|S_{t+1})\pi(a'|S_{t+2})$. Каждое действие третьего уровня входит с весом $\pi(A_{t+1}|S_{t+1})\pi(a'|S_{t+2})\pi(a''|S_{t+3})$ и т. д. Можно сказать, что каждая стрелка, ведущая в какой-нибудь узел действия на диаграмме, снабжена весом, равным вероятности выбора этого действия при следовании целевой стратегии, а если из этого узла действия растет дерево, то этот вес применяется ко всем его листовым узлам.

Можно считать, что обновление в 3-шаговом алгоритме обновления по дереву состоит из 6 чередующихся полушагов: выборочных – из действия в последующее состояние – и ожидаемых – в которых рассматриваются все возможные действия в этом состоянии с вероятностями их выбора при данной стратегии.

А теперь выпишем детальные уравнения для n -шагового алгоритма обновления по дереву. Одношаговый доход (цель) – такой же, как в алгоритме Expected Sarsa:

$$G_{t:t+1} \doteq R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q_t(S_{t+1}, a) \quad (7.15)$$

для $t < T-1$, а двухшаговый доход имеет вид

$$\begin{aligned} G_{t:t+2} &\doteq R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+1}(S_{t+1}, a) \\ &\quad + \gamma \pi(A_{t+1}|S_{t+1}) \left(R_{t+2} + \gamma \sum_a \pi(a|S_{t+2}) Q_{t+1}(S_{t+2}, a) \right) \\ &= R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+1}(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:t+2} \end{aligned}$$

для $t < T-2$. Последняя формула наводит на мысль об общем рекурсивном определении n -шагового дохода в алгоритме обновления по дереву:

$$G_{t:t+n} \doteq R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+n-1}(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:t+n}, \quad t < T-1, \quad (7.16)$$

для $t < T-1$, $n \geq 2$, причем случай $n = 1$ учтен в формуле (7.15), при условии что $G_{T-1:t+n} \doteq R_T$. Эта цель затем используется в сочетании с обычным правилом обновления ценности действий из n -шагового Sarsa:

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha [G_{t:t+n} - Q_{t+n-1}(S_t, A_t)] \quad (7.5)$$

для $0 \leq t < T$, тогда как ценности всех остальных пар состояние–действие остаются без изменения: $Q_{t+n}(s, a) = Q_{t+n-1}(s, a)$ для всех s, a таких, что $s \neq S_t$ или $a \neq A_t$. Псевдокод этого алгоритма приведен ниже.

Упражнение 7.11. Покажите, что если приближенные ценности действий не изменяются, то доход в алгоритме обновления по дереву (7.16) можно записать в виде суммы TD-ошибок, основанных на математическом ожидании:

$$G_{t:t+n} = Q(S_t, A_t) + \sum_{k=t}^{\min(t+n-1, T-1)} \delta_k \prod_{i=t+1}^k \gamma \pi(A_i|S_i),$$

где $\delta_t \doteq R_{t+1} + \gamma \bar{V}_t(S_{t+1}) - Q(S_t, A_t)$, а $\bar{V}_t$ определено формулой (7.8). $\square$

n -шаговый алгоритм обновления по дереву для оценивания $Q \approx q_*$ или q_π

Инициализировать $Q(s, a)$ произвольным образом для всех $s \in \mathcal{S}$, $a \in \mathcal{A}$

Инициализировать π как жадную стратегию относительно Q или как фиксированную заданную стратегию

Параметры алгоритма: размер шага $\alpha \in (0, 1]$, целое положительное число n

Все операции сохранения и доступа принимают индекс по модулю $n + 1$

Повторять для каждого эпизода:

Инициализировать и сохранить $S_0 \neq$ заключительное состояние

Произвольным образом выбрать действие A_0 как функцию S_0 ; сохранить

A_0

$T \leftarrow \infty$

Повторять для $t = 0, 1, 2, \dots$:

Если $t < T$, то:

Предпринять действие A_t ; наблюдать и сохранить следующее вознаграждение и состояние как R_{t+1}, S_{t+1}

Если S_{t+1} – заключительное состояние, то:

$T \leftarrow t + 1$

иначе:

Произвольным образом выбрать действие A_{t+1} как функцию S_{t+1} ; сохранить A_{t+1}

$\tau \leftarrow t + 1 - n$ (τ – момент времени, для которого обновляется оценка)

Если $\tau \geq 0$:

Если $t + 1 \geq T$:

$G \leftarrow R_T$

иначе

$G \leftarrow R_{T+1} + \gamma \sum_a \pi(a | S_{t+1}) Q(S_{t+1}, a)$

Повторять с уменьшением для $k = \min(t, T - 1)$ до $\tau + 1$:

$G \leftarrow R_k + \gamma \sum_{a \neq A_k} \pi(a | S_k) Q(S_k, a) + \gamma \pi(A_k | S_k) G$

$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$

Если π обучается, то позаботиться о том, чтобы $\pi(\cdot | S_t)$ была жадной относительно Q

пока не $\tau = T - 1$

7.6. *Унифицированный алгоритм: n -шаговый $Q(\sigma)$

До сих пор в этой главе мы рассматривали три вида алгоритмов вычисления ценности действий, соответствующих первым трем диаграммам предшествующих состояний на рис. 7.5. В n -шаговом Sarsa все переходы были выборочными, в алгоритме обновления по дереву все переходы из состояния в действие полностью ветвились без какой-либо выборки, а в n -шаговом Expected Sarsa все переходы были выборочными, кроме последнего перехода из состояния в действие, который полностью ветвился, давая ожидаемую ценность. В какой мере эти алгоритмы поддаются унификации?

Одну идею унификации подсказывает четвертая диаграмма предшествующих состояний на рис. 7.5. Заключается она в том, что на каждом шаге можно принимать решение: предпринять выборочное действие, как в Sarsa, или рассмотреть математическое ожидание по всем действиям, как в алгоритме обновления по дереву. Тогда если всегда принимается решение о выборке, то мы получаем алгоритм Sarsa, а если такое решение не принимается никогда, то алгоритм обновления по дереву. Алгоритм Expected Sarsa мы получаем, когда решение о выборке принимается на всех шагах, кроме последнего.

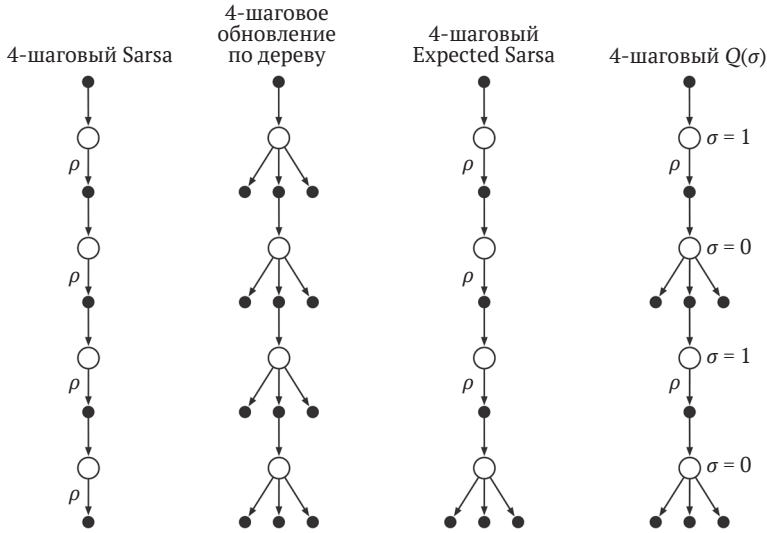


Рис. 7.5 ❖ Диаграммы предшествующих состояний для трех видов n -шагового обновления ценностей действий, рассмотренных выше в этой главе (4-шаговый случай), плюс диаграмма для четвертого, унифицированного вида обновления. Буквы ρ обозначают полупереходы, на которых в случае разделенной стратегии требуется выборка по значимости. Четвертый вид обновления объединяет все остальные за счет того, что в каждом состоянии принимается решение, производить выборку ($\sigma_t = 1$) или нет ($\sigma_t = 0$)

Конечно, существует много других возможностей, одна из которых показана на последней диаграмме. Чтобы еще расширить круг возможностей, можно рассмотреть непрерывный выбор между выборкой и математическим ожиданием. Обозначим $\sigma_t \in [0, 1]$ степень выборки на шаге t , причем $\sigma_t = 1$ означает чистую выборку, а $\sigma_t = 0$ – чистое математическое ожидание вообще без выборки. Случайная величина σ_t может быть задана в виде функции от состояния, действия или пары состояние–действие в момент t . Назовем этот новый алгоритм n -шаговым $Q(\sigma)$.

Теперь выпишем уравнения для n -шагового $Q(\sigma)$. Сначала выразим n -шаговый доход в алгоритме обновления по дереву (7.16) через горизонт $h = t + n$, а затем через приближенное математическое ожидание ценности $\bar{V}$ (7.8):

$$G_{t:h} = R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{h-1}(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:h}$$

$$\begin{aligned}
&= R_{t+1} + \gamma \bar{V}_{h-1}(S_{t+1}) - \gamma \pi(A_{t+1} | S_{t+1}) Q_{h-1}(S_{t+1}, A_{t+1}) + \gamma \pi(A_{t+1} | S_{t+1}) G_{t+1:h} \\
&= R_{t+1} + \gamma \pi(A_{t+1} | S_{t+1}) (G_{t+1:h} - Q_{h-1}(S_{t+1}, A_{t+1})) + \gamma \bar{V}_{h-1}(S_{t+1}),
\end{aligned}$$

после чего он выглядит в точности как n -шаговый доход для Sarsa с переменным управлением (7.14), только вместо коэффициента выборки по значимости ρ_{t+1} подставлена вероятность действия $\pi(A_{t+1} | S_{t+1})$. Для получения $Q(\sigma)$ мы осуществляем линейный переход между этими двумя случаями:

$$\begin{aligned}
G_{t:h} &\doteq R_{t+1} + \gamma (\sigma_{t+1} \rho_{t+1} + (1 - \sigma_{t+1}) \pi(A_{t+1} | S_{t+1})) (G_{t+1:h} - Q_{h-1}(S_{t+1}, A_{t+1})) \\
&\quad + \gamma \bar{V}_{h-1}(S_{t+1})
\end{aligned} \tag{7.17}$$

для $t < h \leq T$. Рекурсия заканчивается на $G_{h:h} \doteq Q_{h-1}(S_h, A_h)$, если $h < T$, или на $G_{T-1:T} \doteq R_T$, если $h = T$. Затем используется общее (с разделенной стратегией) обновление для n -шагового Sarsa (7.5). Во врезке приведен полный алгоритм.

n -шаговый алгоритм $Q(\sigma)$ с разделенной стратегией для оценивания $Q \approx q_*$ или q_π

Вход: произвольная поведенческая стратегия b такая, что $b(a | s) > 0$ для всех $s \in \mathcal{S}$, $a \in \mathcal{A}$

Инициализировать $Q(s, a)$ произвольным образом для всех $s \in \mathcal{S}$, $a \in \mathcal{A}$

Инициализировать π как ε -жадную стратегию относительно Q или как фиксированную заданную стратегию

Параметры алгоритма: размер шага $\alpha \in (0, 1]$, небольшое $\varepsilon > 0$ и целое положительное число n

Все операции сохранения и доступа принимают индекс по модулю $n + 1$

Повторять для каждого эпизода:

Инициализировать и сохранить $S_0 \neq$ заключительное состояние

Выбрать и сохранить действие $A_0 \sim b(\cdot | S_0)$

$T \leftarrow \infty$

Повторять для $t = 0, 1, 2, \dots$:

Если $t < T$, то:

Предпринять действие A_t ; наблюдать и сохранить следующее вознаграждение и состояние как R_{t+1}, S_{t+1}

Если S_{t+1} – заключительное состояние, то:

$T \leftarrow t + 1$

иначе:

Выбрать и сохранить действие $A_{t+1} \sim b(\cdot | S_{t+1})$

Выбрать и сохранить σ_{t+1}

$\tau \leftarrow t - n + 1$ (τ – момент времени, для которого обновляется оценка)

Если $\tau \geq 0$:

Если $t + 1 < T$:

$G \leftarrow Q(S_{t+1}, A_{t+1})$

Повторять с уменьшением для $k = \min(t + 1, T)$ до $\tau + 1$:

Если $k = T$:

$G \leftarrow R_T$

иначе

$$\bar{V} \leftarrow \sum_a \pi(a|S_k) Q(S_k, a)$$

$$G \leftarrow R_k + \gamma(\sigma_k \rho_k + (1 - \sigma_k) \pi(A_k|S_k)(G - Q(S_k, A_k) + \gamma \bar{V})$$

$$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha[G - Q(S_\tau, A_\tau)]$$

Если π обучается, то позаботиться о том, чтобы $\pi(\cdot|S_t)$ была жадной относительно Q

пока не $\tau = T - 1$

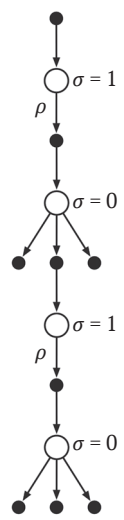
7.7. РЕЗЮМЕ

В этой главе мы разработали ряд методов обучения на основе временных различий, располагающихся между одношаговыми TD-методами из предыдущей главы и методами Монте-Карло из главы 5. Методы, включающие промежуточный объем бутстрэппинга, важны, потому что, как правило, работают лучше, чем крайние случаи.

Предметом этой главы являлись n -шаговые методы, которые заглядывают вперед на n вознаграждений, состояний и действий. На двух 4-шаговых диаграммах предшествующих действий справа показана квинтэссенция большинства рассмотренных методов. Обновление ценности состояний иллюстрируется на примере n -шагового TD с выборкой по значимости, а обновление ценности действий – на примере n -шагового $Q(\sigma)$, который обобщает алгоритмы Expected Sarsa и Q -обучения. Во всех n -шаговых методах имеется задержка на n временных шагов перед началом обновления, поскольку только после этого становятся известны все необходимые будущие события. Еще один недостаток – больший объем вычислений на каждом шаге, чем в ранее рассмотренных методах. По сравнению с одношаговыми методами, n -шаговые также потребляют больше памяти для запоминания состояний, действий, вознаграждений, а иногда и других переменных, относящихся к последним n временным шагам. В главе 12 мы покажем, как можно реализовать многошаговые TD-методы с минимальным потреблением памяти и вычислительной сложностью, применяя следы приемлемости, но все равно вычислений всегда будет больше, чем в одношаговом случае. Впрочем, эти затраты с лихвой окупаются избавлением от тирании единого временного шага.

Хотя n -шаговые методы сложнее тех, в которых используются следы приемлемости, у них есть важное достоинство – концептуальная ясность. Мы постарались воспользоваться им при разработке двух подходов к обучению с разделенной стратегией в n -шаговом случае. Один, основанный на выборке по значимости, концептуально прост, но может обладать высокой дисперсией. Если целевая и поведенческая стратегии сильно различаются, то, вероятно, понадобятся какие-то

4-шаговый TD

4-шаговый $Q(\sigma)$ 

новые идеи, чтобы сделать алгоритм эффективным и практически пригодным. Другой, основанный на обновлении по дереву, является естественным обобщением Q -обучения на многошаговый случай со стохастической целевой стратегией. Выборка по значимости в нем отсутствует, но если целевая стратегия сильно отличается от поведенческой, то бутстрэппинг может охватывать всего несколько шагов, даже если n велико.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

Понятие n -шагового дохода введено в работе Watkins (1989), где также впервые обсуждалось присущее ему свойство уменьшения ошибки. n -шаговые алгоритмы изучались в первом издании этой книги, где рассматривались как представляющие концептуальный интерес, но не реализуемые практически. Работы Cichosz (1995) и особенно van Seijen (2016) показали, что на самом деле эти алгоритмы вполне применимы на практике. С учетом этого, а также их концептуальной прозрачности и простоты мы решили подробнее остановиться на них во втором издании. В частности, мы теперь откладываем обсуждение обратного представления и следов приемлемости до главы 12.

7.1–2 Результаты в примерах случайного блуждания были подготовлены для этой книги на основе работ Sutton (1988) и Singh and Sutton (1996). Диаграммы предшествующих состояний для описания этих и других алгоритмов ранее не применялись.

7.3–5 Материал этих разделов основан на работах Precup, Sutton, and Singh (2000), Precup, Sutton, and Dasgupta (2001) и Sutton, Mahmood, Precup, and van Hasselt (2014).

Алгоритм обновления по дереву описан в работе Precup, Sutton, and Singh (2000), но здесь изложен по-новому.

7.6 Алгоритм $Q(\sigma)$ впервые представлен в этой книге, но тесно связанные с ним алгоритмы изучались впоследствии в работе De Asis, Hernandez-Garcia, Holland, and Sutton (2017).

Глава 8

Планирование и обучение табличными методами

В этой главе мы разовьем общий подход к методам обучения с подкреплением, которым требуется модель окружающей среды, таким как динамическое программирование и эвристический поиск, а также к методам, допускающим использование без модели, таким как методы Монте-Карло и методы на основе временных различий. Они называются соответственно методами *на основе модели* и *безмодельными*. Методы на основе модели полагаются на *планирование*, являющееся их главной компонентой, тогда как безмодельные методы – на *обучение*. Между двумя видами методов существуют как вполне осязаемые различия, так и общие черты. В частности, те и другие базируются на вычислении функции ценности. Кроме того, все методы разделяют общие принципы: заглядывание вперед для рассмотрения будущих событий, вычисление восстановленного значения и последующее использование его как цели обновления для приближенной функции ценности. Ранее в этой книге мы представили методы Монте-Карло и TD-методы как альтернативы, а затем показали, как их можно унифицировать с помощью *n*-шаговых методов. В этой главе нашей целью будет аналогичная унификация методов на основе модели и безмодельных методов. В предыдущих главах мы установили их различия, а теперь посмотрим, в какой мере их можно сочетать друг с другом.

8.1. Модели и планирование

Под *моделью* окружающей среды мы понимаем всё, что агент может использовать для предсказания реакции среды на свои действия. Зная состояние и действие, модель порождает предсказание следующего состояния и вознаграждения. Если модель стохастическая, то существует несколько возможных следующих состояний и вознаграждений, с каждым из которых связана вероятность. Некоторые модели порождают описание всех возможностей и их вероятностей; они называются *моделями распределения*. Другие модели порождают только одну из возможностей, выбранную согласно вероятности; они называются *моделями выборки*. Рассмотрим, к примеру, моделирование суммы очков, выпавших на двенадцати игровых костях. Модель распределения породила бы все возможные суммы вместе с вероятностями каждой, тогда как модель выборки – только одну сум-

му, выбранную в соответствии с распределением вероятностей. В динамическом программировании – оценивании динамики МППР, $p(s', r|s, a)$ – подразумевается модель распределения. А в примере игры в блэкджек из главы 5 используется модель выборки. Модели распределения более универсальны, чем модели выборки, поскольку их всегда можно использовать для порождения выборки. Но во многих приложениях гораздо проще получить модель выборки, чем модель распределения. Бросание нескольких костей как раз из числа таких приложений. Легко написать программу, которая моделирует бросок костей и возвращает выпавшую сумму, а вот перечислить все возможные суммы и их вероятности труднее и чревато ошибками.

Модели можно использовать для имитации опыта. Если известны начальное состояние и действие, то модель выборки порождает возможный переход, а модель распределения генерирует все возможные переходы, каждому из которых сопоставлен вес, определяемый его вероятностью. Если известны начальное состояние и стратегия, то модель выборки могла бы породить весь эпизод, а модель распределения – все возможные эпизоды и их вероятности. В любом случае мы говорим, что модель используется для *имитации* окружающей среды и порождения *имитированного опыта*.

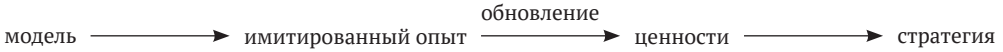
Слово «планирование» имеет различный смысл в разных областях. Мы понимаем под этим любой вычислительный процесс, который принимает на входе модель и порождает или улучшает стратегию взаимодействия с моделируемой средой.



В области искусственного интеллекта есть два подхода к так определенному планированию. *Планирование в пространстве состояний*, включающее описанный в этой книге подход, рассматривается в основном как поиск оптимальной стратегии или оптимального пути к цели в пространстве состояний. Действия вызывают переходы из одного состояния в другое, а функции ценности вычисляются для состояний. Имеется также *планирование в пространстве планов*, когда поиск производится в пространстве планов. Операторы преобразуют один план в другой, а функции ценности, если таковые существуют, определяются в пространстве планов. Планирование в пространстве планов включает эволюционные методы и «планирование с частичным порядком» – часто встречающийся в искусственном интеллекте вид планирования, при котором порядок шагов не полностью определен на всех этапах планирования. Методы планирования в пространстве планов не подходят для эффективного применения к стохастическим задачам последовательного принятия решений, к которым приковано внимание в обучении с подкреплением, поэтому в дальнейшем мы не будем их рассматривать (но см., например, Russell and Norvig, 2010).

Унифицированный подход, излагаемый в этой главе, заключается в том, что все методы планирования в пространстве состояний имеют одинаковую структуру, которая присутствует также в методах обучения, представленных в этой книге. Разработке данного подхода посвящен весь остаток этой главы, но основных идей всего две: (1) все методы планирования в пространстве состояний включают

вычисление функций ценности в качестве ключевого промежуточного шага в направлении улучшения стратегии и (2) они вычисляют функции ценности с помощью операций обновления, или восстановления, применяемых к имитированному опыту. Эту общую структуру можно схематически представить следующим образом:



Очевидно, что методы динамического программирования обладают такой структурой: они выполняют проходы по пространству состояний, генерируя для каждого состояния распределение возможных переходов. Затем каждое распределение используется для вычисления цели обновления и новой оценки ценности состояния. В этой главе будет показано, что и многие другие методы планирования в пространстве состояний также обладают этой структурой, а различаются только способами обновления, порядком их выполнения и сроками хранения информации, используемой для обновления.

Такой взгляд на методы планирования подчеркивает их связь с методами обучения, описываемыми в этой книге. В основе методов обучения и планирования лежит оценивание функций ценности с помощью операции обновления. А различие между ними заключается в том, что при планировании используется имитированный опыт, генерируемый моделью, а при обучении – реальный опыт, генерируемый окружающей средой. Конечно, из этого различия вытекает ряд других, например в способах оценивания эффективности и в том, насколько гибко может генерироваться опыт. Но наличие общей структуры означает, что многие идеи и алгоритмы переносятся с планирования на обучение и наоборот. В частности, во многих случаях вместо ключевого шага обновления в методе планирования можно подставить алгоритм обучения. На вход методов обучения необходимо подавать только опыт, и во многих случаях этот опыт может быть как реальным, так и имитированным. Во врезке ниже приведен пример простого метода планирования, основанного на одношаговом табличном Q-обучении и случайных выборках из модели выборки. Этот метод, который мы назовем *одношаговым табличным Q-планированием со случайной выборкой*, сходится к оптимальной стратегии для модели при тех же условиях, при которых одношаговое табличное Q-обучение сходится к оптимальной стратегии для реальной окружающей среды (каждая пара состояние–действие должна выбираться бесконечное число раз на шаге 1, и параметр α должен соответствующим образом убывать с течением времени).

Одношаговое табличное Q-планирование со случайной выборкой

Повторять бесконечно:

1. Выбрать случайным образом состояние $S \in \mathcal{S}$ и действие $A \in \mathcal{A}(S)$
2. Передать S, A модели выборки и получить следующее выборочное вознаграждение R и следующее выборочное состояние S'
3. Применить одношаговое табличное Q-обучение к S, A, R, S' :

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$$

Помимо выработки единого подхода к методам планирования и обучения, в этой главе рассматривается еще одна тема: преимущества планирования мелкими шагами. Это позволяет прерывать планирование или изменять его направление в любой момент с небольшими потерями на лишние вычисления, что является ключевым требованием к эффективному сочетанию планирования с исполнением и с обучением модели. Планирование очень мелкими шагами может оказаться наиболее эффективным подходом даже при решении задачи чистого планирования, если она слишком велика для точного решения.

8.2. Dyna: ОБЪЕДИНЕНИЕ ПЛАНИРОВАНИЯ, ИСПОЛНЕНИЯ И ОБУЧЕНИЯ

Если планирование производится в онлайн-режиме, в процессе взаимодействия с окружающей средой, то возникает ряд интересных вопросов. Новая информация, получаемая в ходе взаимодействия, может изменять модель и, стало быть, взаимодействовать с планированием. Иногда желательно тем или иным способом подстраивать процесс планирования к состояниям или решениям, анализируемым в данный момент или ожидаемым в ближайшем будущем. Если процессы принятия решений и обучения модели требуют большого объема вычислений, то располагаемые вычислительные ресурсы, возможно, придется разделить между ними. Перед тем как приступить к исследованию этих вопросов, мы в этом разделе представим Dyna-Q, простую архитектуру, объединяющую основные функции, необходимые агенту онлайн-планирования. Все функции в Dyna-Q имеют простую, почти тривиальную форму. В последующих разделах мы подробно рассмотрим некоторые альтернативные способы реализации каждой функции и компромиссы между ними. А пока мы хотим лишь проиллюстрировать идеи и выработать относящиеся к ним интуитивные представления.

Внутри агента планирования у реального опыта есть по меньшей мере две функции: он может использоваться для улучшения модели (чтобы она более точно отражала реальную окружающую среду) и для непосредственного улучшения функции ценности и стратегии с применением тех методов обучения с подкреплением, которые мы обсуждали в предшествующих главах. Первая функция называется *обучением модели*, вторая – *прямым обучением с подкреплением* (прямым ОП). Возможные взаимосвязи между опытом, моделью, ценностями и стратегией показаны на рисунке справа. Каждая стрелка обозначает связь между воздействием и предполагаемым улучшением. Обратите внимание, что опыт может улучшить функции ценности и стратегии как непосредственно, так и опосредованно с помощью модели. Последнее, иногда называемое *косвенным обучением с подкреплением*, является составной частью планирования.

У прямых и косвенных методов есть свои достоинства и недостатки. Косвенные методы зачастую более полно используют ограниченный объем опыта и потому



достигают лучшей стратегии при меньшем взаимодействии с окружающей средой. С другой стороны, прямые методы гораздо проще и не подвержены влиянию пристрастий при проектировании модели. Одни исследователи утверждают, что косвенные методы всегда превосходят прямые, другие – что на прямых методах основана большая часть обучения человека и животных. Аналогичные споры в области психологии и искусственного интеллекта касаются противопоставления процесса познания обучению методом проб и ошибок, а также противопоставления тщательного планирования и принятия решений в ответ на текущие изменения (см. обсуждение некоторых из этих вопросов с точки зрения психологии в главе 14). Мы полагаем, что различия между альтернативами в таких спорах излишне преувеличены и что лучшего понимания можно добиться, если сосредоточиться на сходствах методов, а не их различиях. Например, в этой книге мы подчеркивали глубокое сходство между методами динамического программирования и методами на основе временных различий, хотя первые были разработаны для планирования, а вторые – для безмодельного обучения.

Dyna-Q включает в себя все процессы, показанные на рисунке, – планирование, исполнение, обучение модели и прямое обучение с подкреплением (ОП), – причем все они протекают непрерывно. Метод планирования представляет собой одношаговый табличный метод Q-планирования, описанный на стр. 197. Метод прямого ОП – это одношаговое табличное Q-обучение. Метод обучения модели также табличный и предполагает, что окружающая среда детерминированна. После каждого перехода $S_t, A_t \rightarrow R_{t+1}, S_{t+1}$ модель запоминает в элементе таблицы, соответствующем паре S_t, A_t , предсказание о том, что за ним детерминированно следует пара R_{t+1}, S_{t+1} . Таким образом, если модель получает запрос о паре состояние–действие, которая уже наблюдалась ранее, то она просто возвращает в качестве предсказания последнее наблюдавшееся следующее состояние и следующее вознаграждение. Во время планирования алгоритм Q-планирования производит случайные выборки только из пар состояние–действие, которые наблюдались ранее (на шаге 1), поэтому модель никогда не получает запросов о парах, для которых у нее нет информации.

На рис. 8.1 показана общая архитектура агентов Dyna, примером которых (далеко не единственным) является алгоритм Dyna. В среднем столбце представлено основное взаимодействие между агентом и окружающей средой, результатом которого является траектория реального опыта. Левая стрелка на рисунке обозначает прямое обучение с подкреплением, в котором для улучшения функции ценности и стратегии используется реальный опыт. Справа показаны процессы на основе модели. Модель обучается на реальном опыте и порождает имитированный опыт. Мы употребляем термин *управление поиском* для обозначения процесса, который выбирает начальные состояния и действия для имитированного опыта, сгенерированного моделью. Наконец, планирование осуществляется путем применения методов обучения с подкреплением к имитированному опыту, как будто этот опыт действительно имел место. Как правило, в т. ч. в Dyna-Q, один и тот же метод обучения с подкреплением применяется как для обучения на реальном опыте, так и для планирования на основе имитированного опыта. Поэтому метод обучения с подкреплением является «общим инструментом» обучения и планирования. Обучение и планирование глубоко интегрированы в том смысле, что пользуются практически одними и теми же механизмами, а различаются только источником получения опыта.



Рис. 8.1 ❖ Общая архитектура Dyna. Реальный опыт, передаваемый между окружающей средой и стратегией, оказывает на стратегию и функции ценности практически такое же влияние, как имитированный опыт, сгенерированный моделью окружающей среды

Концептуально планирование, исполнение, обучение модели и прямое ОП происходят в Dyna-агентах одновременно и параллельно. Но, чтобы иметь возможность реализовать эти процессы на последовательном компьютере, мы должны точно определить их порядок в течение одного временного шага. В Dyna-Q процессы исполнения, обучения модели и прямого ОП требуют небольшого объема вычислений, поэтому предполагается, что они занимают небольшую долю времени. Оставшееся время на каждом шаге может быть потрачено на процесс планирования, который выполняет много вычислений (и тут ничего не поделаешь). Предположим, что на каждом шаге после исполнения, обучения модели и прямого ОП остается время для выполнения n итераций (шаги 1–3) алгоритма Q-планирования. В псевдокоде алгоритма Dyna-Q, приведенном во врезке ниже, $Model(s, a)$ обозначает содержание модели (предсказанное следующее состояние и вознаграждение) для пары состояние–действие (s, a) . Прямое обучение с подкреплением, обучение модели и планирование реализованы в шагах (d), (e) и (f) соответственно. Если опустить шаги (e) и (f), то получится алгоритм одношагового табличного Q-обучения.

Табличный Dyna-Q

Инициализировать $Q(s, a)$ и $Model(s, a)$ для всех $s \in \mathcal{S}$ и $a \in \mathcal{A}(S)$

Повторять бесконечно:

- (a) $S \leftarrow$ текущее (незаключительное) состояние
- (b) $A \leftarrow \varepsilon$ -жадная(S, Q)
- (c) Предпринять действие A ; наблюдать результирующее вознаграждение R и состояние S'
- (d) $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
- (e) $Model(S, A) \leftarrow R, S'$ (в предположении детерминированной среды)
- (f) Повторить n раз:
 - $S \leftarrow$ случайное ранее наблюдавшееся состояние
 - $A \leftarrow$ случайное действие, ранее предпринятое в состоянии S
 - $R, S' \leftarrow Model(S, A)$
 - $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$

Пример 8.1. Лабиринт Dyna. Рассмотрим простой лабиринт, изображенный на рис. 8.2. В каждом из 47 состояний возможны четыре действия: up, down, right и left, которые детерминированно переводят агента в соответствующее соседнее состояние, если только перемещению не мешает препятствие или граница лабиринта, а в этом случае агент остается на месте. Вознаграждение равно 0 за любое перемещение, кроме тех, которые ведут в целевое состояние, а в этом случае вознаграждение равно +1. Достигнув целевого состояния (), агент возвращается в начальное состояние (S), и начинается новый эпизод. Это эпизодическая задача с обесцениванием, в которой $\gamma = 0.95$.

В основной части рис. 8.2 изображены усредненные кривые обучения, полученные в эксперименте, в котором агенты Dyna-Q применялись к решению задачи о лабиринте. Начальные ценности действий были равны нулю, размер шага $\alpha = 0.1$, а параметр исследования $\varepsilon = 0.1$. При жадном выборе действий неоднозначности разрешались случайным образом. Агенты различались по количеству шагов планирования n , совершаемых на одном реальном шаге. Для каждого n кривые показывают число шагов, совершенных агентом для достижения цели в каждом эпизоде, усредненное по 30 повторениям эксперимента. При каждом повторении начальное значение для генератора случайных чисел оставалось одним и тем же для всех алгоритмов. Поэтому первый эпизод в точности повторяется (содержит примерно 1700 шагов) для всех значений n , и его данные не показаны на рисунке. После первого эпизода результаты улучшаются для всех значений n , но особенно быстро для больших. Напомним, что при $n = 0$ агент ничего не планирует, т. е. использует только прямое обучение с подкреплением (одношаговое табличное Q-обучение). В этой задаче такой агент является самым медленным, несмотря на то что параметры (α и ε) были выбраны для него оптимально. Непланирующему агенту потребовалось примерно 25 эпизодов для достижения ε -оптимального качества, тогда как агенту с $n = 5$ хватило пяти эпизодов, а агенту с $n = 50$ – всего трех эпизодов.

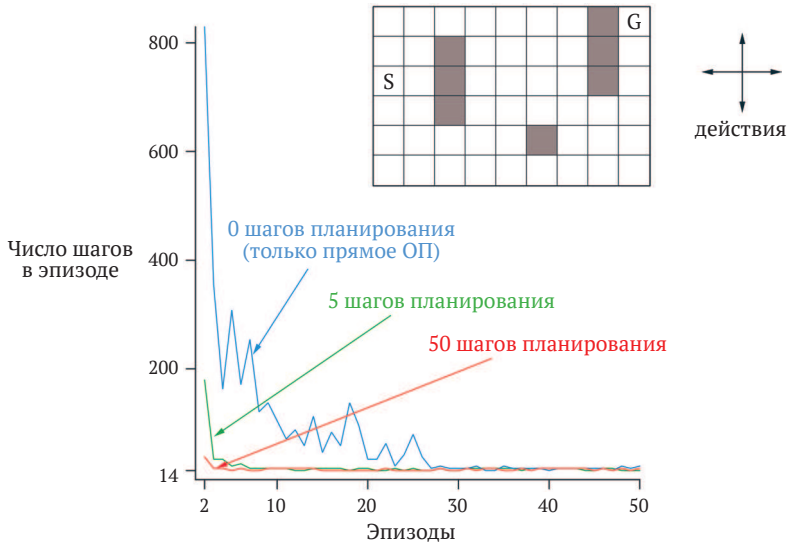


Рис. 8.2 ❖ Простой лабиринт и усредненные кривые обучения для агентов Dyna-Q с различным количеством шагов планирования (n) на одном реальном шаге. Задача заключается в том, чтобы как можно быстрее перейти из S в G

По рис. 8.3 видно, почему планирующие агенты находили решение настолько быстрее, чем непланирующие. Показаны стратегии, найденные агентами с $n = 0$ и $n = 50$ к середине второго эпизода. Без планирования ($n = 0$) каждый эпизод добавляет к стратегии лишь один дополнительный шаг, и, следовательно, агент обучался только одному (последнему) шагу. При наличии планирования в первом эпизоде агент также обучался только одному шагу, но во втором эпизоде была выработана более сложная стратегия, которая к концу эпизода почти достигнет начального состояния. Эта стратегия построена процессом планирования, в то время как агент все еще блуждает в окрестности начального состояния. К концу третьего эпизода будет найдена оптимальная стратегия и достигнуто идеальное качество.

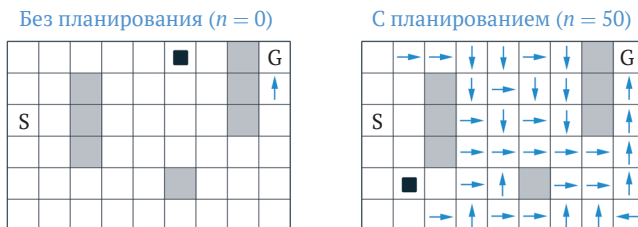


Рис. 8.3 ❖ Стратегии, найденные планирующими и непланирующими агентами Dyna-Q к середине второго эпизода. Стрелки показывают жадные действия в каждом состоянии; если для некоторого состояния не показана ни одна стрелка, то ценности всех действий в нем одинаковы. Черным квадратиком обозначено местоположение агента

В Dyna-Q обучение и планирование реализованы одним и тем же алгоритмом, который использует реальный опыт для обучения и имитированный для планирования. Поскольку планирование производится пошагово, чередовать планирование и исполнение очень просто. Оба процесса протекают настолько быстро, насколько позволяет алгоритм. Агент всегда готов скорректировать свое поведение и мгновенно реагирует на поступающую с датчиков информацию, но при этом постоянно продолжает планирование в фоновом режиме. Также в фоновом режиме протекает процесс обучения модели. При поступлении новой информации модель корректируется, чтобы лучше соответствовать реальным условиям. По мере изменения модели протекающий процесс планирования постепенно вырабатывает новую линию поведения, отвечающую новой модели.

Упражнение 8.1. На рис. 8.3 непланирующий метод выглядит особенно плохо, поскольку является одношаговым; метод с многошаговым бутстрэппингом работал бы лучше. Как вы думаете, может ли один из методов с многошаговым бутстрэппингом, рассмотренных в главе 7, показать такие же хорошие результаты, как метод Dyna? Объясните свой ответ. □

8.3. Когда модель НЕВЕРНА

В примере лабиринта, рассмотренном в предыдущем разделе, изменения модели были сравнительно небольшими. Вначале модель была пуста, а затем заполнена абсолютно точной информацией. В общем случае ожидать такой удачи не приходится. Модель может оказаться неправильной либо из-за того, что окружающая среда стохастическая и наблюдалось лишь ограниченное количество примеров, либо из-за того, что при обучении модели использовалась приближенная функция, которая плохо обобщается, либо просто из-за того, что среда изменилась и ее новое поведение еще не изучено. Если модель неправильна, то процесс планирования, скорее всего, будет вычислять неоптимальную стратегию.

В некоторых случаях вычисленная процессом планирования неоптимальная стратегия быстро приводит к обнаружению и исправлению ошибок моделирования. Обычно это происходит, когда модель оптимистичная в том смысле, что предсказывается большее вознаграждение или лучшие переходы состояний, чем возможно на самом деле. Спланированная стратегия пытается воспользоваться этими возможностями и по ходу дела обнаруживает, что их нет в природе.

Пример 8.2. Заблокированный лабиринт. На рис. 8.4 показан пример лабиринта, иллюстрирующий эту сравнительно незначительную ошибку моделирования и способ ее исправления. Изначально существует короткий путь от старта к цели справа от барьера, показанный на левом верхнем рисунке. После 1000 временных шагов короткий путь оказывается «заблокированным», но открывается более длинный путь слева от барьера, показанный на правом верхнем рисунке. На графике показано среднее накопленное вознаграждение для агента Dyna-Q и улучшенного агента Dyna-Q+, который будет описан ниже. Из первой части графика видно, что оба Dyna-агента нашли короткий путь за 1000 шагов. Когда окружающая среда изменилась, на графиках появилось горизонтальное плато, соответствующее периоду, в течение которого агенты не получали никакого вознаграждения.

дения, потому что находились за барьером. Однако спустя некоторое время они сумели найти новый проход и новую оптимальную линию поведения.

Более серьезные трудности возникают, когда в процессе изменения окружающая среда становится лучше, чем прежде, но выработанная ранее правильная стратегия не замечает этих улучшений. В такой ситуации ошибка моделирования может быть обнаружена спустя длительное время, а то и вовсе не обнаруживается.

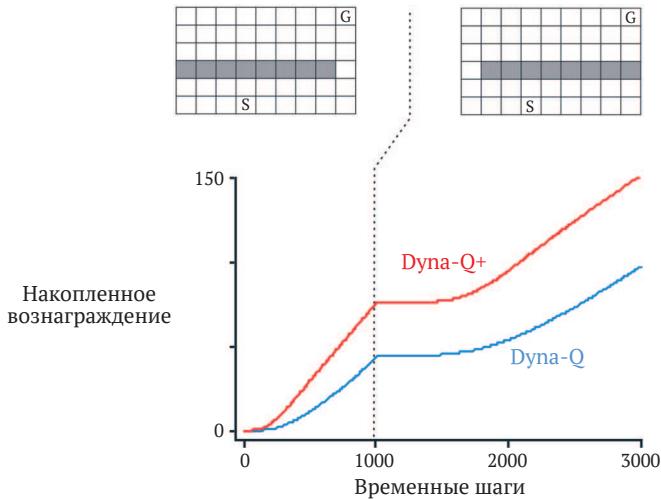


Рис. 8.4 ❖ Усредненные результаты Dyna-агентов в задаче о заблокированном лабиринте. На протяжении первых 1000 шагов использовалась левая окружающая среда, а затем правая. Dyna-Q+ – это агент Dyna-Q с призом, поощряющим исследование

Пример 8.3. Короткий путь в лабиринте. Проблема, вызванная этим типом изменения окружающей среды, иллюстрируется примером лабиринта на рис. 8.5. Изначально оптимальный путь проходит слева от барьера (левый верхний рисунок). Но после 3000 шагов открывается более короткий путь справа от барьера, при этом длинный путь тоже остается открытым (правый верхний рисунок). Из графика видно, что обычный агент Dyna-Q так ни разу и не воспользовался коротким путем. На самом деле он даже не понял, что такой путь существует. Его модель сообщала, что нет никакого короткого пути, поэтому чем дольше он планировал, тем меньше становилась вероятность сделать шаг вправо и обнаружить этот путь. Даже при ϵ -жадной стратегии крайне маловероятно, что агент будет предпринимать так много исследовательских действий, что обнаружит короткий путь.

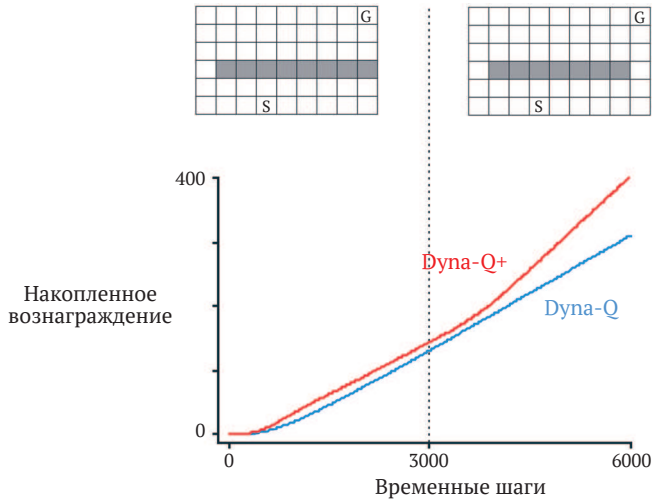


Рис. 8.5 ❖ Усредненные результаты Dyna-агентов в задаче о коротком пути в лабиринте. На протяжении первых 3000 шагов использовалась левая окружающая среда, а затем правая

В данном случае общая проблема заключается еще в одном варианте конфликта между исследованием и использованием. В контексте планирования исследование означает, что нужно пытаться выбирать действия, улучшающие модель, а использование – что нужно выбирать оптимальную линию поведения в рамках текущей модели. Мы хотим, чтобы агент занимался исследованием и обнаруживал изменения в окружающей среде, но не настолько интенсивно, чтобы результаты заметно ухудшились. Как и в предыдущем конфликте, вероятно, не существует решения, которое одновременно было бы идеальным и практически применимым, но простые эвристики часто оказываются эффективными.

Агент Dyna-Q+, который нашел короткий путь в лабиринте, использует одну из таких эвристик. Для каждой пары состояние–действие этот агент запоминает, сколько временных шагов прошло с момента последнего опробования этой пары в реальном взаимодействии со средой. Чем больше прошло времени, тем больше (предположительно) шансов, что динамика этой пары изменилась и что ее модель уже неверна. Чтобы побудить агента совершать давно не проверявшиеся действия, назначается специальное «бонусное вознаграждение» за имитированный опыт, приводящий к таким действиям. В частности, если смоделированное вознаграждение за переход равно r и этот переход не выполнялся в течение t временных шагов, то производится обновление планирования, как если бы за данный переход полагалось вознаграждение $r + \kappa\sqrt{t}$ при некотором небольшом значении κ . Это побуждает агента продолжать проверку всех доступных переходов состояний и даже находить длинные последовательности действий с целью осуществить такие проверки¹. Ко-

¹ В агент Dyna-Q+ внесено еще два изменения. Во-первых, действия, которые раньше никогда не проверялись в некотором состоянии, разрешено рассматривать на шаге планирования (f) табличного алгоритма Dyna-Q, приведенного во врезке выше. Во-вторых, начальная модель таких действий подразумевала, что они должны осуществлять переход в то же самое состояние с нулевым вознаграждением.

нечно, эти проверки имеют свою цену, но во многих ситуациях, в т. ч. в задаче о коротком пути в лабиринте, такое «любопытство» к дополнительному исследованию с лихвой окупается.

Упражнение 8.2. По какой причине Dyna-агент с призом за исследование, Dyna-Q+, показывает лучшие результаты и в первой, и во второй частях экспериментов с блокированием и коротким путем? □

Упражнение 8.3. Если внимательно посмотреть на рис. 8.5, то можно заметить, что различие между агентами Dyna-Q+ и Dyna-Q немного меньше в первой части эксперимента. Чем это можно объяснить? □

Упражнение 8.4 (требуется программирование). Описанный выше приз за исследование фактически изменяет оценки ценностей состояний и действий. Так ли это необходимо? Предположим, что приз $\kappa\sqrt{t}$ используется не при обновлении, а только при выборе действий, т. е. что всегда выбирается такое действие, для которого $Q(S_t, a) + \kappa\sqrt{t(S_t, a)}$ достигает максимума. Проведите эксперимент в сеточном мире, который демонстрирует сильные и слабые стороны такого подхода. □

Упражнение 8.5. Как можно было бы изменить табличный алгоритм Dyna-Q на стр. 201, чтобы он был применим к стохастической окружающей среде? Почему такая модификация могла бы показывать плохие результаты в изменяющейся окружающей среде типа рассмотренной в этом разделе? Как можно было бы изменить алгоритм, чтобы он хорошо работал и в стохастической, и в изменяющейся среде? □

8.4. ПРИОРИТЕТНЫЙ ПРОХОД

Для Dyna-агентов, представленных в предыдущих разделах, имитированные переходы начинаются в парах состояние–действие, которые случайно и равномерно выбираются из всех уже наблюдавшихся пар. Но равномерная выборка обычно не самая лучшая; планирование может быть гораздо более эффективным, если имитированные переходы и обновления сосредоточены на конкретных парах состояние–действие. Например, рассмотрим, что происходит во втором эпизоде первой задачи о лабиринте (рис. 8.3). В начале этого эпизода положительную ценность имеет только пара состояние–действие, которая непосредственно ведет в целевую ячейку; ценности остальных пар пока равны 0. Это означает, что почти для всех переходов обновление не имеет смысла, поскольку они переводят агента из одного состояния с нулевой ценностью в другое такое же состояние, а следовательно, обновление не возымеет никакого эффекта. И лишь обновление при переходе в состояние, непосредственно предшествующее цели, или при переходе из этого состояния изменяет какие-то значения ценностей. Если имитированные переходы генерируются равномерно, то, прежде чем мы наткнемся на один из этих полезных переходов, будет произведено много бессмысленных операций обновления. При дальнейшем планировании область полезных обновлений будет расти, но пла-

нирование все равно гораздо менее эффективно, чем если бы было сконцентрировано на тех областях, где может принести максимальную пользу. В крупномасштабных задачах, которые и представляют для нас основной интерес, количество состояний настолько велико, что хаотичный поиск будет крайне неэффективен.

Этот пример показывает, что поиск стоит производить в направлении *от* целевых состояний. Конечно, мы не хотели бы придумывать какие-то методы специально под идею «целевого состояния». Мы хотим, чтобы методы работали для общих функций вознаграждения. Целевые состояния – это просто частный случай, помогающий лучше понять проблему на интуитивном уровне. В общем случае необходим процесс, работающий не только в направлении от целевых состояний, но и от любого состояния, ценность которого была изменена. Предположим, что в начальный момент ценности соответствуют имеющейся модели, как в задаче о лабиринте, до того как была обнаружена цель. Теперь предположим, что агент обнаружил изменение в окружающей среде и изменил оценку ценности одного состояния – увеличил или уменьшил. Обычно это означает, что нужно изменить ценности многих других состояний, но полезными одношаговыми обновлениями будут только обновления действия, которые ведут непосредственно в то единственное состояние, ценность которого была изменена. Если ценности этих действий обновлены, то, в свою очередь, могут измениться ценности предшествующих состояний. В таком случае должны быть обновлены ценности ведущих в них действий, а стало быть, изменены ценности предшествующих *им* состояний. Таким образом, мы можем двигаться обратно из любых состояний с изменившейся ценностью, либо производя полезные обновления, либо завершив процесс распространения. Эту общую идею можно назвать *обратной фокусировкой* планирования.

Распространяясь в обратном направлении, фронт полезных обновлений зачастую быстро расширяется, порождая много пар состояние–действие, которые могли бы быть с пользой обновлены. Но не все они одинаково полезны. Ценности одних состояний могут измениться значительно, а других – лишь ненамного. Те пары, которые предшествуют сильно измененным состояниям, с высокой вероятностью также будут сильно изменены. В стохастической окружающей среде колебания оценок вероятностей переходов также вносят вклад в разброс величин изменений и в определение порядка обновления пар. Естественно было бы задавать приоритеты обновлений в соответствии с их срочностью и затем выполнять обновления в порядке приоритета. Эта идея лежит в основе *приоритетного прохода*. Создается очередь пар состояние–действие, для которых оценка ценности существенно изменилась бы при обновлении. После обновления пары, стоящей в начале очереди, вычисляется его влияние на все предшествующие пары. Если эффект превосходит некоторый небольшой порог, то данная пара помещается в очередь с новым приоритетом (если эта пара уже присутствует в очереди, то в результате вставки в очереди остается только пара с большим приоритетом). Таким образом, влияние изменений распространяется в обратном направлении до полного затухания. Полный псевдокод алгоритма для случая детерминированной окружающей среды приведен во врезке ниже.

Приоритетный проход для детерминированной окружающей среды

Инициализировать $Q(s, a)$ и $Model(s, a)$ для всех s, a и пустую очередь $PQueue$
 Повторять бесконечно:

- (a) $S \leftarrow$ текущее (незаключительное) состояние
- (b) $A \leftarrow \text{стратегия}(S, Q)$
- (c) Предпринять действие A ; наблюдать результирующее вознаграждение R и состояние S'
- (d) $Model(S, A) \leftarrow R, S'$
- (e) $P \leftarrow |R + \gamma \max_a Q(S', a) - Q(S, A)|$
- (f) если $P > \theta$, то поместить S, A в очередь $PQueue$ с приоритетом P
- (g) Повторить n раз, пока $PQueue$ не пуста:

$S, A \leftarrow \text{first}(PQueue)$

$R, S' \leftarrow Model(S, A)$

$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$

Повторять для всех $\bar{S}, \bar{A}$, которые, согласно предсказанию, ведут в S :

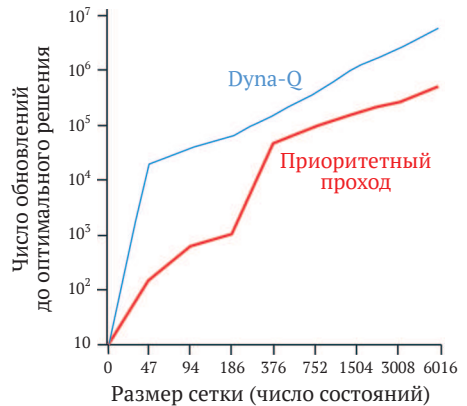
$\bar{R} \leftarrow$ предсказанное вознаграждение для $\bar{S}, \bar{A}, S$

$P \leftarrow |R + \gamma \max_a Q(S, a) - Q(\bar{S}, \bar{A})|$

если $P > \theta$, то поместить $\bar{S}, \bar{A}$ в очередь $PQueue$ с приоритетом P

Пример 8.4. Приоритетный проход в лабиринтах.

Обнаружилось, что приоритетный проход значительно увеличивает скорость нахождения оптимальных решений в задачах о лабиринтах, зачастую в 5–10 раз. Типичный пример показан на рисунке справа. Эти данные относятся к последовательности задач о лабиринтах, имеющих в точности такую же структуру, как задача на рис. 8.2, с тем отличием, что разрешение сетки меняется. Приоритетный проход имеет безусловное преимущество над неприоритетным алгоритмом Dyna-Q. Обоим системам потребовалось не более $n = 5$ обновлений за один акт взаимодействия с окружающей средой. Приводится с разрешения авторов работы Peng and Williams (1993).



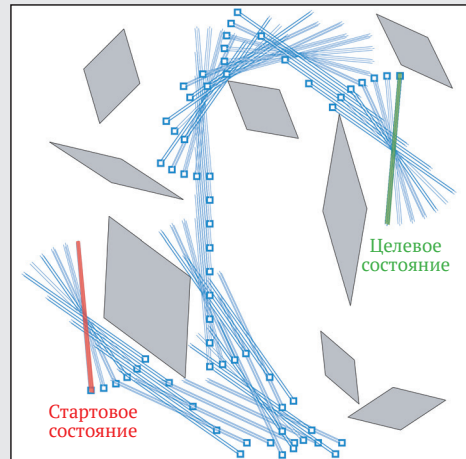
Обобщение идеи приоритетного прохода на стохастические окружающие среды не составляет труда. Для поддержания модели в актуальном состоянии хранятся две величины: сколько раз наблюдалась каждая пара состояние–действие и каковы при этом были следующие состояния. Далее естественно обновлять каждую пару не выборочно, как мы делали до сих пор, а полностью, вычисляя математическое ожидание с учетом всех возможных следующих состояний и вероятностей их возникновения.

Приоритетный проход – это лишь один из способов распределить вычисления для повышения эффективности планирования, и, пожалуй, даже не лучший. Одно из его ограничений заключается в использовании *полного* обновления, что в сто-

хастической среде может приводить к большому объему бесполезных вычислений для маловероятных переходов. В следующем разделе мы покажем, что выборочное обновление во многих случаях может дать гораздо лучшее приближение к истинной функции ценности, несмотря даже на дисперсию вследствие выборки. Выборочное обновление оказывается лучше, потому что позволяет разбить полное вычисление обновления на меньшие части, соответствующие отдельным переходам, а значит, уделить больше внимания тем частям, которые дают наибольший эффект. Эта идея была доведена до логического завершения в методе «малых обновлений», описанном в работе van Seijen and Sutton (2013). Имеются в виду обновления на одном переходе, как в случае выборочного обновления, но основанные на вероятности этого перехода без осуществления выборки, как при полном обновлении. Выбирая порядок осуществления небольших обновлений, можно добиться эффективности планирования, намного превосходящей ту, что дает приоритетный проход.

Пример 8.5. Приоритетный проход в задаче о перемещении стержня

Задача заключается в том, чтобы провести стержень между препятствиями, произвольно расположенными в ограниченном прямоугольнике пространстве, доставив его с целевого положения за наименьшее количество шагов. Стержень можно перемещать параллельно или перпендикулярно его оси, а также вращать в любом направлении относительно центра. Длина каждого перемещения составляет примерно $1/20$ часть рабочего пространства, а угол поворота равен 10 градусам. Перемещения стержня детерминированы и квантованы на 20×20 позиций. На рисунке показаны препятствия и кратчайший путь от старта до цели, найденный методом приоритетного прохода. Данная задача является детерминированной, но в ней 4 действия и 14 400 потенциальных состояний (некоторые из них недостижимы из-за расположений препятствий). Задача слишком велика для решения непериприоритетными методами. Рисунок перепечатан из статьи Moore and Atkeson (1993).



В этой главе мы высказали мысль о том, что все варианты планирования в пространстве состояний можно рассматривать как последовательности обновления ценностей, различающиеся только типом обновления, выборочное или полное, его масштабом и порядком выполнения обновлений. В этом разделе мы сосредоточились на обратной фокусировке, но это лишь одна из возможных стратегий. Можно было бы вместо этого классифицировать состояния, исходя из того, насколько легко их можно достичь из состояний, которые часто посещаются при текущей стратегии, – этот подход можно было бы назвать *прямой фокусировкой*. В работах Peng and Williams (1993) и Barto, Bradtke and Singh (1995) исследованы различные варианты прямой фокусировки, а методы, описанные в следующих нескольких разделах, доводят ее до логического предела.

8.5. СРАВНЕНИЕ ВЫБОРОЧНОГО И ПОЛНОГО ОБНОВЛЕНИЙ

Примеры из предыдущих разделов дают некоторое представление о разнообразии возможностей комбинирования методов обучения и планирования. Далее в этой главе мы проанализируем некоторые сопутствующие идеи, начиная с относительных преимуществ полного и выборочного обновлений.

Большая часть этой книги посвящена различным видам обновления функций ценности, и мы уже видели много их разновидностей. Рассмотрим одношаговые обновления. Они различаются в основном тремя бинарными факторами. Первые два фактора описывают, что обновляется – ценности состояний или ценности действий, и производится ли оценка ценности для оптимальной или для произвольно заданной стратегии. Эти два фактора определяют четыре класса обновлений для аппроксимации четырех функций ценности: q_* , v_* , q_π и v_π . Третий бинарный фактор описывает, является ли обновление *полным*, т. е. учитывающим все возможные события, или *выборочным*, т. е. учитывающим только одну выборку возможных событий. Три бинарных фактора определяют восемь случаев, семь из которых соответствуют конкретным алгоритмам, показанным на рис. 8.6. (Восьмому случаю, похоже, не соответствует никакое полезное обновление.) Любое из этих одношаговых обновлений можно использовать в методах планирования. В рассмотренных выше агентах Dyna-Q используются выборочные обновления q_* , но с тем же успехом можно было бы использовать полные обновления q_* либо выборочные или полные обновления q_π . В системе Dyna-AC используются выборочные обновления v_π в сочетании со структурой стратегии обучения (как в главе 13). Для стохастических задач всегда производится приоритетный проход с использованием какого-либо вида полного обновления.

Знакомясь в главе 6 с одношаговыми выборочными обновлениями, мы представляли их как замену полным обновлениям. В отсутствие модели распределения полное обновление невозможно, а выборочное можно произвести с помощью выборочных переходов из модели окружающей среды или модели выборки. Неявно подразумевалось, что полное обновление, если оно возможно, предпочтительнее выборочного. Но так ли это? Полные обновления, безусловно, дают лучшие оценки, поскольку не искажаются ошибкой выборки, но они также требуют большего объема вычислений, а вычислительные ресурсы на этапе планирования зачастую ограничены. Чтобы адекватно оценить относительные преимущества полного и выборочного обновлений применительно к планированию, необходимо учитывать различные требования к объему вычислений.

Для определенности рассмотрим полные и выборочные обновления для аппроксимации функции q_* , а также частный случай дискретных состояний и действий, табличное представление приближенной функции ценности Q и модель в форме оценки динамики $\hat{p}(s', r | s, a)$. Полное обновление для пары состояние–действие s, a имеет вид:

$$Q(s, a) \leftarrow \sum_{s', r} \hat{p}(s', r | s, a) \left[r + \gamma \max_{a'} Q(s', a') \right]. \quad (9.1)$$

Соответствующее выборочное обновление для s, a при заданных выборочном следующем состоянии и вознаграждении, S' и R (из модели), представляет собой обновление, похожее на выполняемое в алгоритме Q -обучения:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [R + \gamma \max_{a'} Q(S', a') - Q(s, a)], \quad (9.2)$$

где α – как всегда, положительный размер шага.

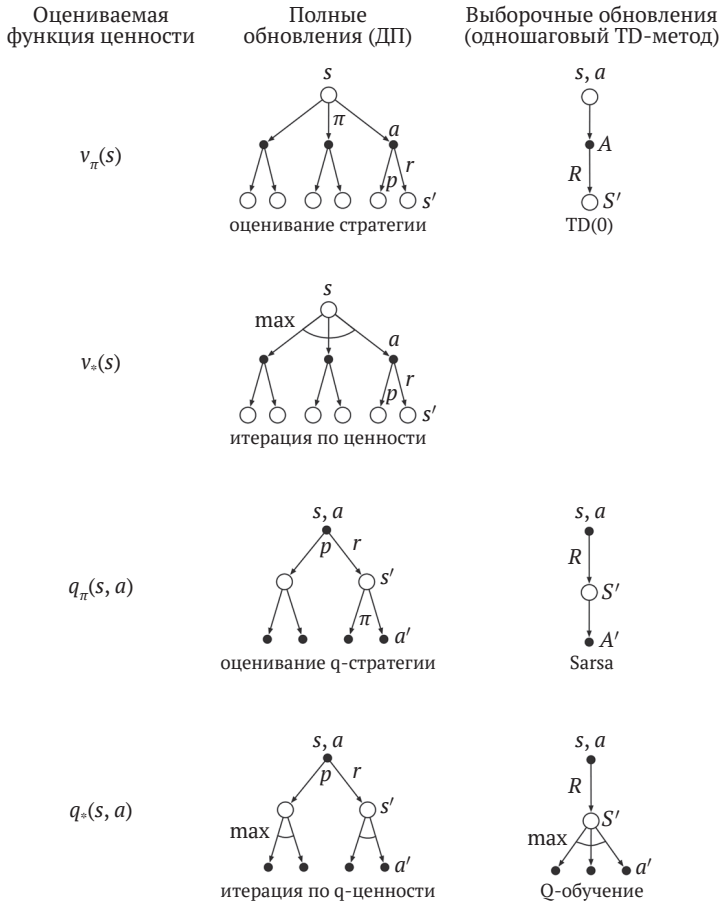


Рис. 8.6 ❖ Диаграммы предшествующих состояний для всех одношаговых обновлений, рассматриваемых в этой книге

Различие между полным и выборочным обновлениями существенно, когда окружающая среда стохастическая, а точнее обусловлено тем, что при заданном состоянии и действии может существовать много следующих состояний с различными вероятностями. Если возможно только одно следующее состояние, то полное и выборочное обновления, показанные выше, совпадают (при $\alpha = 1$). Если следующих состояний может быть много, то различия могут оказаться значительными. В пользу полного обновления говорит тот факт, что это точное вычисление, результатом которого является новое значение $Q(s, a)$, точность которого ограничена лишь точностью значения $Q(s', a')$ в состояниях-преемниках. На выборочное обновление также оказывает влияние ошибка выборки. С другой стороны, выборочное обновление вычислительно проще, потому что в нем учитывается только

одно следующее состояние, а не все сразу. На практике объем вычислений при выполнении операций обновления обычно определяется, прежде всего, количеством пар состояние–действие, в которых оценивается функция Q . Для начальной пары s, a обозначим b коэффициент ветвления (т. е. число возможных состояний s' , для которых $\hat{p}(s'|s, a) > 0$). Тогда для полного обновления этой пары потребуется примерно в b раз больше вычислений, чем для выборочного обновления.

Если времени для полного обновления достаточно, то полученная оценка в общем случае будет лучше, чем в результате b выборочных обновлений, из-за отсутствия ошибки выборки. Но если времени недостаточно, то выборочное обновление предпочтительнее, потому что оно, по крайней мере, дает некоторое улучшение оценки ценности при количестве обновлений, меньшем b . В задаче с большим числом пар состояние–действие мы часто сталкиваемся именно со второй ситуацией. В этом случае обновление всех пар заняло бы слишком много времени. Гораздо эффективнее может оказаться применение небольшого числа обновлений ко многим парам состояние–действие, чем полное обновление небольшого числа пар. Если объем вычислительных затрат фиксирован, то на что эффективнее его потратить: на несколько полных обновлений или на в b раз большее количество выборочных обновлений?

На рис. 8.7 показаны результаты экспериментального анализа этого вопроса. Приведена зависимость ошибки оценки от времени вычислений для полного выборочного обновления с различными коэффициентами ветвления b . Предполагалось, что все b состояний-преемников равновероятны и что ошибка начальной оценки равна 1. Предполагается также, что ценности следующих состояний определены правильно, т. е. по завершении полного обновления ошибка уменьшается до нуля. В данном случае выборочное обновление уменьшает ошибку по закону $\sqrt{(b-1)bt}$, где t – количество произведенных выборочных обновлений (предполагается усреднение по выборке, т. е. $\alpha = 1/t$). Ключевое наблюдение заключается в том, что для умеренно больших b ошибка очень быстро уменьшается после выполнения числа обновлений, составляющего лишь малую часть b . При таких обстоятельствах ценность многих пар состояние–действие можно было бы значительно улучшить, так что она отличалась бы от результатов полного обновления всего на несколько процентов, и для этого пришлось бы затратить столько же времени, сколько занимает полное обновление всего одной пары состояние–действие.

Показанное на рис. 8.7 преимущество в реальных условиях, возможно, будет даже более значительным. В реальной задаче ценности состояний-преемников были бы оценками, которые сами подвергаются обновлению. Поскольку при выборочном обновлении оценки быстрее становятся более точными, у такого способа обновления обнаруживается второе преимущество: ценности, обновленные на основе последующих состояний, оказываются более точными. Эти результаты говорят о том, что выборочные обновления, скорее всего, превосходят полные в задачах с большим стохастическим коэффициентом ветвления и слишком большим числом состояний, чтобы их ценности можно было вычислить точно.

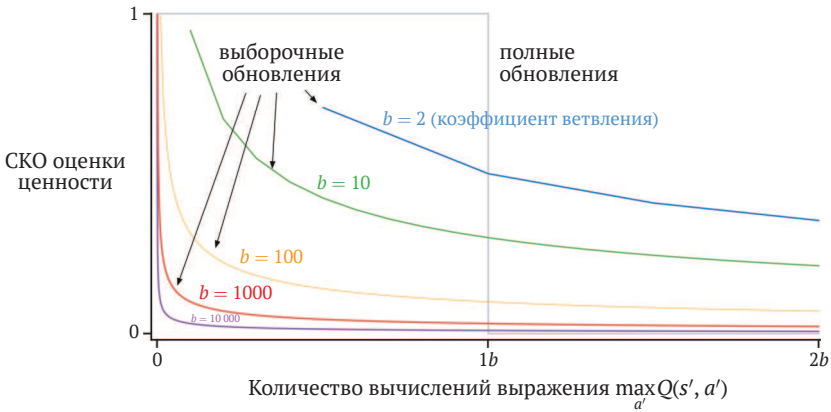


Рис. 8.7 ❖ Сравнение эффективности полного и выборочного обновлений

Упражнение 8.6. В приведенном выше анализе предполагалось, что все b возможных следующих состояний равновероятны. Предположим вместо этого, что распределение сильно несимметрично, т. е. некоторые из b состояний возникают с гораздо большей вероятностью, чем другие. При таком предположении увеличится или уменьшится преимущество выборочных обновлений над полными? Аргументируйте свой ответ. □

8.6. ТРАЕКТОРНАЯ ВЫБОРКА

В этом разделе мы сравним два способа распределения обновлений. Согласно классическому подходу, пришедшему из динамического программирования, проход осуществляется по всему пространству состояний (или пар состояние–действие), при этом каждое состояние (или пара состояние–действие) обновляется один раз за проход. Это может вызвать проблемы при решении больших задач, потому что может не хватить времени для завершения даже одного прохода. Во многих задачах подавляющее большинство состояний не представляет никакого интереса, поскольку они посещаются только при очень плохих стратегиях или с очень низкой вероятностью. Исчерпывающий проход уделяет одинаковое время всем частям пространства состояний, вместо того чтобы сосредоточиться на тех областях, где это необходимо. В главе 4 мы говорили о том, что исчерпывающий проход и свойственное ему одинаковое рассмотрение всех состояний не являются обязательными свойствами динамического программирования. Теоретически обновления могут иметь какое угодно распределение (для гарантии сходимости все состояния или пары состояние–действие в пределе должны посещаться бесконечное число раз, хотя в разделе 8.7 ниже обсуждается исключение из этого правила), но на практике часто производится исчерпывающий проход.

Согласно второму подходу, выборка из пространства состояний или пар состояние–действие производится в соответствии с некоторым распределением. Распределение может быть равномерным, как в случае агента Dyna-Q, однако это по-

рождает те же проблемы, что исчерпывающий проход. Заманчивее производить выборку из распределения с единой стратегией, т. е. того, которое наблюдается при следовании текущей стратегии. Одним из преимуществ такого распределения является простота его генерации; нужно лишь взаимодействовать с моделью, придерживаясь текущей стратегии. В эпизодической задаче процесс начинается в стартовом состоянии (или в состоянии, выбираемом из распределения стартовых состояний) и далее производится имитация вплоть до заключительного состояния. В непрерывной задаче процесс начинается в любом месте, и далее производится имитация. В обоих случаях выборочные переходы состояний и вознаграждения определяются моделью, а выборочные действия – текущей стратегией. Иными словами, происходит имитация отдельных траекторий и выполняются обновления всех встретившихся по пути состояний или пар состояние–действие. Такой способ генерации опыта и обновлений называется *траекторной выборкой*.

Сложно представить себе более эффективный способ распределения обновлений, соответствующий распределению с единой стратегией, чем траекторная выборка. Если бы имелось явное представление распределения с единой стратегией, то можно было бы пройтись по всем состояниям, назначив обновлению каждого вес, соответствующий этому распределению, но при этом снова возникает вопрос о вычислительных затратах, связанных с исчерпывающим проходом. Быть может, можно было бы выбирать из распределения с единой стратегией отдельные пары состояние–действие и обновлять их, но даже если бы мы смогли сделать это эффективно, какие преимущества мы получили бы по сравнению с имитацией траекторий? Да и получить распределение с единой стратегией в явной форме практически невозможно. Распределение меняется всякий раз, как изменяется стратегия, и для расчета распределения требуется выполнить объем вычислений, сравнимый с полным оцениванием стратегии. Таким образом, рассмотрев другие возможности, мы приходим к выводу, что траекторная выборка представляется эффективным и элегантным решением.

Является ли распределение обновлений с единой стратегией хорошим? Интуитивно кажется, что да, по крайней мере оно лучше равномерного распределения. Например, обучаясь играть в шахматы, вы изучаете позиции, которые могут возникнуть в реальных партиях, а не случайные расстановки шахматных фигур. Последние могут оказаться допустимыми позициями, но для точной оценки их ценности необходимы другие навыки, чем при оценке реальных позиций. В части II мы также увидим, что распределение с единой стратегией имеет существенные преимущества при использовании приближенных функций. Но вне зависимости от того, используется аппроксимация функций или нет, можно ожидать, что применение распределения с единой стратегией значительно увеличит скорость планирования.

Использование распределения с единой стратегией может быть выгодно, потому что при этом игнорируются обширные области пространства состояний, не представляющие интереса. Но оно может оказаться и вредным, поскольку приводит к многократному обновлению одних и тех же частей пространства. Для эмпирической оценки этого эффекта мы провели эксперимент. Чтобы изолировать влияние распределения обновлений, мы ограничились только лишь одношаговыми полными табличными обновлениями, определенными формулой (8.1). В *равномерном* случае мы перебирали все пары состояние–действие, обновляя каждую

на месте, а в случае единой стратегии имитировались эпизоды, начинающиеся в одном и том же состоянии, в которых обновлению подвергалась каждая пара состояние–действие, возникающая при следовании ε -жадной стратегии ($\varepsilon = 0.1$). Эпизодические задачи без обесценивания случайно генерировались следующим образом. В каждом из $|\mathcal{S}|$ состояний было возможно два действия, каждое из которых с равной вероятностью приводило к одному из b следующих состояний, причем для каждой пары состояние–действие эти b состояний выбирались случайно. Коэффициент ветвления b был одинаковым для всех пар состояние–действие. Кроме того, из каждого состояния с вероятностью 0.1 вел переход в заключительное состояние, завершающее эпизод. Ожидаемое вознаграждение на каждом переходе выбиралось из нормального распределения со средним 0 и дисперсией 1. В любой момент процесс планирования можно было остановить и выполнить полное вычисление $v_{\pi}(s_0)$, истинной ценности стартового состояния при жадной стратегии π и известной текущей функции ценности действий Q . Это позволило бы узнать, насколько хорошо агент проявил бы себя в новом эпизоде, в котором он действовал бы жадным образом (мы все время предполагаем, что модель верна).

В левой части рис. 8.8 показаны результаты, усредненные по 200 выборочным задачам с 1000 состояний и коэффициентами ветвления 1, 3 и 10. Качество найденных стратегий представлено в виде функции от количества завершенных полных обновлений. Во всех случаях результатом выборки в соответствии с распределением с единой стратегией стало ускорение планирования вначале и замедление при длительной работе. При небольших коэффициентах эффект проявлялся сильнее, а начальный период ускоренного планирования длился дольше. В других экспериментах мы обнаружили, что эти эффекты усиливаются также при возрастании числа состояний. Например, в правой части рис. 8.8 показаны результаты для коэффициента ветвления 1 в задаче с 10 000 состояний. В данном случае преимущество использования единой стратегии заметнее и продолжительнее.

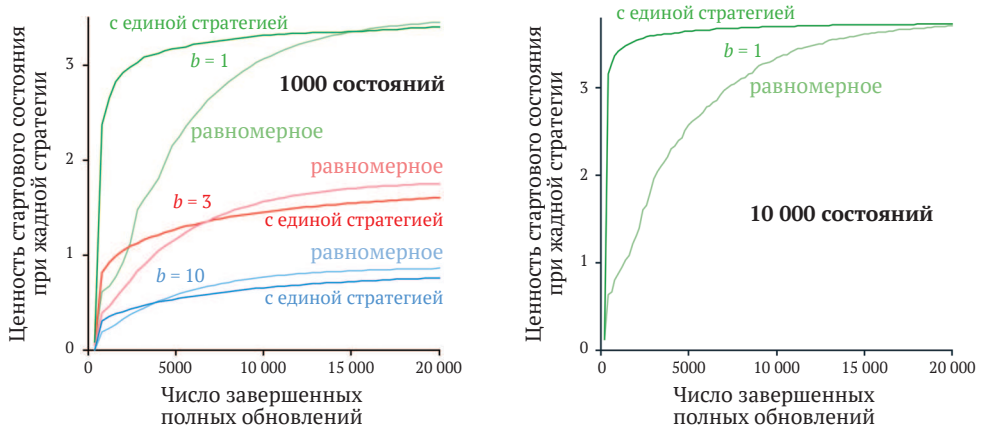


Рис. 8.8 ❖ Относительная эффективность обновлений, распределенных равномерно в пространстве состояний, и обновлений, в которых используются имитированные траектории с единой стратегией. Все эпизоды начинаются в одном и том же состоянии. Результаты соответствуют случайно сгенерированным задачам двух разных размеров с различными коэффициентами ветвления b

Все эти результаты ожидаемы. В краткосрочной перспективе выборка согласно распределению с единой стратегией помогает сосредоточиться на состояниях, не-далеко отстоящих от стартового. Если состояний много и коэффициент ветвления мал, то этот эффект будет значительным и продолжительным. В долгосрочной перспективе использование распределения с единой стратегией может нанести вред, поскольку для часто встречающихся состояний уже вычислены правильные ценности. Выборка таких состояний бесполезна, тогда как выборка других состояний может дать положительный эффект. Вероятно, именно по этой причине исчерпывающий бессистемный подход оказывается лучше в долгосрочной перспективе, по крайней мере для небольших задач. Эти результаты не являются окончательными, поскольку относятся лишь к задачам, сгенерированным конкретным случайным образом, однако указывают на то, что выборка в соответствии с распределением с единой стратегией может иметь заметное преимущество в больших задачах, особенно таких, где при использовании подобного распределения посещается лишь небольшое подмножество пространства состояние–действие.

Упражнение 8.7. Некоторые графики на рис. 8.8 имеют всплески и неровности на ранних участках, особенно это относится к верхнему графику для случая $b = 1$ и равномерного распределения. Как это можно объяснить? Какие аспекты представленных данных подтверждают вашу гипотезу? □

Упражнение 8.8 (требуется программирование). Повторите эксперимент, результаты которого представлены в нижней части рис. 8.8, а затем попробуйте выполнить его же со значением $b = 3$. Обсудите полученные результаты. □

8.7. ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ В РЕАЛЬНОМ ВРЕМЕНИ

Динамическое программирование в реальном времени, или ДПРВ, – это вариант алгоритма динамического программирования (ДП) с итерацией по ценности, в котором применяется траекторная выборка с единой стратегией. Будучи тесно связанным с традиционной итерацией по стратегиям на основе прохода по пространству состояний, ДПРВ особенно наглядно иллюстрирует некоторые преимущества траекторной выборки с единой стратегией. При использовании ДПРВ ценности состояний, посещенных на фактических или имитированных траекториях, обновляются посредством полного табличного обновления с итерацией по ценности, определенного уравнением (4.10). По сути дела, именно этот алгоритм порождал результаты для единой стратегии на рис. 8.8.

Тесная связь между ДПРВ и традиционным ДП позволяет получить некоторые теоретические результаты с помощью адаптации существующей теории. ДПРВ – пример асинхронного алгоритма ДП, описанного в разделе 4.5. Асинхронные алгоритмы ДП строятся не на основе систематического прохода по пространству состояний, они обновляют ценности состояний в произвольном порядке, используя уже вычисленные ценности других состояний. В ДПРВ порядок обновления определяется порядком, в котором посещаются состояния на реальных или имитированных траекториях.

Если траектории могут начинаться только в выделенном множестве стартовых состояний и нас интересует задача предсказания для данной стратегии, то траекторная выборка с единой стратегией позволяет алгоритму полностью пропускать состояния, недостижимые при следовании данной стратегии ни из одного стартового состояния; такие состояния *нерелевантны* задаче предсказания. В случае задачи управления, когда целью является нахождение оптимальной стратегии, а не оценивание данной стратегии, вполне могут существовать состояния, недостижимые из стартовых ни при какой оптимальной стратегии, и для таких нерелевантных состояний совершенно необязательно описывать оптимальные действия. В действительности нам необходима *оптимальная частичная стратегия*, т. е. стратегия, которая оптимальна для релевантных состояний, но может выбирать произвольные действия или даже быть вовсе неопределенной для нерелевантных.



Но для *нахождения* оптимальной частичной стратегии методом управления на основе траекторной выборки с единой стратегией, например Sarsa (раздел 6.4), в общем случае требуется бесконечное число раз посетить все пары состояние–действие, даже те, которые в итоге окажутся нерелевантными. Это можно сделать, например, применив исследовательские старты (раздел 5.3). Это справедливо также для ДПРВ: для эпизодических задач с исследовательскими стартами ДПРВ является асинхронным алгоритмом итерации по ценности, который сходится к оптимальным стратегиям для конечных МППР с обесцениванием (и для случая без обесценивания при некоторых условиях). В отличие от ситуации в задаче предсказания, вообще говоря, невозможно прекратить обновление какого-либо состояния или пары состояние–действие, если нам важна сходимость к оптимальной стратегии.

Самый интересный результат для ДПРВ заключается в том, что для некоторых типов задач, удовлетворяющих разумным условиям, ДПРВ гарантированно находит стратегию, оптимальную на релевантных состояниях, не посещая каждое состояние бесконечно часто, а иногда даже не посещая некоторые состояния вообще. Действительно, в некоторых задачах необходимо посетить лишь небольшую часть пространства состояний. Это может оказаться серьезным преимуществом в задачах с очень большим множеством состояний, когда даже один полный проход практически нереализуем.

Этот результат имеет место, в частности, для эпизодических задач без обесценивания для МППР с поглощающими целевыми состояниями, приносящими

нулевое вознаграждение (см. раздел 3.4). На каждом шаге реальной или имитированной траектории ДПРВ выбирает жадное действие (неоднозначности разрешаются произвольным образом) и применяет к текущему состоянию операцию полного обновления итерацией по ценности. Можно также на каждом шаге обновлять ценности произвольных наборов других состояний; например, можно обновлять ценности состояний, посещенных при поиске из текущего состояния с заглядыванием вперед на ограниченное число шагов.

В таких задачах для каждого эпизода, начинающегося в состоянии, случайно выбранном из множества стартовых состояний, и заканчивающегося в целевом состоянии, ДПРВ сходится с вероятностью 1 к стратегии, оптимальной для релевантных состояний, при следующих условиях: 1) начальная ценность каждого целевого состояния равна нулю; 2) существует по крайней мере одна стратегия, гарантирующая достижимость целевого состояния с вероятностью 1 из любого стартового состояния; 3) все вознаграждения за переходы из нецелевых состояний строго отрицательны; 4) начальные ценности всех состояний больше или равны их оптимальным ценностям (чтобы удовлетворить этому условию, можно просто положить начальные ценности всех состояний равными нулю). Этот результат был доказан в работе Barto, Bradtke, and Singh (1995) путем объединения результатов для асинхронного ДП с результатами, касающимися алгоритма эвристического поиска, который назван *обучением A^* в реальном времени* в работе Korf (1990).

Задачи, обладающие этими свойствами, являются примерами задач стохастической оптимизации, которые обычно формулируются в терминах минимизации затрат, а не максимизации вознаграждения, как здесь. Максимизация отрицательного дохода в нашем варианте эквивалентна минимизации стоимости пути из стартового состояния в целевое. Примерами могут служить задачи управления с минимальным временем, когда за каждый временной шаг на пути к достижению цели начисляется вознаграждение -1 , или задачи типа игры в гольф из раздела 3.5, где цель – загнать мяч в лунку за наименьшее число ударов.

Пример 8.6. ДПРВ на гоночной трассе. Задача о кольцевых гонках из упражнения 5.12 является задачей стохастической оптимизации. Сравнение ДПРВ с традиционным алгоритмом ДП методом итерации по ценности в этой задаче иллюстрирует некоторые преимущества траекторной выборки с единой стратегией.

Напомним, что в вышеупомянутом упражнении агент должен был обучиться проходить повороты типа показанного на рис. 5.5 и как можно скорее пересекать финишную черту, не вылетев за пределы трассы. Все стартовые состояния располагаются на стартовой линии, и скорость в них равна нулю; целевыми являются все состояния, из которых можно за один временной шаг пересечь финишную черту, оставшись на трассе. В отличие от упражнения 5.12, здесь нет ограничения на скорость машины, поэтому множество состояний потенциально бесконечно. Однако множество состояний, достижимых из множества стартовых состояний при любой стратегии, конечно и может рассматриваться как пространство состояний задачи. Каждый эпизод начинается в случайно выбранном стартовом состоянии и завершается, когда машина пересечет финишную черту. За каждый шаг, предшествующий пересечению финишной черты, начисляется вознаграждение -1 . Если машина задевает границу трассы, она возвращается в случайное стартовое состояние, и эпизод продолжается.

Гоночная трасса, изображенная в левой части рис. 5.5, имеет 9115 состояний, достижимых из стартовых состояний при какой-либо стратегии, но из них только 599 релевантны, т. е. достижимы из какого-то стартового состояния при какой-то оптимальной стратегии. (Для оценки количества релевантных состояний подсчитывались состояния, посещенные при выборе оптимальных действий в 10^7 эпизодах.)

В таблице ниже сравниваются решения этой задачи методами традиционного ДП и ДПРВ. Результаты усреднены по 25 прогонам, каждый из которых случайно генерировался с новым начальным значением. Методом традиционного ДП в данном случае являлась итерация по ценности с применением исчерпывающего прохода по множеству состояний, при этом состояния обновлялись на месте по одному за раз, т. е. при обновлении каждого состояния использовались самые актуальные ценности других состояний (это вариант Гаусса–Зейделя итерации по ценности, который для данной задачи работает примерно в два раза быстрее варианта Якоби. См. раздел 4.8). Порядок обновлений специально не выбирался; возможно, при другом порядке сходимость была бы быстрее. В каждом прогоне обоих методов начальные ценности были равны нулю. Считалось, что ДП сошелся, когда максимальное изменение ценности состояний после прохода было меньше 10^{-4} , а ДПРВ – когда время до пересечения финишной черты, усредненное по 20 эпизодам, выраженное в количестве шагов, стабилизировалось. В этом варианте ДПРВ на каждом шаге обновлялась только ценность текущего состояния.

	ДП	ДПРВ
Средний объем вычислений до сходимости	28 проходов	4000 эпизодов
Среднее число обновлений до сходимости	252 784	127 600
Среднее число обновлений в эпизоде	–	31.9
% состояний, обновленных ≤ 100 раз	–	98.45
% состояний, обновленных ≤ 10 раз	–	80.51
% состояний, обновленных 0 раз	–	3.18

Оба метода выработали стратегии, требующие в среднем от 14 до 15 шагов для пересечения финишной черты, но в случае ДПРВ понадобилось почти вдвое меньше обновлений, чем в случае ДП. Это результат траекторной выборки с единой стратегией, примененной в ДПРВ. Если в случае ДП на каждом проходе обновлялась ценность каждого состояния, то в случае ДПРВ обновлялось меньшее число состояний. В среднем метод ДПРВ обновлял ценности 98.45 % состояний не более 100 раз, а ценности 80.51 % состояний не более 10 раз. Ценности примерно 290 состояний в среднем не обновлялись вовсе. ■

Еще одно преимущество ДПРВ состоит в том, что когда функция ценности стремится к оптимальной функции ценности v_* , стратегия, применяемая агентом для генерации траекторий, стремится к оптимальной стратегии, поскольку она всегда является жадной относительно текущей функции ценности. В этом состоит отличие от ситуации в традиционной итерации по ценности. На практике итерация по ценности завершается, когда изменение функции ценности на проходе невелико, и именно так мы и поступали при получении результатов, показанных в таблице выше. В этот момент функция ценности хорошо аппроксимирует v_* , а жадная

стратегия близка к оптимальной. Однако может статься, что стратегии, жадные относительно последней функции ценности, были оптимальными или близкими к оптимальным задолго до завершения итерации по ценности. (Напомним, что в главе 4 было показано, что оптимальные стратегии могут быть жадными относительно многих разных функций ценности, а не только v_* .) В традиционном алгоритме ДП не проверяется, является ли оптимальной стратегия, выработанная до сходимости итерации по ценности, такая проверка потребовала бы большого объема дополнительных вычислений.

В примере кольцевых гонок прогонялось много тестовых эпизодов после каждого прохода ДП, и действия выбирались жадно относительно результата этого прохода. В итоге появилась возможность оценить самую раннюю точку в процессе вычислений ДП, в которой приближенная оптимальная функция оценивания стала достаточно хорошей, чтобы соответствующая ей жадная политика была почти оптимальной. В этом примере близкая к оптимальной стратегия была выработана после 15 проходов итерации по ценности, или после 136 725 обновлений. Это значительно меньше 252 784 обновлений в методе ДП, необходимых для сходимости к v_* , но все еще больше 127 600 обновлений в случае ДПРВ.

Хотя это имитационное моделирование, безусловно, не является авторитетным сравнением ДПРВ с традиционной итерацией по ценности на основе проходов, оно все же иллюстрирует некоторые преимущества траекторной выборки с единой стратегией. В то время как традиционный метод итерации по ценности продолжал обновлять ценности всех состояний, метод ДПРВ акцентировал внимание на подмножествах пространства состояний, релевантных целевой функции задачи. Чем дольше продолжалось обучение, тем уже становилось это множество. Поскольку к имитациям применима теорема о сходимости для ДПРВ, мы знаем, что в конечном итоге ДПРВ сфокусировался бы только на релевантных состояниях, т. е. на состояниях, образующих оптимальные пути. Для достижения почти оптимального управления ДПРВ хватило примерно вдвое меньше вычислений, чем методу итерации по ценности на основе проходов.

8.8. ПЛАНИРОВАНИЕ В МОМЕНТ ПРИНЯТИЯ РЕШЕНИЙ

Планирование можно использовать по меньшей мере двумя способами. Ранее в этой главе мы рассмотрели подход, типичными представителями которого являются динамическое программирование и Dyna и который состоит в использовании планирования для постепенного улучшения стратегии или функции ценности на основе имитированного опыта, полученного от модели (выборки или распределения). Тогда выбор действий сводится к сравнению ценностей действий в текущем состоянии. Эти ценности могут быть получены из таблицы в табличном случае, который мы изучали до сих пор, или путем вычисления математического выражения в случае приближенных методов, рассматриваемых в части II. Задолго до того как выбрано действие в каком-либо текущем состоянии S_t , планирование играло роль в улучшении элементов таблицы или математического выражения, необходимого для выбора действий во многих состояниях, в т. ч. S_t . При таком использовании планирование не сфокусировано на текущем состоянии и называется *фоновым планированием*.

Другой способ использования планирования – начать и закончить его после встречи нового состояния S_t , рассматривая как вычисление, результатом которого является выбор одного действия A_t ; на следующем шаге планирование начинается заново для состояния S_{t+1} и порождает состояние A_{t+1} . И так далее. Простейший, почти вырожденный пример такого использования планирования возникает, когда доступны только ценности состояний, а действие выбирается путем сравнения ценностей предсказанных моделью следующих состояний для каждого действия (или путем сравнения ценностей послесостояний, как в примере игры в крестики-нолики в главе 1). В общем случае планирование, используемое таким образом, может оказаться гораздо глубже, чем заглядывание на один шаг вперед; можно оценивать выбор действий, приводящих ко многим различным предсказанным состояниям и траекториям вознаграждения. В отличие от первого способа, здесь планирование сфокусировано на конкретном состоянии. Мы называем его *планированием в момент принятия решений*.

Эти два варианта интерпретации планирования – использование имитированного опыта для постепенного улучшения стратегии или функции ценности и использование имитированного опыта для выбора действия в текущем состоянии – можно совмещать естественными и интересными способами, но исторически они исследовались по отдельности, и для первоначального понимания это удобно. Давайте более пристально рассмотрим планирование в момент принятия решений.

Хотя планирование производится только в момент принятий решений, мы все равно можем рассматривать его, как делали в разделе 8.1, как переработку имитированного опыта в обновления и ценности, а в конечном итоге – в стратегии. Просто теперь ценности и стратегия специфичны для текущего состояния и доступных в нем действий – настолько, что ценности и стратегия, выработанные процессом планирования, как правило, отбрасываются после использования для выбора текущего действия. Во многих приложениях это небольшая потеря, потому что состояний очень много, и мы вряд ли скоро вернемся в то же состояние. В общем случае иногда имеет смысл сочетать оба подхода: сфокусировать планирование на текущем состоянии и сохранять результаты планирования, чтобы ускорить обработку, если когда-то мы снова окажемся в нем. Планирование в момент принятия решений особенно полезно в приложениях, где не требуется быстрая реакция. Например, в программе игры в шахматы разрешено обдумывать каждый ход в течение нескольких секунд или минут, а хорошие программы могут в течение этого времени просчитать на несколько десятков ходов вперед. С другой стороны, если на первом месте стоит скорость выбора действия, то, вообще говоря, эффективнее заниматься планированием в фоновом режиме и вычислять стратегию, которая затем может быть быстро применена к любому встретившемуся состоянию.

8.9. ЭВРИСТИЧЕСКИЙ ПОИСК

Классические методы планирования в пространстве состояний, применяемые в области искусственного интеллекта, относятся к методам планирования в момент принятия решений и в совокупности известны под названием *эвристический поиск*. В эвристическом поиске для каждого встретившегося состояния рассматри-

вается большое дерево возможных продолжений. К листовым узлам применяется приближенная функция ценности, а затем производится обновление текущего состояния в корне. Процесс обновления в дереве поиска такой же, как процесс полного $\max$ -обновления (для функций v_* и q_*), рассматриваемый в данной книге. Обновление останавливается в узлах состояние–действие, относящихся к текущему состоянию. После того как обновленные ценности этих узлов вычислены, лучшее из них выбирается в качестве текущего действия, а все остальные обновленные ценности отбрасываются.

В традиционном эвристическом поиске не предпринимается никаких усилий, чтобы сохранить обновленные значения путем изменения приближенной функции ценности. Фактически функция ценности в общем случае задается человеком и никогда не изменяется в результате поиска. Тем не менее было бы естественно рассмотреть случай, когда разрешено улучшать функцию ценности со временем, используя либо обновленные ценности, вычисленные в процессе эвристического поиска, либо какой-нибудь другой метод из числа представленных в этой книге. В некотором смысле мы все время применяли такой подход. Наши жадные, ϵ -жадные и основанные на ВДГ (раздел 2.7) методы выбора действий схожи с эвристическим поиском, хотя и в меньшем масштабе. Например, чтобы вычислить жадное действие при известной модели и функции ценности состояний, мы должны заглянуть вперед для каждого возможного действия в каждое возможное следующее состояние, принять во внимание вознаграждения и оценки ценностей, а затем выбрать наилучшее действие. Как и в случае традиционного эвристического поиска, этот процесс вычисляет обновленные значения возможных действий, но не пытается сохранить их. Таким образом, эвристический поиск можно рассматривать как обобщение идеи жадной стратегии на количество шагов, большее одного.

Смысл поиска с заглядыванием далее, чем на один шаг, заключается в том, чтобы более эффективно выбирать действия. Если имеется идеальная модель и неидеальная функция ценности действий, то более глубокий поиск обычно находит более эффективные стратегии¹. Очевидно, что если поиск осуществляется до самого конца эпизода, то последствия неидеальности функции ценности устраняются, и определенное таким образом действие должно быть оптимальным. Если глубина поиска k настолько велика, что величина γ^k очень мала, то действия будут почти оптимальными. С другой стороны, чем глубже поиск, тем больше объем вычислений, и обычно это приводит к увеличению времени отклика. Хороший пример – разработанная Тезауро программа TD-Gammon для игры в нарды на уровне гроссмейстера (раздел 16.1). В этой системе использовалось TD-обучение для нахождения функции ценности послесостояний, для этого программа сыграла много игр сама с собой, применяя эвристический поиск хороших ходов. В качестве модели в TD-Gammon использовались априорные знания о вероятностях выпадения комбинаций на костях и предположение о том, что противник всегда выбирает действия, которые TD-Gammon оценила бы как наилучшие. Тезауро обнаружил, что чем глубже эвристический поиск, тем лучше ходы, найденные TD-Gammon, но и тем больше времени требуется для каждого хода. Для игры в нарды характерен большой коэффициент ветвления, но на ход все равно отводится не более нескольких секунд. Практически осуществимым оказалось загля-

¹ Из этого утверждения существуют любопытные исключения. См. книгу *Pearl* (1984).

дывание вперед всего на несколько шагов, но даже такой поиск позволил значительно улучшить алгоритм выбора действий.

Не следует упускать из вида самый очевидный способ сфокусировать обновления в эвристическом поиске: на текущем состоянии. Своей эффективностью эвристический поиск в значительной мере обязан тому, что его дерево поиска четко сфокусировано на состояниях и действиях, которые могут последовать непосредственно за текущим состоянием. Возможно, вы чаще играете в шахматы, чем в шашки, но, играя в шашки, следует думать о шашках, о текущей позиции на доске, о возможных следующих ходах и образующихся в результате позициях. Как бы вы ни выбирали действия, именно эти состояния и действия наиболее важны для обновлений, и именно в этой области следует максимально точно аппроксимировать функцию ценности. Близким событиям следует не только отдавать большую часть вычислительных ресурсов, но и ограниченные ресурсы памяти. В шахматах, например, возможных позиций слишком много, чтобы хранить оценки ценности для каждой, но шахматные программы, основанные на эвристическом поиске, легко могут хранить оценки каждой из миллионов позиций, встретившихся при заглядывании вперед из одной позиции. Такое сосредоточение ресурсов памяти и вычислительных ресурсов на принятии текущего решения, по-видимому, и является причиной высокой эффективности эвристического поиска.

Распределение обновлений может быть аналогично изменено для фокусирования на текущем состоянии и его вероятных преемниках. В предельном случае мы могли бы использовать сами методы эвристического поиска для построения дерева поиска, а затем выполнить индивидуальные одношаговые обновления снизу вверх, как показано на рис. 8.9. Если обновления упорядочены таким образом и используется табличное представление, то будет иметь место в точности такое же итоговое обновление, как при эвристическом поиске в глубину. Любой поиск в пространстве состояний можно рассматривать таким образом – как соединение большого числа отдельных одношаговых обновлений. Следовательно, улучшение

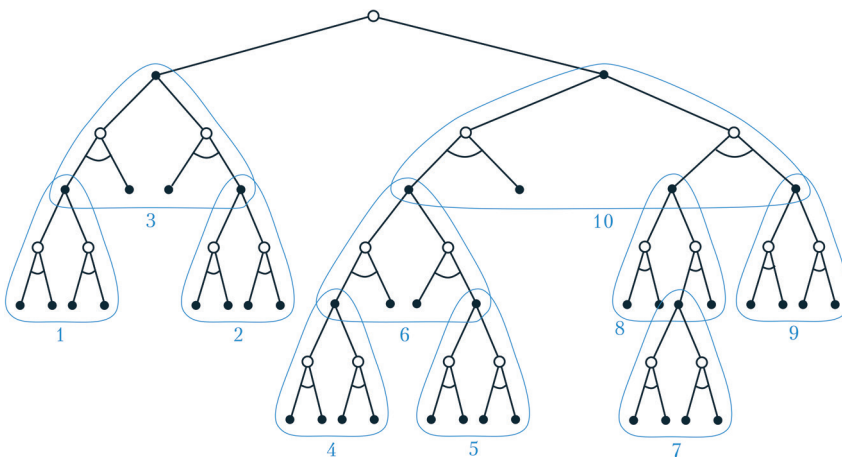


Рис. 8.9 ❖ Эвристический поиск можно реализовать как последовательность одношаговых обновлений (показаны синими линиями), поднимающих ценности от листовых узлов к корню. Показанный на рисунке порядок соответствует избирательному поиску в глубину

результатов, наблюдаемое при более глубоком поиске, достигается не благодаря использованию многошаговых обновлений как таковых, а благодаря концентрации обновлений на состояниях и действиях, непосредственно следующих за текущим состоянием. Выделяя большую часть вычислительных ресурсов для обработки наиболее перспективных действий, планирование в момент принятия решений может находить лучшие решения, чем при бессистемном обновлении.

8.10. РАЗЫГРЫВАЮЩИЕ АЛГОРИТМЫ

Разыгрывающие алгоритмы (rollout algorithm) – это алгоритмы планирования в момент принятия решений, основанные на управлении методом Монте-Карло, применяемым к имитированным траекториям, которые начинаются в текущем состоянии окружающей среды. Они оценивают ценности действий для данной стратегии посредством усреднения доходов на многих имитированных траекториях, которые начинаются с каждого возможного действия, а затем следуют заданной стратегии. Когда будет сочтено, что оценки ценности действий достаточно точны, выполняется действие (или одно из действий) с максимальной оценкой ценности, после чего процесс начинается заново в получившемся следующем состоянии. Как объяснено в работе Тезауро и Гальперина (Tesauro and Galperin, 1997), которые экспериментировали с разыгрывающими алгоритмами для игры в нарды, сам термин «разыгрывающий» (rollout) связан с тем, что оценка ценности позиции в нардах производится путем многократного «разыгрывания» этой позиции до конца игры со случайно сгенерированными последовательностями бросания костей, когда оба игрока делают ходы, следуя некоторой фиксированной стратегии.

В отличие от алгоритмов управления Монте-Карло, описанных в главе 5, цель разыгрывающего алгоритма – не оценить целиком оптимальную функцию ценности действий q_* или функцию действий q_π для заданной стратегии π , а вычислить оценки Монте-Карло только ценностей действий для каждого текущего состояния и для заданной стратегии, которая обычно называется *стратегией разыгрывания*. Будучи алгоритмами планирования в момент принятия решений, разыгрывающие алгоритмы используют найденные оценки ценностей действий и тут же отбрасывают их. Поэтому их довольно просто реализовать, т. к. нет необходимости производить выборку результатов для каждой пары состояние–действие и не нужно аппроксимировать функцию на пространстве состояний или состояний–действий.

Но чего же тогда позволяют достичь разыгрывающие алгоритмы? Теорема об улучшении стратегии из раздела 4.2 утверждает, что если даны две стратегии π и π' , совпадающие всюду, кроме одного состояния s такого, что $\pi'(s) = a \neq \pi(s)$, и если $q_\pi(s, a) \geq v_\pi(s)$, то стратегия π' столь же хороша или лучше, чем π . Более того, если неравенство строгое, то π' на самом деле лучше π . Это применимо к разыгрывающим алгоритмам, если s – текущее состояние, а π – стратегия разыгрывания. Усреднение доходов на имитированных траекториях дает оценки $q_\pi(s, a')$ для каждого действия $a' \in \mathcal{A}(s)$. Тогда стратегия, которая выбирает в s действие, доставляющее максимум этой оценки, а затем следует стратегии π , является хорошим кандидатом на роль стратегии, улучшающей π . Результат получается такой, как на

одном шаге алгоритма динамического программирования, выполняющего итерацию по стратегиям, который обсуждался в разделе 4.3 (хотя он больше похож на один шаг асинхронной итерации по стратегиям из раздела 4.5, потому что изменяет действие только для текущего состояния).

Иными словами, цель разыгрывающего алгоритма – улучшение стратегии разыгрывания, а не нахождение оптимальной стратегии. Опыт показывает, что разыгрывающие алгоритмы могут быть на удивление эффективны. Например, авторы работы Tesauro and Galperin (1997) были поражены тем, насколько резко разыгрывающий алгоритм повысил качество программы игры в нарды. В некоторых приложениях разыгрывающий алгоритм может показывать хорошие результаты, даже если стратегия разыгрывания абсолютно случайна. Но качество улучшенной стратегии зависит от свойств стратегии разыгрывания и ранжирования действий, порождаемых с помощью оценок ценности по методу Монте-Карло. Интуиция подсказывает, что чем лучше стратегия разыгрывания и чем точнее оценки ценности, тем лучше, скорее всего, будет стратегия, выработанная разыгрывающим алгоритмом (однако см. работу Gelly and Silver, 2007).

Тут присутствуют важные компромиссы, потому что для выработки лучшей стратегии разыгрывания обычно необходимо потратить больше времени для имитации траекторий в достаточном количестве, чтобы получить хорошие оценки ценности. Будучи методами планирования в момент принятия решений, разыгрывающие алгоритмы, как правило, должны удовлетворять жестким временным ограничениям. Время, необходимое разыгрывающему алгоритму для вычислений, зависит от количества действий, которые нужно оценить для каждого решения, количества временных шагов в имитированных траекториях, необходимого для получения полезного выборочного дохода, времени, которое требуется стратегии разыгрывания для принятия решений, и количества имитированных траекторий, нужного для получения хороших оценок ценности действий методом Монте-Карло.

Нахождение баланса между этими факторами важно в любом приложении разыгрывающих методов, хотя существует несколько способов облегчить задачу. Поскольку испытания Монте-Карло независимы, можно параллельно производить много испытаний на разных процессорах. Еще один подход – усекавать имитированные траектории, не доводя их до полного эпизода, и корректировать усеченный доход с помощью хранимой функции оценивания (и это возвращает на сцену все, что мы говорили об усеченных доходах и обновлениях в предыдущих главах). Можно также, как предлагается в работе Tesauro and Galperin (1997), следить за моделированием Монте-Карло и отсекаать действия-кандидаты, которые вряд ли смогут претендовать на роль лучших или ценность которых достаточно близка к ценности текущих наилучших действий, так что выбор одних вместо других ничего не даст (хотя Тезауро и Гальперин отмечают, что это усложнило бы параллельную реализацию).

Обычно мы не считаем разыгрывающие алгоритмы алгоритмами обучения, потому что они не хранят протяженную историю ценностей или стратегий. Однако эти алгоритмы пользуются некоторыми особенностями обучения с подкреплением, которые были отмечены в данной книге. Будучи примерами методов управления Монте-Карло, они оценивают ценности действий путем усреднения доходов по набору выборочных траекторий, в данном случае траекторий имитированных

взаимодействий с моделью выборки окружающей среды. В этом отношении они похожи на алгоритмы обучения с подкреплением, т. к. избегают исчерпывающего прохода, свойственного динамическому программированию, за счет траекторной выборки и избегают необходимости использовать модели распределения, полагаясь на выборочное, а не на полное обновление. Наконец, разыгрывающие алгоритмы пользуются свойством улучшения стратегии, действуя жадно относительно оценок ценности действий.

8.11. Поиск по дереву методом Монте-Карло

Поиск по дереву методом Монте-Карло (ПДМК) (Monte Carlo Tree Search – MCTS) – недавно открытый и поразительно успешный пример планирования в момент принятия решений. В основе своей ПДМК – описанный выше разыгрывающий алгоритм, но дополненный средствами запоминания оценок ценностей, полученных при моделировании методом Монте-Карло, чтобы успешно направить прямые имитации в сторону траекторий, сулящих большее вознаграждение. Именно благодаря ПДМК удалось поднять уровень программы для игры в го с любительского в 2005 году до гроссмейстерского (шестой дан и выше) в 2015-м. Было разработано много вариантов базового алгоритма, в т. ч. вариант, обсуждаемый в разделе 16.6, который сыграл важнейшую роль в ошеломительных победах программы AlphaGo на 18-м мировом чемпионате по игре в го в 2016 году. ПДМК доказал свою эффективность на многих конкурсах, включая общеигровые (см., например, Finnsson and Björnsson, 2008; Genesereth and Thielscher, 2014), но он не ограничен только играми; его вполне можно использовать в одноагентных задачах последовательного принятия решений, если имеется модель окружающей среды, достаточно простая для быстрой многошаговой имитации.

ПДМК выполняется после обнаружения каждого нового состояния с целью выбрать действие агента в этом состоянии; он выполняется снова для выбора действия в следующем состоянии и т. д. Как и в разыгрывающем алгоритме, каждое выполнение – это итеративный процесс, который имитирует много траекторий, начинающихся в текущем состоянии и продолжающихся до заключительного (или пока обесценивание не сделает дальнейшие вознаграждения пренебрежимо малыми и не влияющими на доход). Основная идея ПДМК – последовательно фокусироваться на нескольких имитациях, начинающихся в текущем состоянии, расширяя начальные участки траекторий, получивших высокие оценки в прежних имитациях. ПДМК не обязан сохранять приближенные функции ценности или стратегии, вычисленные в процессе одного выбора действия, до следующего, хотя во многих реализациях все же сохраняются ценности выбранных действий, которые с большой вероятностью могут оказаться полезными при следующем выполнении.

Действия в имитированных траекториях генерируются в основном с помощью простой стратегии, которая обычно называется стратегией разыгрывания, как и для более простых разыгрывающих алгоритмов. Если ни стратегия разыгрывания, ни модель не требуют большого объема вычислений, то за короткое время можно сгенерировать много имитированных траекторий. Как и в любом табличном методе Монте-Карло, ценность пары состояние–действие оценивается как среднее по (имитированным) доходам, полученным от этой пары. Оценки ценно-

сти Монте-Карло сохраняются только для подмножества пар состояние–действие, которые с наибольшей вероятностью будут достигнуты за несколько шагов, а это подмножество образует дерево с корнем в текущем состоянии, показанное на рис. 8.10. ПДМК инкрементно расширяет это дерево – добавляет в него узлы, представляющие наиболее многообещающие состояния, основываясь на результатах имитированных траекторий. Любая имитированная траектория будет проходить по этому дереву и завершаться в каком-то листовом узле. Вне дерева и в листовых узлах для выбора действий используется стратегия разыгрывания, но для состояний во внутренних узлах дерева есть более привлекательная альтернатива. В этих состояниях у нас имеются оценки хотя бы некоторых действий, поэтому мы можем выбирать из них, применяя так называемую *древесную стратегию*, сочетающую исследование и использование. Например, древесная стратегия могла бы выбирать действия, применяя ϵ -жадное правило или правило ВДГ (глава 2).

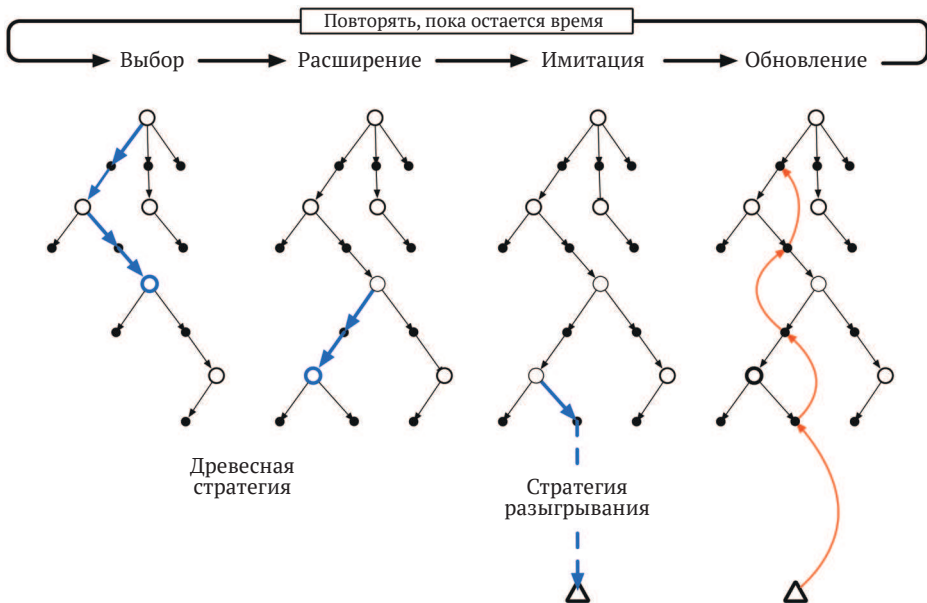


Рис. 8.10 ❖ Поиск по дереву методом Монте-Карло (ПДМК). Когда среда переходит в новое состояние, ПДМК выполняет столько итераций, сколько возможно, пока не возникнет необходимость в выборе действия. При этом инкрементно строится дерево, корень которого представляет текущее состояние. На каждой итерации производится четыре операции: **Выбор**, **Расширение** (на некоторых итерациях может пропускаться), **Имитация** и **Обновление**. Они объяснены в тексте и показаны жирными стрелками в деревьях. Рисунок заимствован из работы Chaslot, Bakkes, Szita, and Spronck (2008)

Точнее, каждая итерация в базовой версии ПДМК состоит из следующих четырех шагов, показанных на рис. 8.10:

- 1) **выбор**. Начав с корневого узла, *древесная стратегия* обходит дерево для выбора листового узла, основываясь на ценностях действий, которыми помечены ребра дерева;

- 2) **расширение.** На некоторых итерациях (каких именно, зависит от реализации) дерево расширяется путем добавления одного или нескольких потомков выбранного листового узла, достижимых из выбранного узла с помощью действий, не требующих исследования;
- 3) **имитация.** Из выбранного узла или одного из его вновь добавленных потомков (если таковые имеются) запускается имитация полного эпизода с действиями, выбираемыми стратегией разыгрывания. Результатом является испытание Монте-Карло с действиями, выбранными сначала древесной стратегией, а вне дерева – стратегией разыгрывания;
- 4) **обновление.** Доход, сгенерированный в имитированном эпизоде, используется, чтобы обновить или инициализировать ценности действий, присоединенные к ребрам дерева, которое обходит древесная стратегия на этой итерации ПДМК. Ценности состояний и действий, посещенных согласно стратегии разыгрывания вне дерева, не сохраняются. На рис. 8.10 это иллюстрируется обновлением узла состояние–действие, в котором началась стратегия разыгрывания, непосредственно из заключительного состояния имитированной траектории.

ПДМК продолжает выполнять эти четыре шага, каждый раз начиная с корня дерева, пока не исчерпает отведенное время или еще какой-то вычислительный ресурс. Затем в корневом узле (который все еще представляет текущее состояние окружающей среды) выбирается действие согласно некоторому механизму, зависящему от накопленной в дереве статистики; например, это может быть действие с наибольшей ценностью среди всех действий, доступных в корневом состоянии, или действие с наибольшим количеством посещений, чтобы избежать выбросов. Это и будет действие, выбранное методом ПДМК. После того как среда перейдет в новое состояние, ПДМК запускается опять. Иногда он начинается с дерева, содержащего только корневой узел, который представляет новое состояние, но чаще с дерева, содержащего всех потомков этого узла, оставшихся от дерева, построенного при предыдущем выполнении ПДМК; все остальные узлы отбрасываются вместе с ценностями действий, которые были с ними ассоциированы.

ПДМК впервые был предложен для выбора ходов в программах для игр с двумя участниками, типа го. В играх каждый имитированный эпизод – это одна полная партия, в которой оба игрока выбирают действия, следуя древесной стратегии и стратегии разыгрывания. В разделе 16.6 описывается обобщение ПДМК, использованное в программе AlphaGo, в котором оценивание методом Монте-Карло сочетается с ценностями действий, вычисленными глубокой нейронной сетью посредством обучения с подкреплением на играх программы с собой же.

Соотнесение метода ПДМК с принципами обучения с подкреплением, описываемыми в этой книге, проливает свет на то, как этот метод достигает столь впечатляющих результатов. В основе своей ПДМК – это алгоритм планирования в момент принятия решений, основанный на управлении Монте-Карло, которое применяется к имитациям, начинающимся в корневом состоянии, т. е. это вариант разыгрывающего алгоритма, описанного в предыдущем разделе. Поэтому он извлекает пользу из онлайн-ового, инкрементного, основанного на выборке оценивания и улучшения стратегии. Дополнительно он сохраняет оценки ценности действий, присоединяя их к ребрам дерева, и обновляет их, применяя выборочное обновление, как в обучении с подкреплением. В результате испытания Мон-

те-Карло фокусируются на траекториях, начальные участки которых являются общими с ранее имитированными высокодоходными траекториями. Далее, путем инкрементного расширения дерева ПДМК, по существу, расширяет таблицу соответствия, представляющую частичную функцию ценности действий, добавляя в нее записи об оценке ценностей пар состояние–действие, посещенных на начальных участках высокодоходных выборочных траекторий. Таким образом, ПДМК обходит проблему глобальной аппроксимации функции ценности действий, сохраняя при этом все преимущества использования прошлого опыта для управления исследованием.

Поразительный успех планирования в момент принятия решений, воплощенного в ПДМК, оказал глубокое влияние на искусственный интеллект. Многие исследователи изучают модификации и обобщения базовой процедуры для применения в играх и одноагентных приложениях.

8.12. РЕЗЮМЕ ГЛАВЫ

Для планирования необходима модель окружающей среды. *Модель распределения* состоит из вероятностей следующих состояний и вознаграждений за возможные действия; *модель выборки* порождает одиночные переходы и вознаграждения, генерируемые в соответствии с этими вероятностями. Динамическое программирование нуждается в модели распределения, потому что производит *полное обновление*, включающее вычисление математического ожидания по всем возможным следующим состояниям и действиям. С другой стороны, *модель выборки* – это то, что необходимо для имитации взаимодействия с окружающей средой, в процессе которого можно использовать выборочное обновление, как во многих алгоритмах обучения с подкреплением. В общем случае модели выборки получить проще, чем модели распределения.

Мы представили точку зрения, в которой подчеркиваются удивительно тесные связи между планированием оптимального поведения и обучением оптимальному поведению. И там, и там участвует оценивание одних и тех же функций ценности, и в обоих случаях естественно производить обновление оценок инкрементно, в ходе длинной последовательности мелких операций обновления. Поэтому процессы планирования и обучения несложно объединить, просто разрешив обоим обновлять одну и ту же оценку функции ценности. Кроме того, любой метод обучения можно преобразовать в метод планирования, просто применив его к имитированному (сгенерированному с помощью модели), а не к реальному опыту. В таком случае обучение и планирование становятся еще более схожими; это может быть один и тот же алгоритм, применяемый к двум разным источникам опыта.

Нетрудно интегрировать инкрементные методы планирования с исполнением и обучением модели. Планирование, исполнение и обучение модели взаимодействуют циклически (как показано на рисунке на стр. 198); на каждом этапе порождается то, что необходимо для улучшения остальных, а другие взаимодействия между ними не требуются, но и не запрещаются. Самый естественный подход – выполнять все процессы асинхронно и параллельно. Если процессы должны совместно пользоваться какими-то вычислительными ресурсами, то разделением

можно управлять почти произвольно – выбирайте такую организацию, которая наиболее удобна и эффективна для решения конкретной задачи.

В этой главе мы затронули различные настраиваемые параметры методов планирования в пространстве состояний. Один из них – размер обновлений. Чем он меньше, тем более инкрементными могут быть методы планирования. К числу методов с наименьшим размером обновления относятся одношаговые выборочные обновления, например Дуна. Еще один важный параметр – распределение обновлений, т. е. направленность поиска. Методы с приоритетным проходом фокусируются на предшественниках состояний, ценности которых были недавно изменены. Методы траекторной выборки с единой стратегией – на состояниях или парах состояние–действие, которые агент с большой вероятностью встретит в процессе управления окружающей средой. Иногда это позволяет отказаться от вычислений в тех частях пространства состояний, которые нерелевантны задаче предсказания или управления. Метод динамического программирования в реальном времени, вариант итерации по ценности, основанный на траекторной выборке с единой стратегией, иллюстрирует некоторые преимущества этого подхода по сравнению с традиционной итерацией по стратегиям на основе прохода по пространству состояний.

Планирование может быть также устремлено вперед от наиболее интересных состояний, например от состояний, которые фактически встречались в ходе взаимодействия агента с окружающей средой. Самая важная форма такого подхода – когда планирование производится в момент принятия решений, т. е. является частью процесса выбора действия. Примером может служить классический эвристический поиск, изучаемый в искусственном интеллекте. Другие примеры – разыгрывающие алгоритмы и поиск по дереву методом Монте-Карло, в основе которых лежит онлайнное, инкрементное, основанное на выборке оценивание ценности и улучшение стратегии.

8.13. РЕЗЮМЕ ЧАСТИ I: ОСИ

Этой главой завершается первая часть книги. В ней мы стремились представить обучение с подкреплением не как собрание отдельных методов, а как связный набор идей, лежащих в основе всех методов. Каждую идею можно рассматривать как ось, вдоль которой располагаются методы. Множество таких осей образует большое пространство возможных методов. Исследуя это пространство на уровне осей, мы надеемся получить максимально широкое понимание, которое долго не устареет. В этом разделе мы используем концепцию осей в пространстве методов, чтобы конспективно охарактеризовать тот взгляд на обучение с подкреплением, который мы развивали в данной книге.

Все рассмотренные выше методы характеризуются тремя общими ключевыми идеями. Во-первых, все они оценивают функции ценности. Во-вторых, принцип действия каждого основан на обновлении ценностей вдоль реальных или возможных траекторий изменения состояний. В-третьих, все они исповедуют подход на основе обобщенной итерации по стратегиям (ОИС), т. е. хранят приближенную функцию ценности и приближенную стратегию и постоянно пытаются улучшить

одну на основе другой. Это три центральные идеи для всех тем, обсуждаемых в книге. Мы полагаем, что функции ценности, обновление ценностей и ОИС – главные организующие принципы, свойственные любой модели интеллекта, искусственного или естественного.

На рис. 8.11 показаны две наиболее важные оси, вдоль которых располагаются различные методы. Они относятся к типу обновлений, применяемых для улучшения функции ценности. Вдоль горизонтальной оси располагаются выборочные обновления (основанные на выборочной траектории) и полные обновления (основанные на распределении возможных траекторий). Для полных обновлений необходима модель распределения, а для выборочных – только модель выборки, хотя их можно производить на основе реального опыта, вообще без модели (еще одна ось изменения). Вертикальная ось соответствует глубине обновлений, т. е. степени бутстрэппинга. В трех из четырех углов пространства находятся основные методы оценивания ценности: динамическое программирование, TD и Монте-Карло. Вдоль левого края пространства располагаются методы выборочного обновления – от одношаговых TD-обновлений до обновлений Монте-Карло с полным доходом. Между ними размещается целый спектр методов, включающий методы на основе n -шаговых обновлений (а в главе 12 мы обобщим их на смеси n -шаговых обновлений, в частности λ -обновления, реализованные с помощью следов приемлемости).

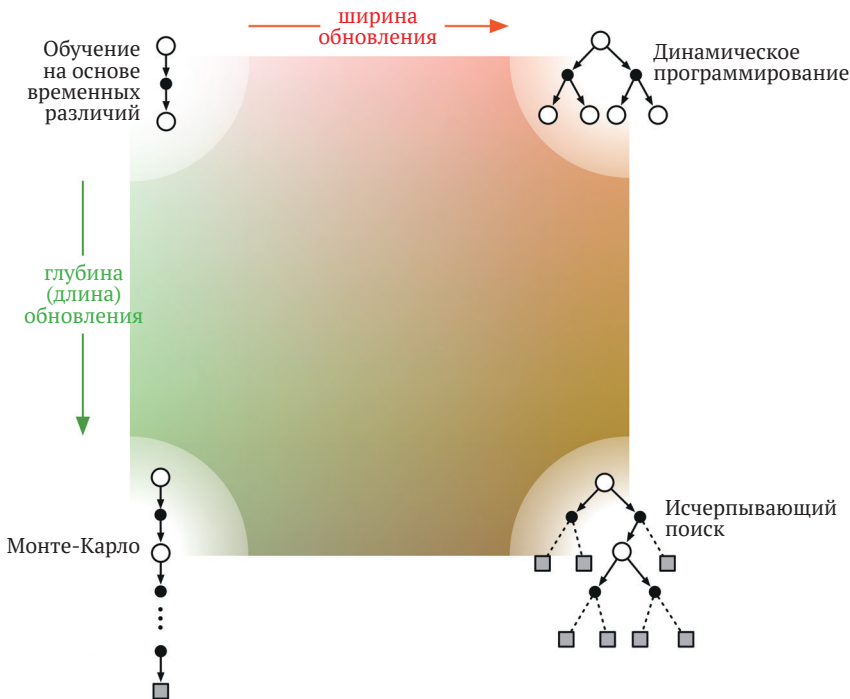


Рис. 8.11 ❖ Срез пространства методов обучения с подкреплением, на котором показаны две наиболее важные оси, изученные в первой части книги: глубина и ширина обновлений

Методы динамического программирования показаны в правом верхнем углу, поскольку включают одношаговое полное обновление. В правом нижнем углу расположен случай максимально глубокого полного обновления, продолжающегося до заключительного состояния (или, в непрерывной задаче, до тех пор, пока вклад дальнейших вознаграждений не упадет до пренебрежимо малого уровня). Это случай исчерпывающего поиска. К промежуточным методам вдоль этой оси относятся эвристический поиск и связанные с ним методы, которые производят поиск и обновление до определенного уровня, быть может, избирательно. Существуют также методы, сочетающие полное и выборочное обновления, а еще методы, сочетающие модель выборки и модель распределения в одном обновлении. Залитая внутренняя часть квадрата представляет пространство всех таких промежуточных методов.

Третья ось, о которой мы много говорили выше, – это бинарное различие между методами с единой и разделенной стратегиями. В первом случае агент обучается функции ценности для той стратегии, которой следует, а во втором – функции ценности для другой стратегии, зачастую такой, которую в данный момент считает лучшей. Обычно поведение, генерирующее стратегию, отличается от предположительно оптимального из-за необходимости продолжать исследование. Третью ось можно представлять себе как перпендикуляр к плоскости, изображенной на рис. 8.11.

Помимо трех вышеупомянутых осей, в книге упоминается ряд других.

- **Определение дохода.** Является ли задача эпизодической или непрерывной, с обесцениванием или без него?
- **Ценности действий, ценности состояний и ценности послесостояний.** Какого рода ценности следует оценивать? Если оцениваются только ценности состояний, то для выбора действий необходима либо модель, либо отдельная стратегия (как в методах типа исполнитель–критик).
- **Выбор действия или исследование.** Как производится выбор действий, чтобы обеспечить приемлемый компромисс между исследованием и использованием? Мы рассмотрели только простейшие способы: ϵ -жадный, оптимистическая инициализация ценностей, softmax и верхняя доверительная граница.
- **Синхронно или асинхронно.** Производятся ли обновления всех состояний одновременно или поочередно в некотором порядке?
- **Реальный или имитированный.** Должно ли обновление быть основано на реальном или имитированном опыте? Если на обоих, то каким должно быть соотношение между ними?
- **Место обновления.** Какие состояния или пары состояние–действие следует обновлять? Безмодельные методы могут выбирать только среди состояний или пар состояние–действие, которые реально встречались, а методам, основанным на модели, доступен произвольный выбор. Тут открывается много возможностей.
- **Время обновления.** Следует ли производить обновление как часть выбора действий или только после него?
- **Память для обновлений.** В течение какого времени следует хранить обновленные ценности? Нужно ли их хранить постоянно или только пока вычисляется, какое действие выбрать, как в эвристическом поиске?

Конечно, этот перечень осей неполон, и оси не являются взаимно исключаящими. Алгоритмы могут различаться и во многих других отношениях, а есть такие алгоритмы, которые располагаются в нескольких местах на нескольких осях. Например, в Дина-методах используется как реальный, так и имитированный опыт для вычисления одной и той же функции ценности. Также ничто не мешает вычислять сразу несколько функций ценности разными способами или используя разные представления состояний и действий. Но эти оси составляют комплекс тесно связанных между собой идей для описания и исследования широкого пространства возможных методов.

Здесь не упомянута важнейшая ось, которая не рассматривалась в первой части книги, – аппроксимация функций. Ее можно рассматривать как совершенно независимый спектр возможностей, на одном конце которого находятся табличные методы, а далее следуют агрегирование состояний, разнообразные линейные и, наконец, нелинейные методы. Эту ось мы будем исследовать во второй части книги.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

- 8.1 Представленный здесь общий взгляд на планирование и обучение складывался постепенно на протяжении ряда лет, и к этому приложили руку авторы (Sutton, 1990, 1991a, 1991b; Barto, Bradtke, and Singh, 1991, 1995; Sutton and Pinette, 1985; Sutton and Barto, 1981b). Сильное влияние на него оказали работы Agre and Chapman (1990), Agre (1988), Bertsekas and Tsitsiklis (1989), Singh (1993) и др. На авторов также оказали влияние работы по *латентному обучению* в психологии (Tolman, 1932) и представления психологов о природе мышления (например, Galanter and Gerstenhaber, 1956; Craik, 1943; Campbell, 1960; Dennett, 1978). В разделе 14.6 третьей части книги производится сопоставление основанных на модели и безмодельных методов с психологическими теориями обучения и поведения, а в разделе 15.11 обсуждаются идеи о том, как мозг мог бы реализовать эти типы методов.
- 8.2 Термины *прямое* и *косвенное*, которые мы употребляем для описания различных видов обучения с подкреплением, заимствованы из литературы по адаптивному управлению (см., например, Goodwin and Sin, 1984), где они используются для проведения аналогичного различия. То, что мы называем *обучением модели*, в адаптивном управлении обозначается термином *идентификация системы* (Goodwin and Sin, 1984; Ljung and Söderstrom, 1983; Young, 1984). Архитектура Dyna предложена в статье Sutton (1990), а результаты, изложенные в этом и следующем разделах, основаны на полученных в этой работе. В работе Barto and Singh (1990) рассматриваются некоторые вопросы сравнения прямых и косвенных методов обучения с подкреплением. Одна из первых работ по обобщению Dyna на линейные аппроксимации функций (глава 9) выполнена в статьях Sutton, Szepesvári, Geramifard, and Bowling (2008) и Parr, Li, Taylor, Painter-Wakefield, and Littman (2008).
- 8.3 Существует несколько работ по обучению с подкреплением на основе модели, в которых идеи приза за исследование и оптимистической инициа-

лизации доведены до логического завершения, когда предполагается, что за все не полностью исследованные варианты назначается максимальное вознаграждение, и для их проверки вычисляются оптимальные пути. Алгоритм E3, описанный в работе Kearns and Singh (2002), и алгоритм R-max, описанный в работе Brafman and Tennenholtz (2003), гарантированно находят почти оптимальное решение за время, полиномиально зависящее от количества состояний и действий. На практике это обычно слишком медленно, но, возможно, это лучшее, чего можно добиться в худшем случае.

- 8.4 Метод приоритетного прохода был разработан одновременно и независимо в работах Moore and Atkeson (1993) и Peng and Williams (1993). Результаты, приведенные во врезке на стр. 208, взяты из работы Peng and Williams (1993), а результаты во врезке на стр. 209 – из работы Мура и Аткесона. К числу последующих ключевых достижений в этой области относятся работы McMahan and Gordon (2005) и van Seijen and Sutton (2013).
- 8.5 На этот раздел оказали сильное влияние эксперименты Сингха (Singh, 1993).
- 8.6–7 Траекторная выборка неявно была частью обучения с подкреплением с самого начала, но в явном виде сформулирована в работе Barto, Bradtke, and Singh (1995), где приводится введение в ДППВ. Авторы осознали, что *алгоритм обучения в реальном времени A** Корфа (Korf, 1990) (LRTA*) является асинхронным алгоритмом ДП, который применим к стохастическим задачам, так же как и к детерминированным, которыми занимался Корф. В дополнение к LRTA* ДППВ включает возможность обновления ценностей многих состояний в промежутки времени между выполнением действий. В работе Barto et al. (1995) упомянутый здесь результат о сходимости доказан путем объединения доказательства сходимости LRTA* из работы Korf (1990) с результатом Бертсекаса (Bertsekas, 1982, см. также Bertsekas and Tsitsiklis, 1989), устанавливающим сходимость асинхронного ДП для задач о кратчайшем пути в стохастической сети в случае без обесценивания. Сочетание обучения модели с ДППВ, называемое адаптивным ДППВ, также представлено в работе Barto et al. (1995) и обсуждается в работе Barto (2011).
- 8.9 Читателям, интересующимся эвристическим поиском, мы рекомендуем учебники и обзоры, в т. ч. Russell and Norvig (2009) и Korf (1988). В работе Peng and Williams (1993) прямая фокусировка обновлений изучается в том же ключе, что в этом разделе.
- 8.10 Модель ожидаемого выхода Абрамсона (Abramson, 1990) представляет собой разыгрывающий алгоритм, применимый к играм с двумя участниками, в которых оба имитированных игрока выбирают случайные ходы. Автор утверждает, что даже при случайной игре это «мощная эвристика», потому что «дает точные и верные результаты, легко поддающиеся оценке, допускающие эффективное вычисление и не зависящие от предметной области». В работе Tesauro and Galperin (1997) продемонстрирована эффективность разыгрывающих алгоритмов для улучшения программ игры в нарды – путем разыгрывания позиций с различными случайно сгенерированными последовательностями бросания костей. В работе Bertsekas, Tsitsiklis, and Wu (1997) исследуется применение разыгрывающих алгоритмов к задачам

комбинаторной оптимизации, а в работе Bertsekas (2013) приведен обзор их применения к задачам дискретной детерминированной оптимизации, сопровождаемый замечанием о том, что они «зачастую оказываются на удивление эффективны».

- 8.11** Основные идеи ПДМК были сформулированы в работах Coulom (2006) и Kocsis and Szepesvári (2006). Они основаны на предшествующих исследованиях алгоритмов планирования методом Монте-Карло. Работа Browne, Powley, Whitehouse, Lucas, Cowling, Rohlfshagen, Tavener, Perez, Samothrakis, and Colton (2012) представляет собой великолепный обзор методов ПДМК и их приложений. Дэвид Силвер (David Silver) внес вклад в эти идеи и их изложение в данном разделе.

ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ

Во второй части книги мы обобщим табличные методы, описанные в первой части, на задачи со сколь угодно большим пространством состояний. Во многих задачах, к которым мы хотели бы применять обучение с подкреплением, пространство состояний комбинаторное, а его размер огромен; например, количество возможных фотографий многократно превышает количество атомов во Вселенной. В таких случаях нельзя ожидать, что будет найдена оптимальная стратегия или оптимальная функция ценности, даже при наличии неограниченного времени и данных. Наша цель скромнее – найти хорошее приближенное решение в условиях ограниченных вычислительных ресурсов. В этой части книги мы будем изучать такие методы приближенного решения.

Проблема большого числа состояний заключается не только в объеме памяти, необходимом для больших таблиц, но и в том, что для их заполнения нужно много времени и данных. Во многих задачах, которые нас интересуют, почти все встретившиеся состояния раньше не встречались. Для принятия разумных решений в таких состояниях необходима способность к обобщению других, ранее встречавшихся состояний, в чем-то похожих на текущее. Иными словами, ключевым вопросом является *обобщение*. Как можно обобщить опыт, представляющий собой ограниченное подмножество пространства состояний, чтобы получить хорошую аппроксимацию гораздо большего подмножества?

По счастью, обобщение примеров – хорошо изученная область, и нам не нужно изобретать совершенно новые методы для использования в обучении с подкреплением. В общем-то, нам нужно лишь объединить методы обучения с подкреплением и уже существующие методы обобщения. Нужный нам вид обобщения часто называют *аппроксимацией функций*, потому что мы берем примеры желаемой функции (например, функции ценности) и пытаемся обобщить их, т. е. построить аппроксимацию всей функции. Аппроксимация функций – частный случай обучения с учителем, главной темы, изучаемой в таких дисциплинах, как машинное обучение, искусственные нейронные сети, распознавание образов и статистическая аппроксимация кривых. Теоретически в роли метода аппроксимации в алгоритмах обучения с подкреплением может выступать любой метод, изучаемый в этих дисциплинах, но на практике некоторые больше подходят на эту роль, чем другие.

Обучение с подкреплением с аппроксимацией функций поднимает ряд новых вопросов, которые обычно не возникают в традиционном обучении с подкреплением, в том числе нестационарность, бутстрэппинг и отложенные цели. Мы последовательно рассмотрим эти и другие вопросы в последующих главах. Для начала ограничимся обучением с единой стратегией – в главе 9 будет рассмотрен случай задачи предсказания, в которой стратегия задана, а аппроксимируется только ее функция ценности. Затем, в главе 10, рассматривается задача управления, в которой требуется найти аппроксимацию оптимальной стратегии. Трудной задаче обучения с разделенной стратегией с аппроксимацией функций посвящена глава 11. В каждой из этих трех глав мы будем возвращаться к основополагающим принципам и пересматривать цели обучения с учетом аппроксимации функций. В главе 12 вводится и анализируется алгоритмический механизм *следов приемлемости*, который во многих случаях кардинально улучшает вычислительные свойства многошаговых методов обучения с подкреплением. И в последней главе этой части исследуется другой подход к управлению, *методы градиента стратегии*, в которых оптимальная стратегия аппроксимируется непосредственно, а приближенная функция ценности вообще не присутствует (хотя эффективность может оказаться гораздо выше, если аппроксимируется не только стратегия, но и функция ценности).

Предсказание с единой стратегией и аппроксимацией

В этой главе мы начнем изучение аппроксимации функций в обучении с подкреплением с рассмотрения его применения к оцениванию функции ценности состояний по данным единой стратегии. То есть мы будем изучать аппроксимацию v_π на основе опыта, сгенерированного с помощью известной стратегии π . Новое в этой главе то, что приближенная функция ценности представлена не в виде таблицы, а в виде параметрической функции с вектором весов $\mathbf{w} \in \mathbb{R}^d$. Будем обозначать $\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$ приближенную ценность состояния s при заданном векторе весов $\mathbf{w}$. Например, $\hat{v}$ может быть линейной функцией от признаков состояния, а $\mathbf{w}$ – вектором весов признаков. Более общо, $\hat{v}$ может быть функцией, вычисленной многослойной нейронной сетью, тогда $\mathbf{w}$ – вектор весов связей между всеми слоями. За счет подбора весов сеть может реализовать любую функцию из широкого диапазона. Или $\hat{v}$ может быть функцией, вычисляемой решающим деревом, тогда $\mathbf{w}$ – числа, определяющие ценности в точках разделения и в листовых узлах дерева. Обычно количество весов (размерность $\mathbf{w}$) гораздо меньше количества состояний ($d \ll |S|$), и изменение одного веса меняет оценку ценности сразу многих состояний. Следовательно, когда обновляется одно состояние, это изменение обобщается и влияет на ценности многих других состояний. Такое обобщение потенциально может повысить эффективность обучения, но, возможно, понять, что происходит, и управлять процессом будет труднее.

Как ни странно, но распространение обучения с подкреплением на аппроксимацию функций позволяет применить его также к частично наблюдаемым задачам, в которых полное состояние агенту недоступно. Если параметрическая форма функции $\hat{v}$ не допускает зависимости оценки ценности от некоторых аспектов состояния, то можно считать, что эти аспекты ненаблюдаемы. На самом деле все теоретические результаты, описанные в этой части книги для методов с аппроксимацией функций, применимы также к случаям частичной наблюдаемости. Но вот что аппроксимации функций неподвластно, так это пополнение представления состояний воспоминаниями о прошлых наблюдениях. Некоторые возможности развития в этом направлении кратко обсуждаются в разделе 17.3.

9.1. Аппроксимация функции ценности

Все методы предсказания, упомянутые в этой книге, трактовались как обновление оценки функции ценности, которые сдвигают ее значение в конкретных состояниях в направлении «обновленной ценности», или цели обновления для этого состояния. Обозначим отдельное обновление $s \mapsto u$, где s – обновленное состояние, а u – цель обновления, в направлении которого сдвигается оценка ценности s . Например, обновление Монте-Карло для предсказания ценности имеет вид $S_t \mapsto G_t$, обновление TD(0) – $S_t \mapsto R_{t+1} + \gamma \hat{V}(S_{t+1}, \mathbf{w}_t)$, а n -шаговое TD-обновление – $S_t \mapsto G_{t:t+n}$. В случае оценивания стратегии в динамическом программировании (ДП), $s \mapsto \mathbb{E}_\pi[R_{t+1} + \gamma \hat{V}(S_{t+1}, \mathbf{w}_t) | S_t = s]$, обновляется произвольное состояние s , тогда как в остальных случаях – состояние, встретившееся в реальном опыте, S_t .

Естественно интерпретировать каждое обновление как задание примера желательного поведения функции ценности в терминах входа и выхода. В некотором смысле обновление $s \mapsto u$ означает, что оценка ценности состояния s должна быть ближе к цели обновления u . До сих пор фактическое обновление было тривиальным: элемент таблицы, соответствующий оценке ценности s , просто сдвигался на некоторое расстояние в направлении u , а оценки ценностей всех остальных состояний не изменялись. Теперь же для реализации обновления мы разрешаем произвольно сложные методы, а обновление s обобщается, так что оценки ценностей многих других состояний также изменяются. Методы машинного обучения, которые обучаются подражать примерам входа и выхода описанным образом, называются методами обучения с учителем, а если на выходе получаются числа, например u , то процесс часто называется *аппроксимацией функции*. Методы аппроксимации функций на входе получают примеры входов и выходов функции, которую стремятся аппроксимировать. Мы применяем эти методы для предсказания ценности, просто передавая им пары $s \mapsto g$ для каждого обновления в качестве обучающих примеров. А затем порожденная ими приближенная функция интерпретируется как оценка функции ценности.

Рассмотрение каждого обновления как традиционного обучающего примера дает нам возможность применять любой из широкого спектра существующих методов аппроксимации функций для предсказания ценности. В принципе, можно использовать любой метод обучения с учителем на примерах, в т. ч. искусственные нейронные сети, решающие деревья и различные варианты многомерной регрессии. Однако не все методы аппроксимации функции одинаково хорошо подходят для использования в обучении с подкреплением. Наиболее развитые нейросетевые модели и статистические методы предполагают наличие статического обучающего набора, по которому производится несколько проходов. Но в обучении с подкреплением важно, чтобы обучение проходило в онлайн-режиме, в процессе взаимодействия агента с окружающей средой или ее моделью. Для этого необходимы методы, способные эффективно обучаться на основе постепенно поступающих данных. Кроме того, для обучения с подкреплением, вообще говоря, требуются методы аппроксимации функций, способные работать с нестационарной (изменяющейся во времени) целевой функцией. Например, в методах управления, основанных на ОИС (обобщенная итерация по стратегиям), мы часто стремимся обучить q_π , в то время как сама стратегия π изменяется. Даже если стратегия остается неизменной, выходные значения в обучающих при-

мерах будут нестационарными, если генерируются методами бутстрэппинга (ДП и TD-обучение). Методы, которые не могут работать в условиях такой нестационарности, плохо подходят для обучения с подкреплением.

9.2. ЦЕЛЕВАЯ ФУНКЦИЯ ПРЕДСКАЗАНИЯ ($\overline{VE}$)

До сих пор мы не определили явной целевой функции для предсказания. В табличном случае непрерывная мера качества предсказания была не нужна, поскольку обученная функция ценности могла в точности совпасть с истинной. Кроме того, обученные ценности каждого состояния были разделены – обновление одного состояния не влияло на другое. Но при настоящей аппроксимации обновление одного состояния сказывается на многих других, и невозможно сделать так, чтобы ценности всех состояний были правильными. По предположению, состояний гораздо больше, чем весов, поэтому если оценка для одного состояния оказывается точнее, то оценки других неизбежно становятся менее точными. Поэтому мы обязаны сказать, какие состояния для нас более важны. Мы должны определить распределение состояний $\mu(s) \geq 0$, $\sum_s \mu(s) = 1$, описывающее относительную важность ошибки в каждом состоянии s . Под ошибкой в состоянии s понимается квадрат разности между приближенной ценностью $\hat{v}(s, \mathbf{w})$ и истинной ценностью $v_\pi(s)$. Используя распределение μ для взвешивания состояний, мы получаем естественную целевую функцию, *среднеквадратическую ошибку ценности*, которую будем обозначать $\overline{VE}$:

$$\overline{VE}(\mathbf{w}) \doteq \sum_{s \in \mathcal{S}} \mu(s) [v_\pi(s) - \hat{v}(s, \mathbf{w})]^2. \quad (9.1)$$

Квадратный корень из величины $\overline{VE}$ дает грубую оценку отличия приближенных ценностей от истинных и нередко применяется при построении графиков. Часто в качестве $\mu(s)$ принимается доля времени, проведенная в состоянии s . При обучении с единой стратегией это называется *распределением с единой стратегией*; в этой главе мы только его и будем рассматривать. В непрерывных задачах распределение с единой стратегией является стационарным распределением при следовании стратегии π .

Распределение с единой стратегией в эпизодических задачах

В эпизодической задаче распределение с единой стратегией отличается тем, что зависит от выбора начальных состояний эпизодов. Обозначим $h(s)$ вероятность того, что эпизод начинается в состоянии s , а $\eta(s)$ – среднее количество временных шагов, проведенных в состоянии s в одном эпизоде. Считается, что шаг проведен в состоянии s , если эпизод начинается в s или если производится переход в s из предыдущего состояния $\bar{s}$:

$$\eta(s) = h(s) + \sum_{\bar{s}} \eta(\bar{s}) \sum_a \pi(a|\bar{s}) p(s|\bar{s}, a) \quad \text{для всех } s \in \mathcal{S}. \quad (9.2)$$

Из этой системы уравнений можно найти математическое ожидание количества посещений $\eta(s)$. Тогда распределение с единой стратегией – это

доля времени, проведенного в каждом состоянии, нормированная так, чтобы сумма всех долей была равна 1:

$$\mu(s) = \frac{\eta(s)}{\sum_{s'} \eta(s')} \quad \text{для всех } s \in \mathcal{S}. \quad (9.3)$$

Это естественный выбор для случая без обесценивания. Если обесценивание присутствует ($\gamma < 1$), то его следует рассматривать как форму завершения, и это можно сделать, просто включив множитель γ во второй член (9.2).

В обоих случаях, непрерывном и эпизодическом, поведение похоже, но в том, что касается аппроксимации, их необходимо рассматривать по отдельности, как мы неоднократно увидим в данной части книги. На этом формальное определение целевой функции обучения завершается.

Но не вполне очевидно, почему $\overline{VE}$ является правильной целевой функцией для обучения с подкреплением. Напомним, что конечной целью – причиной, по которой мы обучаем функцию ценности, – является нахождение оптимальной стратегии. Функция ценности, наилучшая с этой точки зрения, необязательна та, что минимизирует $\overline{VE}$. Но все равно не понятно, какая альтернатива могла бы быть полезнее для предсказания ценности. Пока что остановимся на $\overline{VE}$.

Идеально было бы найти *глобальный оптимум* $\overline{VE}$, т. е. вектор весов $\mathbf{w}^*$ такой, что $\overline{VE}(\mathbf{w}^*) \leq \overline{VE}(\mathbf{w})$ для всех возможных $\mathbf{w}$. Для простых аппроксимаций функций, например линейной, эта цель достижима, но для таких сложных, как нейронные сети и решающие деревья, это редко удается. Раз так, то мы хотим, чтобы сложные методы аппроксимации сходились хотя бы к *локальному оптимуму*, т. е. такому вектору весов, что $\overline{VE}(\mathbf{w}^*) \leq \overline{VE}(\mathbf{w})$ для всех $\mathbf{w}$ в некоторой окрестности $\mathbf{w}^*$. Хотя подобная гарантия слабо утешает, обычно это лучшее, что можно сделать для нелинейных методов аппроксимации функций, и зачастую этого достаточно. И все равно во многих представляющих интерес случаях обучения с подкреплением нет гарантии сходимости к оптимуму или хотя бы к решению, удаленному от оптимума на ограниченное расстояние. На самом деле некоторые методы вообще могут расходиться, т. е. $\overline{VE}$ стремится к бесконечности.

В последних двух разделах мы конспективно описали архитектуру, объединяющую широкий круг методов обучения с подкреплением для предсказания ценности с не менее широким кругом методов аппроксимации функций – эта архитектура состоит в использовании обновлений, вычисляемых методом первого вида, с целью генерирования обучающих примеров для метода второго вида. Мы также описали показатель качества $\overline{VE}$, который эти методы стремятся минимизировать. Спектр возможных методов аппроксимации слишком широк, чтобы рассмотреть их все, да и в любом случае о большинстве из них известно слишком мало, чтобы можно было с уверенностью рекомендовать их. По необходимости мы рассмотрим всего несколько возможностей. Далее в этой главе мы сосредоточимся на методах аппроксимации, основанных на вычислении градиентов, в частности на методах градиентного спуска. Мы решили остановиться именно на этих методах, отчасти потому, что считаем их особенно многообещающими и что на них можно проиллюстрировать важные теоретические проблемы, а отчасти потому, что они простые, а место в книге ограничено.

9.3. СТОХАСТИЧЕСКИЕ ГРАДИЕНТНЫЕ И ПОЛУГРАДИЕНТНЫЕ МЕТОДЫ

Теперь мы подробно рассмотрим один класс методов обучения для аппроксимации функций в задаче предсказания ценности, а именно методов, основанных на стохастическом градиентном спуске (СГС). Среди всех методов аппроксимации функций СГС-методы относятся к числу наиболее широко используемых и особенно хорошо подходят для онлайн-обучения с подкреплением.

В методах градиентного спуска вектор весов является вектором-столбцом с фиксированным числом вещественных элементов, $\mathbf{w} \doteq (w_1, w_2, \dots, w)^T$ ¹, а приближенная функция ценности $\hat{v}(s, \mathbf{w})$ является дифференцируемой функцией от $\mathbf{w}$ для всех $s \in \mathcal{S}$. Мы будем обновлять $\mathbf{w}$ на каждом из последовательности дискретных временных шагов $t = 0, 1, 2, 3, \dots$, поэтому понадобится обозначение $\mathbf{w}_t$ для вектора весов на каждом шаге. Пока что будем предполагать, что на каждом шаге наблюдается новый пример $S_t \mapsto v_\pi(S_t)$, который состоит из состояния S_t (быть может, выбранного случайно) и его истинной ценности при следовании стратегии. Это могли бы быть последовательные состояния взаимодействия с окружающей средой, однако пока мы не будем предполагать, что это так. Даже несмотря на то, что известны точные и правильные ценности $v_\pi(S_t)$ для каждого состояния S_t , задача остается сложной, поскольку ресурсы нашего аппроксиматора функций ограничены, а значит, ограничена и точность решения. В частности, в общем случае не существует вектора $\mathbf{w}$, который позволяет точно описать все состояния или хотя бы все примеры. А ведь решение должно еще и обобщаться на все остальные состояния, которые не встречались в примерах.

Предполагается, что состояния встречаются в примерах с одним и тем же распределением μ , относительно которого мы стремимся минимизировать $\overline{VE}$, определенное формулой (9.1). В данном случае хорошей стратегией является минимизация ошибки на предъявленных примерах. Методы стохастического градиентного спуска (СГС) добиваются этого, корректируя вектор весов после обработки каждого примера на небольшую величину в сторону наибольшего уменьшения ошибки в данном примере:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t - \frac{1}{2} \alpha \nabla [v_\pi(S_t) - \hat{v}(S_t, \mathbf{w}_t)]^2 \quad (9.4)$$

$$= \mathbf{w}_t + \alpha [v_\pi(S_t) - \hat{v}(S_t, \mathbf{w}_t)] \nabla \hat{v}(S_t, \mathbf{w}_t), \quad (9.5)$$

где α – положительный размер шага, а $\nabla f(\mathbf{w})$ для любой скалярной функции $f(\mathbf{w})$ векторного аргумента (в данном случае $\mathbf{w}$) обозначает вектор-столбец частных производных по элементам вектора:

$$\nabla f(\mathbf{w}) \doteq \left(\frac{\partial f(\mathbf{w})}{\partial w_1}, \frac{\partial f(\mathbf{w})}{\partial w_2}, \dots, \frac{\partial f(\mathbf{w})}{\partial w_d} \right)^T. \quad (9.6)$$

¹ Знаком T обозначается операция транспонирования, которая здесь необходима для преобразования вектора-строки, набранного в тексте горизонтально, в вектор-столбец. В этой книге обычно подразумеваются векторы-столбцы, если только они явно не записаны горизонтально без транспонирования.

Этот вектор частных производных является *градиентом* f по вектору $\mathbf{w}$. Методы СГС называют методами «градиентного спуска», потому что полный шаг в $\mathbf{w}_t$ пропорционален антриградиенту (градиенту со знаком минус) квадратичной ошибки для обработанного примера (9.4). Это то направление, в котором ошибка убывает наиболее быстро. Методы градиентного спуска называются «стохастическими», когда обновление, как в данном случае, применяется только к одному примеру, который может выбираться случайным образом. При наличии большого числа примеров и выборе небольших шагов общий эффект заключается в минимизации средней меры качества, например $\overline{VE}$.

Может показаться непонятным, почему СГС совершает только небольшие шаги в направлении градиента. Нельзя ли двигаться в этом направлении как можно дальше и полностью устранить ошибку в примере? Во многих случаях так поступать можно, но обычно это нежелательно. Напомним, что наша цель – не поиск функции ценности с нулевой ошибкой во всех состояниях, а лишь поиск аппроксимации, которая балансирует ошибки в разных состояниях. Если бы мы полностью устранили ошибку в примере на одном шаге, то не смогли бы найти такой баланс. На самом деле при доказательстве результатов о сходимости методов СГС предполагается, что параметр α убывает со временем. Если он убывает таким образом, что удовлетворяются стандартные условия стохастической аппроксимации (2.7), то гарантируется, что метод СГС (9.5) сходится к локальному оптимуму.

Теперь обратимся к случаю, когда выход $U_t \in \mathbb{R}$ t -го обучающего примера $S_t \mapsto U_t$ является не истинным значением $v_\pi(S_t)$, а некоторой его аппроксимацией, возможно, случайной. Например, U_t может быть зашумленным вариантом $v_\pi(S_t)$ или одной из целей бутстрэппинга, вычисленных с использованием $\hat{v}$, как упоминалось в предыдущем разделе. В таких случаях мы не можем произвести точное обновление (9.5), поскольку $v_\pi(S_t)$ неизвестно, но можем выполнить аппроксимацию, подставив U_t вместо $v_\pi(S_t)$. Это приводит к следующему общему методу СГС для предсказания функции ценности:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha[U_t - \hat{v}(S_t, \mathbf{w}_t)]\nabla \hat{v}(S_t, \mathbf{w}_t). \quad (9.7)$$

Если U_t является *несмещенной* оценкой, т. е. $\mathbb{E}[U_t | S_t = s] = v_\pi(S_t)$ для любого t , то $\mathbf{w}_t$ гарантированно сходится к локальному оптимуму при обычных условиях стохастической аппроксимации (2.7) и убывающем шаге α .

Например, предположим, что состояния в примерах сгенерированы на основе взаимодействия (или имитированного взаимодействия) с окружающей средой при следовании стратегии π . Поскольку истинной ценностью состояния является ожидаемый доход при старте из этого состояния, то цель Монте-Карло $U_t \doteq G_t$ по определению является несмещенной оценкой $v_\pi(S_t)$. При таком выборе общий метод СГС (9.7) сходится к локально оптимальной аппроксимации $v_\pi(S_t)$. Таким образом, вариант метода Монте-Карло для предсказания ценности состояния, основанный на градиентном спуске, гарантированно находит локально оптимальное решение. Полный псевдокод алгоритма приведен во врезке ниже.

Градиентный алгоритм Монте-Карло для оценивания $\hat{v} \approx v_\pi$

Вход: подлежащая оцениванию стратегия π

Вход: дифференцируемая функция $\hat{v}: \mathcal{S} \times \mathbb{R}^d \rightarrow \mathbb{R}$

Параметр алгоритма: размер шага $\alpha > 0$

Произвольно инициализировать веса функции ценности $\mathbf{w} \in \mathbb{R}^d$ (например, $\mathbf{w} = 0$)

Повторять бесконечно (для каждого эпизода):

Сгенерировать эпизод $S_0, A_0, R_1, S_1, A_1, \dots, R_T, S_T$, используя стратегию π

Повторять для каждого шага эпизода $t = 0, 1, \dots, T - 1$:

$\mathbf{w} \leftarrow \mathbf{w} + \alpha[G_t - \hat{v}(S_t, \mathbf{w})] \nabla \hat{v}(S_t, \mathbf{w})$

Мы не получим те же гарантии, если в качестве цели U_t в (9.7) будем использовать бутстрэппинговую оценку $v_\pi(S_t)$. Бутстрэппинговые цели, например n -шаговый доход $G_{t:t+n}$ или цель ДП $\sum_{a,s',r} \pi(a|S_t)p(s', r|S_t, a)[r + \hat{v}(s', \mathbf{w}_t)]$, зависят от текущего значения вектора весов $\mathbf{w}_t$, а это значит, что все они будут смещены и не смогут породить настоящий метод градиентного спуска. Заметим, что важнейший переход от (9.4) к (9.5) опирается на тот факт, что цель не зависит от $\mathbf{w}_t$. Этот переход был бы недопустим, если бы вместо $v_\pi(S_t)$ использовалась бутстрэппинговая оценка. В действительности бутстрэппинговые методы не относятся к методам градиентного спуска в чистом виде (Barnard, 1993). Они учитывают влияние изменения вектора весов $\mathbf{w}_t$ на оценку, но игнорируют его влияние на цель. Они включают только часть градиента и потому называются *полуградиентными методами*.

Хотя полуградиентные (бутстрэппинговые) методы не сходятся так же гарантированно, как градиентные, в некоторых важных случаях они все же сходятся надежно, например в линейном случае, который обсуждается в следующем разделе. Более того, они обладают важными преимуществами, благодаря которым часто являются однозначно предпочтительными. В частности, обычно обучение такими методами происходит гораздо быстрее, что мы и видели в главах 6 и 7. Другое преимущество – возможность непрерывного и онлайн-обучения, не дожидаясь завершения эпизода. Это позволяет использовать их в непрерывных задачах и получать выигрыш в вычислительном плане. Каноническим примером полуградиентного метода является TD(0), в котором в качестве цели выступает $U_t \doteq R_{t+1} + \hat{v}(S_{t+1}, \mathbf{w})$. Полный псевдокод этого метода приведен во врезке ниже.

Полуградиентный алгоритм TD(0) для оценивания $\hat{v} \approx v_\pi$

Вход: подлежащая оцениванию стратегия π

Вход: дифференцируемая функция $\hat{v}: \mathcal{S}^+ \times \mathbb{R}^d \rightarrow \mathbb{R}$ такая, что $\hat{v}(\text{terminal}, \cdot) = 0$

Параметр алгоритма: размер шага $\alpha > 0$

Произвольно инициализировать веса функции ценности $\mathbf{w} \in \mathbb{R}^d$ (например, $\mathbf{w} = 0$)

Повторять для каждого эпизода:

Инициализировать S

Повторять для каждого шага эпизода:
 Выбрать $A \sim \pi(\cdot | S)$
 Предпринять действие A , наблюдать R, S'
 $\mathbf{w} \leftarrow \mathbf{w} + \alpha [R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})] \nabla \hat{v}(S, \mathbf{w})$
 $S \leftarrow S'$
 пока S не является заключительным

Агрегирование состояний – простая форма обобщения аппроксимации функций, в которой состояния объединяются в группы с одной оценкой ценности (одним элементом вектора весов $\mathbf{w}$) для каждой группы. Ценность состояния оценивается как элемент, соответствующий ее группе; при обновлении состояния изменяется только этот элемент. Агрегирование состояний – частный случай алгоритма СГС (9.7), в котором градиент $\nabla \hat{v}(S_t, \mathbf{w}_t)$ равен 1 для элемента, соответствующего группе состояния S_t , и 0 для всех остальных элементов.

Пример 9.1. Агрегирование состояний для случайного блуждания с 1000 состояний. Рассмотрим вариант задачи о случайном блуждании с 1000 состояний (примеры 6.2 и 7.1 на стр. 158 и 179 соответственно). Состояния пронумерованы от 1 до 1000 слева направо, а все эпизоды начинаются в центре, в состоянии 500. Переходы возможны из текущего состояния в одно из 100 соседних состояний слева и в одно из 100 соседних состояний справа, все они равновероятны. Разумеется, если текущее состояние близко к краю, то с соответствующей стороны может быть меньше 100 состояний. В таком случае с вероятностью, равной сумме вероятностей всех переходов в несуществующие соседние состояния, имеет место завершение блуждания на этом краю (т. е. в состоянии 1 вероятность завершения на левом краю равна 0.5, а в состоянии 950 вероятность завершения на правом краю равна 0.25). Как обычно, за завершение на левом краю назначается вознаграждение -1 , а на правом – вознаграждение $+1$. Вознаграждение за все остальные переходы равно 0. Этот пример будет неоднократно использован в данном разделе.

На рис. 9.1 показана истинная функция ценности v_π для этой задачи. Она почти прямая, но немного искривляется в сторону горизонтали в последних 100 состояниях с каждого края. Показана также приближенная функция ценности, обученная градиентным алгоритмом Монте-Карло с агрегированием состояний после 100 000 эпизодов с размером шага $\alpha = 2 \times 10^{-5}$. Для агрегирования состояний все 1000 состояний были разбиты на 10 групп по 100 состояний в каждой (состояния 1–100 в одной группе, 101–200 – в другой и т. д.). Присутствующая на рисунке зубчатость – типичное последствие агрегирования состояний; в каждой группе приближенное значение постоянно, но резко изменяется при переходе от одной группы к другой. Эти приближенные ценности близки к глобальному минимуму $\bar{VE}$ (9.1).

Некоторые особенности приближенных ценностей лучше оценить, сославшись на распределение состояний μ для этой задачи, показанное в нижней части рисунка (шкала справа). Центральное состояние 500 является первым в каждом эпизоде, но оно редко посещается еще раз. В среднем на стартовое состояние приходится 1.37 % временных шагов. Вторыми по частоте посещаемости являются состояния, достижимые из стартового за один шаг, на каждое из них приходится 0.17 % шагов. Далее распределение μ спадает почти линейно, на крайние состоя-

ния 1 и 1000 приходится примерно 0.0147 % времени. Наиболее наглядно эффект этого распределения проявляется в левых группах, ценности которых, очевидно, сдвинуты вверх по сравнению с невзвешенным средним истинных ценностей состояний в группе, и в правых группах, ценности которых сдвинуты вниз. Это объясняется тем, что состояния в этих областях имеют наибольшую асимметрию весов согласно распределению μ . Например, в самой левой группе вес состояния 100 в три с лишним раза больше, чем вес состояния 1. Поэтому оценка этой группы смещена в сторону истинной ценности состояния 100, которая больше истинной ценности состояния 1.

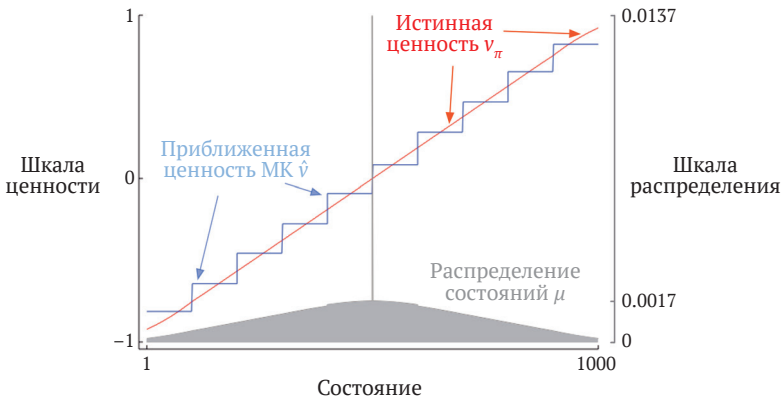


Рис. 9.1 ❖ Аппроксимация функции посредством агрегирования состояний в задаче о случайном блуждании с 1000 состояний градиентным алгоритмом Монте-Карло

9.4. ЛИНЕЙНЫЕ МЕТОДЫ

Одним из наиболее важных частных случаев аппроксимации функций является случай, когда приближенная функция $\hat{v}(\cdot, \mathbf{w})$ линейна относительно вектора весов $\mathbf{w}$. Каждому состоянию s соответствует вещественный вектор $\mathbf{x}(s) \doteq (x_1(s), x_2(s), \dots, x_d(s))^T$ с таким же числом элементов, как у вектора $\mathbf{w}$. Линейные методы аппроксимируют функцию ценности состояний скалярным произведением $\mathbf{w}$ и $\mathbf{x}(s)$:

$$\hat{v}(s, \mathbf{w}) \doteq \mathbf{w}^T \mathbf{x}(s) \doteq \sum_{i=1}^d w_i x_i(s). \quad (9.8)$$

В этом случае говорят, что приближенная функция ценности *линейно зависит от весов*, или просто *линейна*.

Вектор $\mathbf{x}(s)$ называется *вектором признаков*, представляющим состояние s . Каждый элемент $x_i(s)$ вектора $\mathbf{x}(s)$ является значением функции $x_i: \mathcal{S} \rightarrow \mathbb{R}$. Мы рассматриваем каждый *признак* как одну из таких функций целиком и называем *признаком* s его значение для состояния s . Для линейных методов признаки являются *базисными функциями*, потому что образуют линейный базис множества приближенных функций. Конструирование d -мерных векторов признаков для

представления состояний – то же самое, что выбор множества d базисных функций. Признаки можно определить многими способами, некоторые возможности будут рассмотрены в следующих разделах.

Для линейной аппроксимации функции применение СГС-обновлений выглядит естественно. В этом случае градиент приближенной функции ценности по $\mathbf{w}$ имеет вид:

$$\nabla \hat{v}(s, \mathbf{w}) = \mathbf{x}(s).$$

Таким образом, в линейном случае общее СГС-обновление (9.7) принимает особенно простую форму:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha[U_t - \hat{v}(S_t, \mathbf{w}_t)]\mathbf{x}(S_t).$$

В силу своей простоты случай линейного СГС легко поддается математическому анализу. Почти все полезные результаты о сходимости для систем обучения всех видов относятся к линейным (или более простым) методам аппроксимации функций.

В частности, в линейном случае существует только один оптимум (в вырожденных случаях – множество одинаково хороших оптимумов), поэтому любой метод, который гарантированно сходится к локальному оптимуму или близкому к нему решению, автоматически сходится к глобальному оптимуму или почти оптимуму. Например, градиентный алгоритм Монте-Карло, описанный в предыдущем разделе, сходится к глобальному оптимуму $\bar{V}_E$ в случае линейной аппроксимации функций, если α уменьшается со временем и удовлетворяет обычным условиям.

Полуградиентный алгоритм TD(0), представленный в предыдущем разделе, также сходится в случае линейной аппроксимации, но это не следует из общих результатов, относящихся к СГС; необходимо доказывать отдельную теорему. При этом вектор весов, к которому сходится алгоритм, является не глобальным оптимумом, а точкой, расположенной вблизи локального оптимума. Этот важный случай имеет смысл рассмотреть более подробно, особенно в непрерывном случае. Обновление в момент t имеет вид:

$$\begin{aligned} \mathbf{w}_{t+1} &\doteq \mathbf{w}_t + \alpha(R_{t+1} + \gamma \mathbf{w}_t^\top \mathbf{x}_{t+1} - \mathbf{w}_t^\top \mathbf{x}_t) \mathbf{x}_t \\ &= \mathbf{w}_t + \alpha(R_{t+1} \mathbf{x}_t - \mathbf{x}_t(\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\top \mathbf{w}_t) \end{aligned} \quad (9.9)$$

где мы воспользовались сокращенной нотацией $\mathbf{x}_t = \mathbf{x}(S_t)$. После того как система перешла в установившееся состояние, для любого $\mathbf{w}_t$ следующий ожидаемый вектор весов можно записать в виде:

$$\mathbb{E}[\mathbf{w}_{t+1} | \mathbf{w}_t] = \mathbf{w}_t + \alpha(\mathbf{b} - \mathbf{A}\mathbf{w}_t), \quad (9.10)$$

где

$$\mathbf{b} \doteq \mathbb{E}[R_{t+1} \mathbf{x}_t] \in \mathbb{R}^d \quad \text{и} \quad \mathbf{A} \doteq \mathbb{E}[\mathbf{x}_t(\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\top] \in \mathbb{R}^d \times \mathbb{R}^d. \quad (9.11)$$

Из (9.10) очевидно, что если система вообще сходится, то она должна сходиться к вектору весов $\mathbf{w}_{\text{TD}}$, для которого

$$\begin{aligned}
& \mathbf{b} - \mathbf{A}\mathbf{w}_{TD} = 0 \\
\Rightarrow & \quad \mathbf{b} = \mathbf{A}\mathbf{w}_{TD} \\
\Rightarrow & \quad \mathbf{w}_{TD} \doteq \mathbf{A}^{-1}\mathbf{b}.
\end{aligned} \tag{9.12}$$

Эта величина называется *неподвижной точкой TD*. На самом деле линейный полугradientный алгоритм TD(0) сходится к этой точке. Доказательство сходимости и замечания об обратимости матрицы см. во врезке ниже.

Доказательство сходимости линейного TD(0)

Какие свойства гарантируют сходимость линейного алгоритма TD(0) (9.9)? Некоторые идеи могут возникнуть, если переписать (9.10) в виде:

$$\mathbb{E}[\mathbf{w}_{t+1} | \mathbf{w}_t] = (\mathbf{I} - \alpha \mathbf{A})\mathbf{w}_t + \alpha \mathbf{b}. \tag{9.13}$$

Заметим, что на матрицу $\mathbf{A}$ умножается вектор весов $\mathbf{w}_t$, а не $\mathbf{b}$; для сходимости важна только $\mathbf{A}$. Чтобы развить интуицию, рассмотрим частный случай, когда $\mathbf{A}$ – диагональная матрица. Если какой-нибудь диагональный элемент отрицателен, то соответствующий диагональный элемент матрицы $\mathbf{I} - \alpha \mathbf{A}$ будет больше 1, и соответствующий элемент вектора $\mathbf{w}_t$ будет усилен, что после повторения станет причиной расходимости. С другой стороны, если все диагональные элементы $\mathbf{A}$ положительны, то α можно выбрать меньшим, чем 1, поделенная на наибольший из них, так что $\mathbf{I} - \alpha \mathbf{A}$ будет диагональной матрицей, все диагональные элементы которой – числа от 0 до 1. В таком случае первый член в уравнении обновления сжимает $\mathbf{w}_t$, что гарантирует устойчивость. В общем случае $\mathbf{w}_t$ будет уменьшаться, приближаясь к нулю, если матрица $\mathbf{A}$ *положительно определенная*, т. е. $\mathbf{y}^T \mathbf{A} \mathbf{y} > 0$ для любого вещественного вектора $\mathbf{y} \neq 0$. Положительная определенность также гарантирует существование обратной матрицы $\mathbf{A}^{-1}$.

Для линейного алгоритма TD(0) в непрерывном случае с $\gamma < 1$ матрицу $\mathbf{A}$ из формулы (9.11) можно записать в виде:

$$\begin{aligned}
\mathbf{A} &= \sum_s \gamma(s) \sum_a \pi(a|s) \sum_{r,s'} p(r,s'|s,a) \mathbf{x}(s)(\mathbf{x}(s) - \gamma \mathbf{x}(s'))^T \\
&= \sum_s \gamma(s) \sum_{s'} p(s'|s) \mathbf{x}(s)(\mathbf{x}(s) - \gamma \mathbf{x}(s'))^T \\
&= \sum_s \gamma(s) \mathbf{x}(s) \left(\mathbf{x}(s) - \gamma \sum_{s'} p(s'|s) \mathbf{x}(s') \right)^T \\
&= \mathbf{X}^T \mathbf{D} (\mathbf{I} - \gamma \mathbf{P}) \mathbf{X},
\end{aligned}$$

где $\mu(s)$ – стационарное распределение при стратегии π , $p(s'|s)$ – вероятность перехода из s в s' при стратегии π , $\mathbf{P}$ – матрица $|\mathcal{S}| \times |\mathcal{S}|$ этих вероятностей, $\mathbf{D}$ – диагональная матрица $|\mathcal{S}| \times |\mathcal{S}|$ с элементами $\mu(s)$ на диагонали, а $\mathbf{X}$ – матрица $|\mathcal{S}| \times d$, строками которой являются векторы $\mathbf{x}(s)$. Отсюда очевидно, что внутренняя матрица $\mathbf{D}(\mathbf{I} - \gamma \mathbf{P})$ – ключ к установлению положительной определенности $\mathbf{A}$.

Для матрицы такого вида положительная определенность гарантирована, если сумма элементов в каждом столбце неотрицательна. Это было доказано

в работе Sutton (1988, стр. 27) на основе двух ранее установленных теорем. Одна теорема утверждает, что любая матрица $\mathbf{M}$ положительно определена тогда и только тогда, когда положительно определена симметричная матрица $\mathbf{S} = \mathbf{M} + \mathbf{M}^T$ (Sutton 1988, приложение). Вторая теорема утверждает, что любая симметричная вещественная матрица $\mathbf{S}$ является положительно определенной, если каждый ее диагональный элемент положителен и больше суммы абсолютных величин соответствующих внедиагональных элементов (Varga 1962, стр. 23). Для нашей ключевой матрицы $\mathbf{D}(\mathbf{I} - \gamma\mathbf{P})$ диагональные элементы положительны, а внедиагональные отрицательны, поэтому нужно только показать, что сумма элементов в каждой строке плюс сумма элементов в соответствующем столбце положительна. Все суммы по строкам положительны, потому что $\mathbf{P}$ – стохастическая матрица и $\gamma < 1$. Таким образом, осталось показать, что суммы по столбцам неотрицательны. Заметим, что вектор-строка, состоящий из сумм элементов в каждом столбце произвольной матрицы $\mathbf{M}$, можно записать в виде $\mathbf{1}^T \mathbf{M}$, где $\mathbf{1}$ – вектор-столбец, все элементы которого равны 1. Обозначим μ $|S|$ -мерный вектор $\mu(s)$, где $\mu = \mathbf{P}^T \mu$ в силу стационарности распределения μ . Суммы элементов в столбцах нашей ключевой матрицы тогда равны

$$\begin{aligned} \mathbf{1}^T \mathbf{D}(\mathbf{I} - \gamma\mathbf{P}) &= \mu^T (\mathbf{I} - \gamma\mathbf{P}) \\ &= \mu^T - \gamma \mu^T \mathbf{P} \\ &= \mu^T - \gamma \mu^T \quad (\text{поскольку } \mu \text{ – стационарное распределение}) \\ &= (1 - \gamma) \mu^T, \end{aligned}$$

и все элементы этого вектора положительны. Таким образом, ключевая матрица и ее матрица $\mathbf{A}$ положительно определены, и алгоритм TD(0) с единой стратегией устойчив. (Для доказательства сходимости с вероятностью 1 нужны дополнительные условия и определенный характер убывания α со временем.)

Доказано также (в непрерывном случае), что в неподвижной точке TD величина $\overline{\text{VE}}$ не превосходит минимально возможную ошибку, умноженную на константу:

$$\overline{\text{VE}}(\mathbf{w}_{\text{TD}}) \leq \frac{1}{1 - \gamma} \min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w}). \quad (9.14)$$

Иначе говоря, асимптотическая ошибка TD-метода не более чем в $1/(1 - \gamma)$ раз больше наименьшей возможной ошибки, которой в пределе достигает метод Монте-Карло. Поскольку значение γ часто близко к единице, этот повышающий коэффициент может быть довольно большим, поэтому TD-методу потенциально угрожает асимптотическая потеря точности. С другой стороны, вспомним, что TD-методы зачастую обладают гораздо меньшей дисперсией, чем методы Монте-Карло, а значит, работают быстрее, как мы видели в главах 6 и 7. Какой метод лучше, зависит от природы задачи и аппроксимации, а также от продолжительности обучения.

Граница, аналогичная (9.14), имеет место и для других бутстрэппинговых методов с единой стратегией. Например, линейный полугradientный метод ДП (уравнение 9.7 с $U_t \doteq \sum_a \pi(a|S_t) \sum_{s', p} p(s', r|S_t, a) [r + \gamma \hat{v}(s', \mathbf{w}_t)]$), в котором обновления

производятся согласно распределению с единой стратегией, также сходится к неподвижной точке TD. Одношаговые полугradientные методы ценности действий, например полугradientный Sarsa(0), рассматриваемый в следующей главе, сходятся к аналогичной неподвижной точке, и для них имеет место аналогичная граница. Для эпизодических задач существует несколько отличающаяся, но похожая граница (см. Bertsekas and Tsitsiklis, 1996). Существует также несколько технических условий на вознаграждения, признаки и убывание размера шага, которые мы здесь опускаем. Подробности см. в оригинальной статье (Tsitsiklis and Van Roy, 1997).

Для этих результатов о сходимости критически важен тот факт, что состояния обновляются согласно распределению с единой стратегией. Для других распределений обновлений бутстрэппинговые методы с применением аппроксимации функций могут расходиться. Примеры такого рода и обсуждение возможных методов решения см. в главе 11.

Пример 9.2. Бутстрэппинг в случайном блуждании с 1000 состояний. Агрегирование состояний – частный случай аппроксимации линейной функцией, поэтому вернемся к случайному блужданию с 1000 состояний, чтобы проиллюстрировать некоторые наблюдения, сделанные в этой главе. На рис. 9.2 слева показана окончательная функция, обученная полугradientным алгоритмом TD(0) (стр. 244) с таким же агрегированием состояний, как в примере 9.1. Мы видим, что близкая к асимптотической TD-аппроксимация на самом деле отстоит дальше от истинных ценностей, чем аппроксимация методом Монте-Карло, показанная на рис. 9.1.

Тем не менее TD-методы все же обладают большими потенциальными преимуществами в части скорости обучения и являются обобщением методов Монте-Карло, как было установлено при подробном изучении n -шаговых TD-методов в главе 7. На рис. 9.2 справа показаны результаты n -шагового полугradientного TD-метода с агрегированием состояний случайного блуждания с 1000 состояний; они поразительно напоминают результаты, полученные ранее табличными методами для случайного блуждания с 19 состояниями (рис. 7.2). Чтобы получить такие количественно сходные результаты, мы перешли к 20 группам по 50 состояний каждая. Двадцать групп – это примерно столько же, сколько 19 состояний в табличной задаче. Напомним еще, что переходы допустимы в состояние, отстоящее от текущего не более чем на 100 состояний вправо или влево. Это означает, что длина типичного перехода составляет 50 состояний вправо или влево, что количественно аналогично переходу на одно состояние в табличной задаче с 19 состояниями. Для большей корректности сравнения мы используем здесь тот же показатель качества – невзвешенное среднее среднеквадратической ошибки по всем состояниям и первым 10 эпизодам, – а не целевую функцию $\overline{VE}$, которая вообще-то больше подходит для аппроксимации функций.

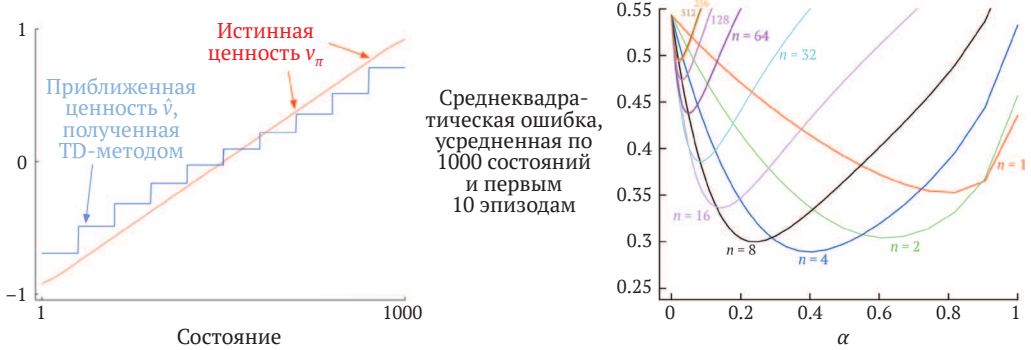


Рис. 9.2 ❖ Бутстрэппинг с агрегированием состояний в задаче о случайном блуждании с 1000 состояний. **Слева:** асимптотические ценности для полугradientного TD хуже, чем для метода Монте-Карло на рис. 9.1. **Справа:** результаты n -шаговых методов с агрегированием состояний поразительно напоминают полученные для табличного представления (см. рис. 7.2). Данные усреднены по 100 прогонам

Полугradientный n -шаговый TD-алгоритм, использованный в примере выше, является естественным обобщением табличного n -шагового TD-алгоритма, представленного в главе 7, на полугradientную аппроксимацию функции. Его псевдокод приведен во врезке ниже.

n -шаговый полугradientный алгоритм TD для оценивания $\hat{v} \approx v_\pi$

Вход: подлежащая оцениванию стратегия π

Вход: дифференцируемая функция $\hat{v}: \mathcal{S}^+ \times \mathbb{R}^d \rightarrow \mathbb{R}$ такая, что $\hat{v}(\text{terminal}, \cdot) = 0$

Параметры алгоритма: размер шага $\alpha > 0$, положительное целое число n

Произвольно инициализировать веса функции ценности $\mathbf{w}$ (например, $\mathbf{w} = 0$)

Все операции сохранения и доступа (S_t и R_t) принимают индекс по модулю $n + 1$

Повторять для каждого эпизода:

Инициализировать и сохранить $S_0 \neq \text{terminal}$

$T \leftarrow \infty$

Повторять для $t = 0, 1, 2, \dots$:

Если $t < T$, то:

Предпринять действие согласно $\pi(\cdot | S_t)$

Наблюдать и сохранить следующее вознаграждение как R_{t+1}

и следующее состояние как S_{t+1}

Если S_{t+1} – заключительное состояние, то $T \leftarrow t + 1$

$\tau \leftarrow t - n + 1$ (τ – момент времени, для которого обновляется оценка)

Если $\tau \geq 0$:

$$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$$

$$\text{Если } \tau + n < T, \text{ то } G \leftarrow G + \gamma^n \hat{v}(S_{\tau+n}, \mathbf{w}) \quad (G_{\tau:\tau+n})$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha [R - \hat{v}(S_\tau, \mathbf{w})] \nabla \hat{v}(S_\tau, \mathbf{w})$$

$$S \leftarrow S'$$

пока не $\tau = T - 1$

Основное уравнение этого алгоритма, аналогичное (7.2), имеет вид:

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha[G_{t:t+n} - \hat{v}(S_t, \mathbf{w}_{t+n-1})]\nabla\hat{v}(S_t, \mathbf{w}_{t+n-1}), \quad 0 \leq t < T, \quad (9.15)$$

где n -шаговый доход обобщает формулу (7.1):

$$G_{t:t+n} \doteq R_{t+1} + R_{t+2} + \dots + \gamma^{n-1}R_{t+n} + \gamma^n\hat{v}(S_{t+n}, \mathbf{w}_{t+n-1}), \quad 0 \leq t \leq T - n. \quad (9.16)$$

Упражнение 9.1. Покажите, что табличные методы типа тех, что представлены в первой части книги, – частный случай линейной аппроксимации функции. Каковы в этом случае векторы признаков? $\square$

9.5. КОНСТРУИРОВАНИЕ ПРИЗНАКОВ ДЛЯ ЛИНЕЙНЫХ МЕТОДОВ

Линейные методы интересны прежде всего тем, что для них гарантирована сходимость, но также и тем, что на практике они могут быть очень эффективны с точки зрения времени вычислений и потребления памяти. Так это или не так, в огромной степени зависит от того, как состояния представлены в виде признаков, и именно этим вопросом мы займемся в этом длинном разделе. Выбор признаков, отвечающих задаче, – важный способ добавить априорную предметную информацию в систему обучения с подкреплением. Интуитивно понятно, что признаки должны соответствовать тем осям пространства состояний, вдоль которых возможно обобщение. Например, если речь идет о геометрических объектах, то можно завести признаки для формы, цвета, размера или назначения. Если же мы оцениваем состояния подвижного робота, то, наверное, понадобятся признаки для местоположения, уровня заряда аккумулятора, показаний ультразвукового локатора и т. д.

Ограничение линейной формы заключается в том, что она не позволяет учесть зависимости между признаками, например что наличие признака i хорошо только в отсутствие признака j . Так, в задаче балансирования стержня (пример 3.4) большая угловая скорость может быть хорошим или плохим признаком в зависимости от угла. Если угол большой, то высокая угловая скорость грозит падением – это плохое состояние, а если небольшой, то высокая угловая скорость означает, что стержень выправляется – хорошее состояние. Эту ситуацию нельзя было бы представить линейной функцией ценности, если бы угол и угловая скорость кодировались отдельными признаками. Вместо этого дополнительно необходимы признаки, описывающие комбинации этих двух базовых характеристик состояния. В следующих разделах мы рассмотрим различные общие способы решить эту задачу.

9.5.1. Полиномы

Во многих задачах состояния первоначально выражаются в виде чисел, например местоположения и скорости в задаче о балансировании стержня (пример 3.4), количества машин на каждой парковке в задаче об аренде автомобилей (пример 4.1) или капитала игрока в задаче игрока (пример 4.2). В таких задачах аппроксимация функций для обучения с подкреплением имеет много общего с хорошо знакомы-

ми интерполяцией и регрессией. Различные семейства признаков, часто встречающиеся в задачах интерполяции и регрессии, можно использовать и в обучении с подкреплением. Полиномы составляют одно из самых простых семейств признаков для интерполяции и регрессии. Хотя базовые полиномиальные признаки, которые мы здесь обсуждаем, работают в области обучения с подкреплением не так хорошо, как признаки других типов, они удобны в качестве введения в силу своей простоты и привычности.

В качестве примера предположим, что в задаче обучения с подкреплением имеется два состояния с двумя числовыми осями. Для репрезентативного состояния s обозначим эти два числа $s_1 \in \mathbb{R}$ и $s_2 \in \mathbb{R}$. Можно было бы просто представлять s двумя компонентами его состояния, так что $\mathbf{x}(s) = (s_1, s_2)^T$, но тогда мы не сможем учесть зависимости между ними. Кроме того, если s_1 и s_2 одновременно равны нулю, то приближенное значение тоже должно было бы быть равно нулю. Оба эти ограничения можно обойти, если представлять s четырехмерным вектором признаков $\mathbf{x}(s) = (1, s_1, s_2, s_1 s_2)^T$. Первый признак 1 позволяет представлять аффинные функции от исходных числовых компонент состояния, а последний признак, произведение $s_1 s_2$, позволяет учесть зависимости. Можно также использовать векторы признаков большей размерности, например $\mathbf{x}(s) = (1, s_1, s_2, s_1 s_2, s_1^2, s_2^2, s_1 s_2^2, s_1^2 s_2, s_1^2 s_2^2)^T$, чтобы учесть более сложные зависимости. Такие векторы признаков допускают аппроксимации произвольными квадратичными функциями от числовых компонент состояния – пусть даже аппроксимация по-прежнему линейно зависит от весов, подлежащих обучению. Обобщая этот пример с двух на k чисел, мы можем представить очень сложные зависимости между компонентами состояний задачи.

Предположим, что каждому состоянию s соответствует k чисел $s_1, s_2, \dots, s_k$, $s_i \in \mathbb{R}$. Для этого k -мерного пространства состояний признаки x_i вида

$$x_i(s) = \prod_{j=1}^k s_j^{c_{i,j}}, \quad (9.17)$$

где $c_{i,j}$ – целые числа, принадлежащие множеству $\{0, 1, \dots, n\}$, и $n \geq 0$ – целое число, составляют полиномиальный базис порядка n , содержащий $(n + 1)^k$ элементов.

Чем больше порядок полиномиального базиса, тем выше точность аппроксимации сложных функций. Но поскольку количество признаков в полиномиальном базисе порядка n экспоненциально возрастает с увеличением размерности k естественного пространства состояний (если $n > 0$), то обычно для аппроксимации необходимо выбирать некоторое их подмножество. Это можно сделать, имея априорные представления о характере аппроксимируемых функций. Некоторые автоматизированные методы отбора, разработанные для полиномиальной регрессии, можно адаптировать для работы в условиях инкрементной и нестационарной природы обучения с подкреплением.

Упражнение 9.2. Почему формула (9.17) определяет $(n + 1)^k$ различных признаков для размерности k ? □

Упражнение 9.3. При каких n и $c_{i,j}$ порождаются векторы признаков $\mathbf{x}(s) = (1, s_1, s_2, s_1 s_2, s_1^2, s_2^2, s_1 s_2^2, s_1^2 s_2, s_1^2 s_2^2)^T$? $\square$

9.5.2. Базис Фурье

Еще один метод линейной аппроксимации основан на достопочтенных рядах Фурье, позволяющих выразить периодическую функцию в виде взвешенной суммы базисных функций (признаков), в роли которых выступают синусоиды и косинусоиды различной частоты. (Функция f называется периодической, если $f(x) = f(x + \tau)$ для всех x и некоторого периода τ .) Ряды Фурье и преобразование Фурье широко применяются в прикладных дисциплинах отчасти потому, что если аппроксимируемая функция известна, то веса базисных функций определяются простыми формулами, а также потому, что, взяв достаточное количество базисных функций, любую функцию можно аппроксимировать со сколь угодно высокой точностью. В обучении с подкреплением, где аппроксимируемая функция неизвестна, базисные функции Фурье представляют интерес, потому что их легко использовать и они дают хорошие результаты при решении широкого круга задач.

Сначала рассмотрим одномерный случай. Обычное представление одномерной функции с периодом τ рядом Фурье имеет вид линейной комбинации синусов и косинусов с периодами, кратными τ (иными словами, их частоты – целые кратные основной частоты $1/\tau$). Если же нас интересует аппроксимация непериодической функции, определенной на ограниченном интервале, то можно использовать в качестве признаков базис Фурье для периода τ , равного длине этого интервала. Тогда интересующая нас функция будет сужением периодической линейной комбинации синусов и косинусов на один период.

Далее, если положить τ равным удвоенной длине интервала и ограничиться аппроксимацией на половине интервала $[0, \tau/2]$, то можно будет использовать только косинусоидальные признаки. Это возможно, потому что любую четную функцию, т. е. любую функцию, симметричную относительно начала координат, можно представить в косинусоидальном базисе. Следовательно, любую функцию на полупериоде $[0, \tau/2]$ можно аппроксимировать с произвольной точностью, пользуясь только косинусоидальными признаками. (Выражение «любая функция» не вполне корректно, потому что функция должна обладать определенными математическими свойствами, но эти технические детали мы опустим.) Можно вместо этого использовать только синусоидальные признаки, линейная комбинация которых всегда дает нечетную функцию, т. е. функцию, антисимметричную относительно начала координат. Но в общем случае рекомендуется использовать косинусоидальные признаки, потому что «четные на полупериоде» функции проще аппроксимировать, чем «нечетные на полупериоде», т. к. последние часто бывают разрывными в начале координат. Конечно, это не означает, что использовать одновременно синусы и косинусы для аппроксимации на интервале $[0, \tau/2]$ запрещено, и при некоторых условиях это даже выгодно.

Следуя этой логике и полагая $\tau = 2$, так чтобы признаки были определены на полуинтервале $[0, 1]$, мы получаем одномерный косинусоидальный базис Фурье порядка n , состоящий из $n + 1$ признаков:

$$x_i(s) = \cos(i\pi s), \quad s \in [0, 1],$$

для $i = 0, \dots, n$. На рис. 9.3 показаны одномерные косинусоидальные признаки Фурье x_i для $i = 1, 2, 3, 4$. Признак x_0 – постоянная функция.

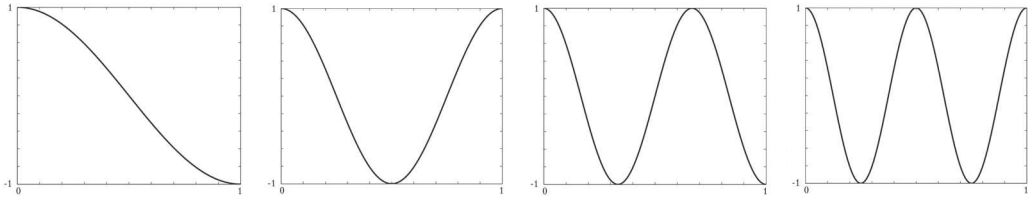


Рис. 9.3 ❖ Признаки в одномерном косинусоидальном базисе Фурье x_i , $i = 1, 2, 3, 4$, для аппроксимации функций на интервале $[0, 1]$. Взято из работы Konidaris et al. (2011)

То же самое рассуждение применимо к аппроксимации рядом Фурье из косинусов в многомерном случае (см. описание во врезке ниже).

Предположим, что каждому состоянию s соответствует вектор из k чисел $\mathbf{s} = (s_1, s_2, \dots, s_k)^T$, где каждое $s_i \in [0, 1]$. i -й признак в косинусоидальном базисе Фурье порядка n можно записать в виде:

$$x_i(s) = \cos(\pi \mathbf{s}^T \mathbf{c}^i), \quad (9.18)$$

где $\mathbf{c}^i = (c_1^i, c_2^i, \dots, c_k^i)^T$, $c_j^i \in \{0, 1, \dots, n\}$ для $j = 1, \dots, k$ и $i = 0, \dots, (n+1)^k$. Эта формула определяет признак для каждого из $(n+1)^k$ возможных целочисленных векторов $\mathbf{c}^i$. Скалярное произведение $\mathbf{s}^T \mathbf{c}^i$ сопоставляет целое число из множества $\{0, 1, \dots, n\}$ каждому измерению $\mathbf{s}$. Как и в одномерном случае, это целое число определяет частоту признака по этому измерению. Конечно, признаки можно сдвигать и масштабировать для адаптации к ограниченному пространству состояний конкретного приложения.

В качестве примера рассмотрим случай $k = 2$, в котором $\mathbf{s} = (s_1, s_2)^T$, где каждый $\mathbf{c}^i = (c_1^i, c_2^i)^T$. На рис. 9.4 показано шесть избранных косинусоидальных признаков Фурье, каждый из которых помечен определяющим его вектором $\mathbf{c}^i$ (s_1 – горизонтальная ось, а $\mathbf{c}^i$ показан в виде вектора-строки с опущенным индексом i). Ноль на любом месте в $\mathbf{c}$ означает, что признак постоянен вдоль этой компоненты состояния. Поэтому если $\mathbf{c} = (0, 0)^T$, то признак постоянен по обоим измерениям; если $\mathbf{c} = (c_1, 0)^T$, то признак постоянен по второму измерению и может меняться по первому с частотой, зависящей от c_1 , и аналогично для $\mathbf{c} = (0, c_2)^T$. Если $\mathbf{c} = (c_1, c_2)^T$ и ни одна компонента c_j не равна 0, то признак изменяется вдоль обоих измерений и представляет зависимость между двумя переменными состояния. Значения c_1 и c_2 определяют частоты вдоль каждого измерения, а их отношение задает направление зависимости.

При использовании косинусоидальных признаков Фурье совместно с алгоритмом обучения, например (9.7), полугradientным TD(0) или полугradientным Sarsa, иногда полезно брать различный размер шага для каждого признака. Обозначим α базовый размер шага, тогда в работе Konidaris, Osentoski, and Thomas

(2011) предлагается брать для признака x_i шаг $\alpha_i = \alpha / \sqrt{(c_1^i)^2 + \dots + (c_k^i)^2}$ (за исключением случая, когда все $c_j^i = 0$, а в этом случае полагать $\alpha_i = \alpha$).

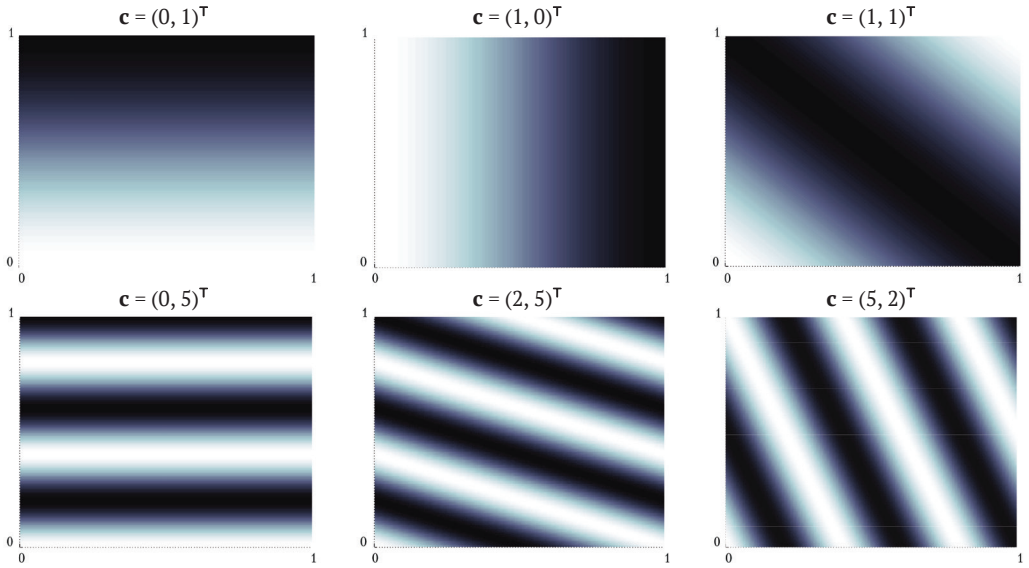


Рис. 9.4 ❖ Шесть избранных косинусоидальных признаков Фурье, каждый из которых помечен определяющим его вектором $\mathbf{c}^i$ (s_1 соответствует горизонтальной оси, а в $\mathbf{c}^i$ индекс i опущен). Заимствовано из работы Konidaris et al. (2011)

Сочетание косинусоидальных признаков Фурье с алгоритмом Sarsa может давать хорошие результаты по сравнению с другими наборами базисных функций, в т. ч. полиномиальных и радиально-базисных. Неудивительно, однако, что признаки Фурье испытывают проблемы с разрывными функциями, потому что трудно избежать «опоясывания» точек разрыва, если только не включать очень высокочастотных базисных функций.

Количество признаков в базисе Фурье порядка n растет экспоненциально с увеличением размерности пространства состояний, но если эта размерность достаточно мала (например, $k \leq 5$), то можно выбрать n так, что станет возможно использовать все признаки Фурье порядка n . Это делает отбор признаков более-менее автоматической процедурой. Однако для пространств состояний высокой размерности необходимо отбирать подмножество этих признаков. Это можно сделать, имея априорные знания о характере аппроксимируемой функции, причем некоторые методы автоматизированного отбора можно адаптировать для работы в условиях инкрементной и нестационарной природы обучения с подкреплением. Базисные признаки Фурье в этом плане хороши тем, что легко отбирать признаки, задавая векторы $\mathbf{c}^i$, так чтобы они учитывали предполагаемые зависимости между переменными состояния, и ограничив значения элементов векторов $\mathbf{c}^i$, так чтобы аппроксимация могла отфильтровать высокочастотные компоненты, считающиеся шумом. С другой стороны, поскольку признаки Фурье отличны от

нуля во всем пространстве состояний (за исключением нескольких нулей), они представляют глобальные свойства состояний, а это осложняет отыскание хороших способов представления локальных свойств.

На рис. 9.5 показано сравнение кривых обучения для базиса Фурье и полиномиального базиса в задаче о случайном блуждании с 1000 состояний. В общем случае мы не рекомендуем использовать полиномы для онлайн-обучения¹.

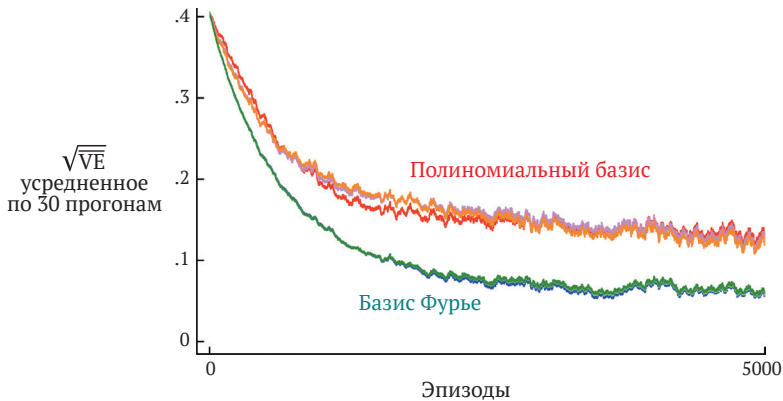


Рис. 9.5 ❖ Сравнение базиса Фурье и полиномиального базиса в задаче о случайном блуждании с 1000 состояний. Показаны кривые обучения для градиентного метода Монте-Карло с базисом Фурье и полиномиальным базисом порядка 5, 10 и 20. Размеры шага были грубо оптимизированы для каждого случая: $\alpha = 0.0001$ для полиномиального базиса и $\alpha = 0.00005$ для базиса Фурье. В качестве показателя качества (ось y) был взят корень из среднеквадратической ошибки ценности (9.1)

9.5.3. Грубое кодирование

Рассмотрим задачу, для которой естественным представлением пространства состояний является непрерывное двумерное пространство. Для этого случая возможное представление состоит из признаков, соответствующих окружностям в пространстве состояний, как показано на рис. 9.6. Если состояние находится внутри окружности, то соответствующий признак равен 1 и говорят, что он *присутствует*; в противном случае признак равен 0 и говорят, что он *отсутствует*. Такие признаки, принимающие только два значения, 1 и 0, называются *бинарными*. Для данного состояния присутствующие признаки показывают, внутри каких окружностей это состояние находится, а значит, грубо кодируют его местоположение. Представление состояния признаками, перекрывающимися таким образом (они не обязаны быть ни окружностями, ни бинарными), называется *грубым кодированием*.

¹ Существуют семейства полиномов более сложные, чем обсуждавшиеся здесь, например различные семейства ортогональных полиномов, которые могут давать лучшие результаты, но в настоящее время опыт их применения в обучении с подкреплением невелик.

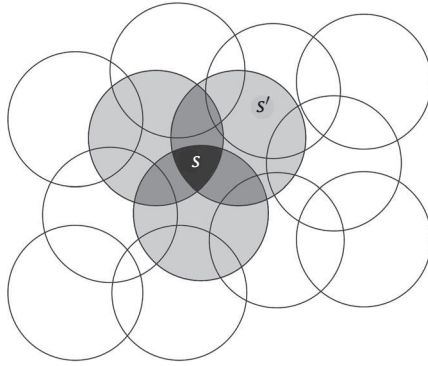


Рис. 9.6 ❖ Грубое кодирование. Обобщение с состояния s на состояние s' зависит от количества их признаков (в данном случае – окружностей) с перекрывающимися рецептивными полями. У этих состояний только один общий признак, поэтому не следует ожидать хорошего обобщения

В предположении линейной аппроксимации методом градиентного спуска рассмотрим влияние размера и плотности расположения окружностей на результат. Каждой окружности соответствует один вес (элемент вектора w), который принимает значение в результате обучения. Обучение, произведенное на одном состоянии (точке в пространстве), повлияет на веса всех окружностей, охватывающих это состояние. Таким образом, в силу (9.8) на приближенную функцию ценности оказывают влияние все состояния внутри объединения окружностей, и влияние тем больше, чем больше количество окружностей, для которых эта точка «общая», как на рис. 9.6. Если окружности малы, то обобщение действует на коротком расстоянии, как на рис. 9.7 (слева), а если велики, то на большом расстоянии, как на рис. 9.7 (в центре). Кроме того, на характер обобщения влияет форма признаков. Например, если бы они были не строго круговыми, а вытянутыми в одном направлении, то и область обобщения изменилась бы аналогично – как на рис. 9.7 (справа).

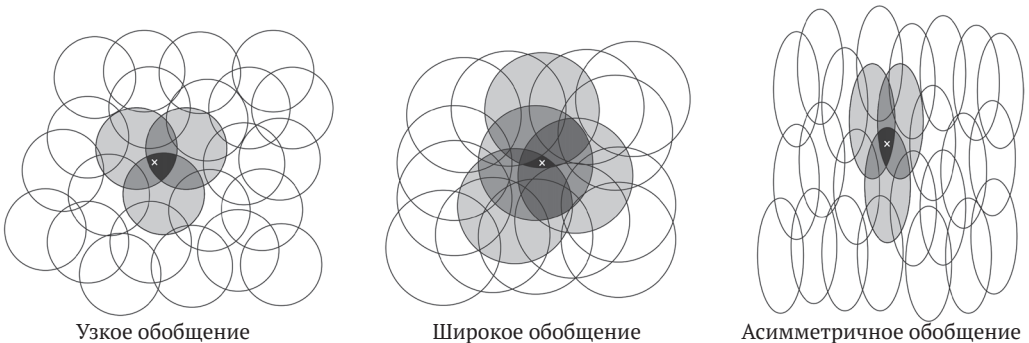


Рис. 9.7 ❖ Обобщение в методах линейной аппроксимации функций зависит от размеров и формы рецептивных полей признаков. Во всех трех случаях количество и плотность признаков примерно одинаковы

Признаки с большим рецептивным полем дают широкое обобщение, но может показаться, что они заодно ограничивают обученную функцию, вынуждая ее быть грубой аппроксимацией, разрешающая способность которой не может быть значительно мельче, чем ширина рецептивного поля. К счастью, это не так. Начальное обобщение с одной точки на другую действительно управляется размером и формой рецептивных полей, но резкость – максимально возможная в конечном итоге разрешающая способность – управляется в большей степени общим числом признаков.

Пример 9.3. Грубость грубого кодирования. Этот пример иллюстрирует влияние размера рецептивных полей при грубом кодировании на обучение. Линейная аппроксимация, основанная на грубом кодировании, и формула (9.7) использовались для обучения одномерной прямоугольной волны (показана на рис. 9.8 сверху). Значения этой функции применялись в качестве целей, U_t . При наличии всего одного измерения рецептивные поля были интервалами, а не окружностями. Обучение было повторено с тремя разными размерами интервалов: узкий, средний и широкий, – как показано в нижней части рисунка. Во всех трех случаях плотность признаков была одинакова, приблизительно 50 на область, в которой обучалась функция. Обучающие примеры случайно генерировались с равномерным распределением в этой области. Размер шага был равен $\alpha = 0.2/n$, где n – количество признаков, присутствовавших в какой-то момент времени. На рис. 9.8 показаны функции, обученные во всех трех случаях, на разных этапах обучения. Заметим, что ширина признаков оказывала большое влияние на ранних стадиях обучения. Для широких признаков и обобщение было широким; для узких признаков изменялись только значения в малой окрестности каждого обучающего примера, поэтому функция оказывалась более «бугристой». Однако окончательная обученная функция лишь в малой степени зависела от ширины признаков. Рецептивные поля действительно оказывают сильное влияние на обобщение, но слабое – на асимптотическое качество решения.

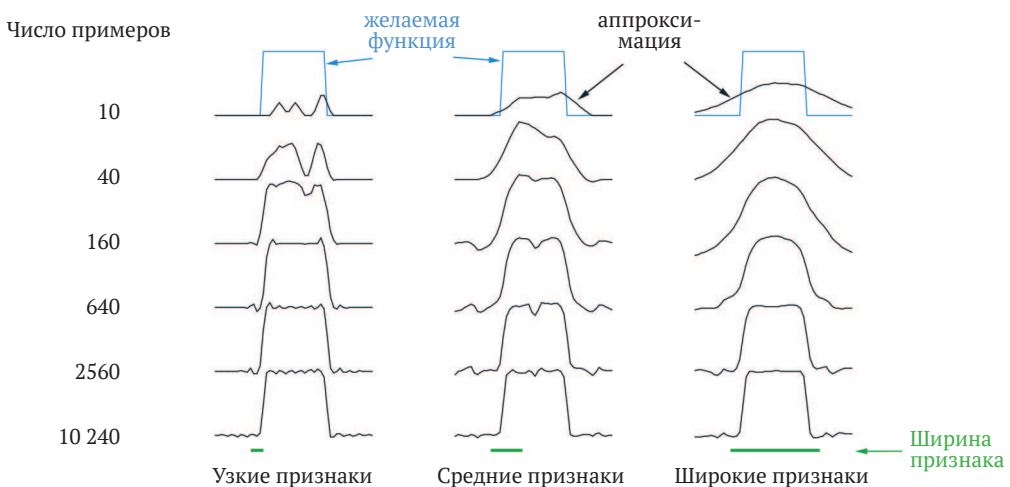


Рис. 9.8 ❖ Пример сильного влияния ширины признаков на начальное обобщение (первая строка) и слабого влияния на асимптотическую точность (последняя строка)

9.5.4. Плиточное кодирование

Плиточное кодирование – форма грубого кодирования для многомерного непрерывного пространства, одновременно гибкая и вычислительно эффективная. Быть может, это самое практичное представление признаков для современных последовательных цифровых компьютеров.

В случае плиточного кодирования рецептивные поля признаков сгруппированы в разбиения пространства состояний. Каждое такое разбиение называется *замощением* (tiling), а каждый элемент замощения – *плиткой* (tile). Например, простейшее замощение двумерного пространства состояний – равномерная сетка, показанная в левой части рис. 9.9. Здесь плитки, или рецептивные поля, – это квадраты, а не окружности, как на рис. 9.6. Если бы использовалось только одно это замощение, то состояние, обозначенное белой точкой, представлялось бы единственным признаком, в плитку которого это состояние попадает; обобщение было бы полным для всех состояний в той же плитке и вообще отсутствовало бы для состояний вне нее. При наличии только одного замощения мы имели бы не грубое кодирование, а просто агрегирование состояний.

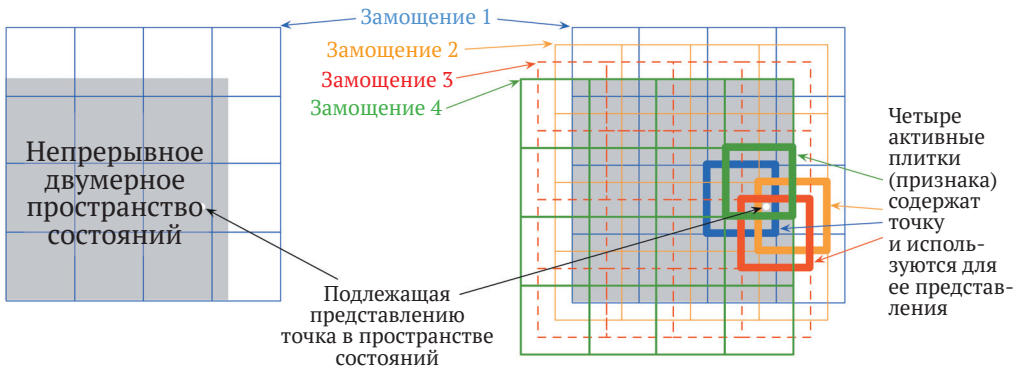


Рис. 9.9 ❖ Несколько перекрывающихся сеточных замощений ограниченной области двумерного пространства. Замощения сдвинуты друг относительно друга на одну и ту же величину по каждой оси

Чтобы проявились сильные стороны грубого кодирования, рецептивные поля должны перекрываться, а, по определению, плитки разбиения не перекрываются. Для получения настоящего грубого кодирования с помощью плиточного кодирования используется несколько замощений, сдвинутых друг относительно друга на величину, меньшую ширины плитки. В правой части рис. 9.9 показан простой случай с четырьмя замощениями. Каждое состояние, в т. ч. обозначенное белой точкой, попадает ровно в одну плитку каждого из четырех замощений. Эти четыре плитки соответствуют четырем признакам, которые становятся активны, когда возникает данное состояние. Точнее, вектор признаков $x(s)$ содержит по одному элементу для каждой плитки из каждого замощения. В этом примере элементов $4 \times 4 \times 4 = 64$, и все они равны 0, кроме четырех, соответствующих плиткам, в которые попало состояние s . На рис. 9.10 показано преимущество нескольких сдвинутых замощений (грубого кодирования) над единственным замощением в задаче о случайном блуждании с 1000 состояний.

Очевидное практическое преимущество плиточного кодирования состоит в том, что поскольку оно работает с разбиениями пространства состояний, общее число признаков, активных одновременно, одинаково в любом состоянии. В каждом замощении присутствует ровно один признак, поэтому общее число присутствующих признаков всегда такое же, как число замощений. Это позволяет задавать размер шага α простым и интуитивно понятным способом. Например, если взять $\alpha = 1/n$, где n – число замощений, то получится обучение ровно с одним испытанием. Если обучение производится на примере $s \mapsto v$, то при любой априорной оценке $\hat{v}(s, \mathbf{w}_t)$ новая оценка будет равна $\hat{v}(s, \mathbf{w}_{t+1}) = v$. Обычно желательнее более медленное изменение, допускающее обобщение и стохастическую вариацию целей (выходов). Например, можно взять $\alpha = 1/10n$, тогда оценка для обученного состояния сдвинулась бы на одну десятую расстояния до цели при одном обновлении, а соседние состояния сдвигались бы меньше, пропорционально количеству общих плиток.

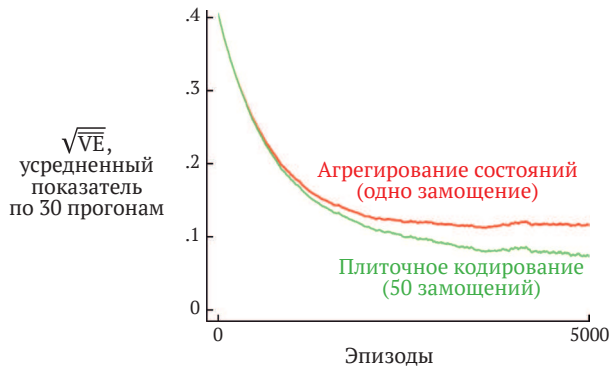


Рис. 9.10 ❖ Почему мы применяем грубое кодирование. Показаны кривые обучения для задачи о случайном блуждании с 1000 состояний, к которой применялся градиентный алгоритм Монте-Карло с одним и с несколькими замощениями. Пространство, содержащее 1000 состояний, рассматривалось как одна непрерывная ось, покрытая плитками шириной 200 состояний каждая. Несколько замощений были сдвинуты друг относительно друга на 4 состояния. Размер шаг выбран так, чтобы начальная скорость обучения в обоих случаях была одинакова: $\alpha = 0.0001$ для одного замощения и $\alpha = 0.0001/50$ для 50 замощений

Плиточное кодирование дает вычислительный выигрыш еще и вследствие использования бинарных векторов признаков. Поскольку каждый элемент вектора равен 0 или 1, вычисление взвешенной суммы, аппроксимирующей функцию ценности (9.8), почти тривиально. Вместо того чтобы выполнять d умножений и сложений, нужно лишь вычислить индексы $n \ll d$ активных признаков, а затем сложить n соответствующих им элементов вектора весов.

Обобщение возможно на те состояния, которые попадают в одну из тех же плиток, что и примеры, предъявленные в процессе обучения, и пропорционально количеству общих плиток. Даже выбор того, на сколько сдвигать замощения друг относительно друга, влияет на обобщение. Если они сдвигаются равномерно, как на рис. 9.9, то различные состояния могут обобщаться качественно по-разному, как показано в верхней части рис. 9.11. На каждом из восьми изображений мы

видим разные паттерны обобщения обучающего примера на близлежащие точки. В этом примере замощений восемь, а значит, внутри одной плитки имеется 64 подобласти, которые обобщаются по-разному, но в соответствии с одним из этих восьми паттернов. Обратите внимание, что во многих паттернах равномерный сдвиг приводит к выраженному эффекту вдоль диагонали. Этих артефактов можно избежать, если сдвигать замощения асимметрично, как показано в нижней части рисунка. Нижние паттерны обобщения лучше, поскольку обучающий пример всегда находится в центре, а явная асимметрия отсутствует.

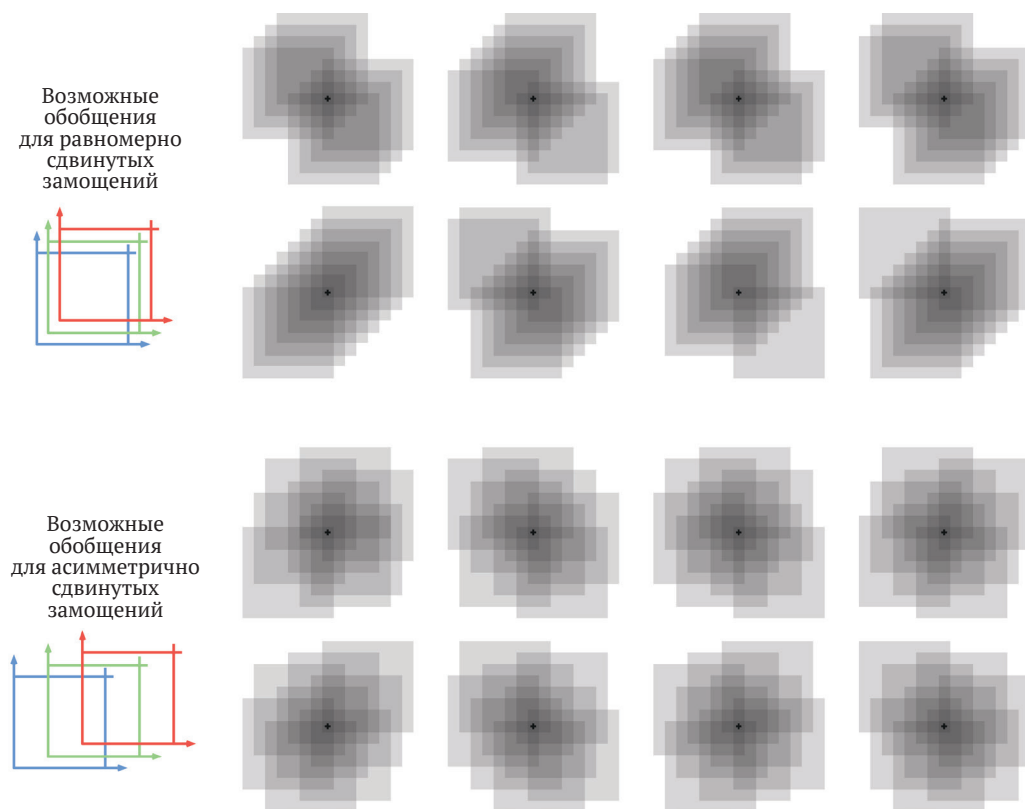


Рис. 9.11 ❖ Почему асимметричный сдвиг предпочтителен при плиточном кодировании. Показана сила обобщения обучающего состояния-примера, обозначенного черным знаком $+$, на близлежащие состояния для случая восьми замощений. Если замощения сдвинуты равномерно (сверху), то имеют место диагональные артефакты и заметные вариации обобщения, а если сдвиг асимметричный, то форма обобщения более однородна и похожа на сферу

Во всех случаях замощения сдвинуты друг относительно друга на часть ширины плитки по каждой оси. Если обозначить w ширину плитки, а n – число замощений, то w/n будет основной единицей. Все состояния в небольших w/n квадратах на каждой стороне активируют одни и те же плитки, имеют одинаковое представление с помощью признаков и одинаковую приближенную ценность. Если со-

стояние сдвинуть на w/n в направлении любой декартовой оси, то представление с помощью признаков изменится на одну компоненту (плитку). В случае равномерного сдвига все замощения сдвинуты друг относительно друга в точности на эту основную единицу расстояния. В двумерном пространстве мы говорим, что каждое замощение сдвинуто на вектор смещения $(1, 1)$; это означает, что оно сдвинуто относительно предыдущего замощения на этот вектор, умноженный на w/n . В этих терминах асимметрично сдвинутые замощения, показанные в нижней части рис. 9.11, сдвинуты друг относительно друга на вектор смещения $(1, 3)$.

Влиянию векторов смещения на обобщение плиточного кодирования посвящено много работ (Parks and Militzer, 1991; An, 1991; An, Miller and Parks, 1991; Miller, An, Glanz and Carter, 1990), в которых оценивается однородность и склонность к диагональным артефактам типа тех, что мы видели для вектора смещения $(1, 1)$. На основе проделанной работы Миллер и Гланц (Miller and Glanz, 1996) рекомендуют использовать векторы смещения, состоящие из первых нечетных чисел. В частности, для непрерывного k -мерного пространства хорошим выбором будет вектор смещения $(1, 3, 5, 7, \dots, 2k - 1)$ и n (количество замощений), равное степени 2, большей или равной $4k$. Именно так мы и поступили при создании замощений, показанных в нижней части рис. 9.11, где $k = 2$, $n = 2^3 \geq 4k$, а вектор смещения равен $(1, 3)$. В трехмерном случае первые четыре замощения были бы сдвинуты от начального положения на $(0, 0, 0)$, $(1, 3, 5)$, $(2, 6, 10)$ и $(3, 9, 15)$. В сети нетрудно найти программы с открытым исходным кодом, которые эффективно строят подобные замощения для любого k .

При выборе стратегии замощения следует принять решение о количестве замощений и форме плиток. Количество замощений вместе с размером плиток определяет разрешающую способность, или мелкость асимптотической аппроксимации, так же как в случае грубого кодирования (см. рис. 9.8). Форма плиток определяет природу обобщения, как показано на рис. 9.7. Квадратные плитки дают примерно одинаковое обобщение по каждой оси, как на рис. 9.11 (внизу). Плитки, вытянутые в одном направлении, например полосы на рис. 9.12 (в центре), способствуют обобщению в этом направлении. Кроме того, замощения на рис. 9.12 (в центре) плотнее и тоньше слева, что улучшает различение при малых значениях вдоль горизонтальной оси. Замощение диагональными полосами на рис. 9.12 (справа) способствует обобщению вдоль одной диагонали. При большем числе измерений полосы в направлении одной оси соответствуют игнорированию некоторых измерений в некоторых замощениях, а именно в гиперплоских слоях. Нерегулярные замощения типа того, что показано на рис. 9.12 (слева), также возможны, хотя на практике встречаются редко и требуют нестандартного программного обеспечения.

На практике часто желательно использовать плитки разной формы в разных замощениях. Например, можно использовать замощения горизонтальными и вертикальными полосами. Это будет способствовать обобщению вдоль обеих осей. Но одно лишь замощение полосами не позволяет обучиться тому факту, что конкретному сочетанию горизонтальных и вертикальных координат соответствует специальное значение ценности (какое бы значение ни было обучено, оно просчитается в состоянии с такими же горизонтальными или вертикальными координатами). Для этой цели необходимы связующие прямоугольные плитки типа тех, что первоначально были показаны на рис. 9.9. С помощью нескольких замоще-

ний – горизонтальных, вертикальных и связующих – можно получить всё разом: предпочтение обобщению вдоль осей и одновременно возможность обучиться специальным значениям при определенных сочетаниях координат (примеры см. в работе Sutton, 1996). Выбор замощений определяет характер обобщения, и до тех пор, пока мы не научимся эффективно автоматизировать этот выбор, важно, чтобы плиточное кодирование позволяло делать его гибко и так, чтобы было понятно человеку.

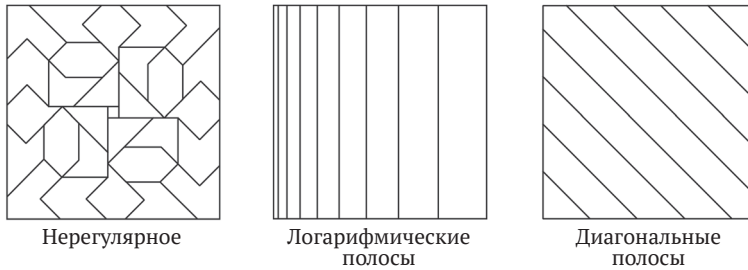
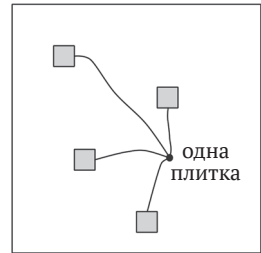


Рис. 9.12 ❖ Замощения необязательно должны быть сетками. Плитки могут иметь произвольную форму, а замощение может быть неравномерным, но при этом вычисления во многих случаях остаются эффективными

Еще один прием, полезный для уменьшения требований к памяти, – *хеширование* – псевдослучайное сворачивание большого замощения в гораздо меньшее множество плиток. Хеширование порождает плитки, состоящие из несмежных случайно разбросанных по пространству состояний областей, которые тем не менее образуют исчерпывающее разбиение. Например, одна плитка может состоять из четырех подплиток, как показано на рисунке справа. Благодаря хешированию потребление памяти часто удается снизить в несколько раз почти без потери качества. Это возможно, потому что высокая разрешающая способность нужна лишь в малой части пространства состояний. Хеширование избавляет нас от проклятия размерности – потребление памяти необязательно должно экспоненциально расти с увеличением количества измерений, оно должно лишь отвечать реальным потребностям задачи. Реализации плиточного кодирования с открытым исходным кодом обычно включают эффективное хеширование.



Упражнение 9.4. Допустим, у нас есть основания полагать, что одно из двух измерений пространства состояний оказывает большее влияние на функцию ценности, чем другое, и что обобщение должно производиться главным образом поперек этого измерения, а не вдоль него. Какого рода замощения можно было бы предложить, чтобы извлечь пользу из этого априорного знания? □

9.5.5. Радиально-базисные функции

Радиально-базисные функции (РБФ) представляют собой естественное обобщение грубого кодирования на непрерывные признаки, которые принимают не два значения, 0 и 1, а, например, любое значение в интервале $[0, 1]$. Это значение описывает *степень* присутствия признака. Типичный РБФ-признак x_i имеет нормальное (колоколообразное) распределение $x_i(s)$, зависящее только от расстояния между состоянием s и каноническим, или центральным, состоянием признака c_i , и приведенное к ширине признака σ_i :

$$x_i(s) \doteq \exp\left(-\frac{\|s - c_i\|^2}{2\sigma_i^2}\right).$$

Норму, или метрику, конечно, можно выбрать любым способом, наиболее подходящим к рассматриваемым состояниям или задаче. На рисунке ниже показан пример в одномерном случае с евклидовой метрикой:

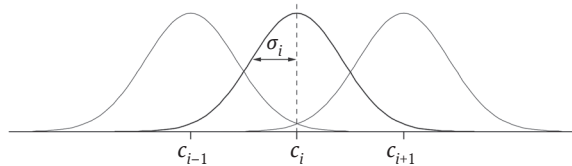


Рис. 9.13 ❖ Одномерные радиально-базисные функции

Главное преимущество РБФ над бинарными признаками заключается в том, что порождаемые ими приближенные функции гладкие и дифференцируемые. Это заманчиво, но в большинстве случаев не имеет практического значения. Тем не менее градуированным функциям отклика, в частности РБФ, в контексте плиточного кодирования посвящены обширные исследования (An, 1991; Miller et al., 1991; An et al., 1991; Lane, Handelman and Gelfand, 1992). Все эти методы вычислительно существенно сложнее плиточного кодирования и зачастую дают худшие результаты, когда размерность пространства состояний больше двух. В многомерных пространствах гораздо важнее поведение на краях плиток, и доказано, что получить хорошо контролируемую градуированную активацию вблизи краев трудно.

РБФ-сеть называется линейный аппроксиматор функций, в котором признаками являются радиально-базисные функции. Обучение определяется уравнениями (9.7) и (9.8) точно так же, как для других линейных аппроксиматоров. Дополнительно в некоторых методах обучения РБФ-сетей изменяются также центр и ширина признаков, что делает их нелинейными. Нелинейные методы позволяют аппроксимировать целевую функцию гораздо точнее. Недостаток РБФ-сетей, особенно нелинейных, – повышенная вычислительная сложность, к тому же для обеспечения надежности и эффективности часто требуется ручная настройка.

9.6. ВЫБОР РАЗМЕРА ШАГА ВРУЧНУЮ

В большинстве СГС-методов проектировщик должен выбрать подходящий размер шага α . В идеале выбор хотелось бы автоматизировать, и в некоторых случаях так оно и есть, но чаще всего приходится задавать его вручную. Чтобы это сделать и лучше понять алгоритмы, полезно развить интуитивные представления о роли размера шага. Можем ли мы сказать, как нужно его задавать в общем случае?

К сожалению, теоретические соображения мало чем помогают. Теория стохастической аппроксимации дает условия (2.7) на медленно убывающую последовательность размеров шага, достаточные для гарантии сходимости, но обычно при этом обучение оказывается слишком медленным. Классический выбор $\alpha_t = 1/t$, порождающий выборочные средние в табличных методах МК, не подходит для TD-методов, для нестационарных задач и для всех методов, в которых используется аппроксимация функций. Для линейных методов имеются рекурсивные методы наименьших квадратов, которые выбирают оптимальный *матричный* размер шага, и их можно обобщить на обучение на основе временных различий, как в методе LSTD, описанном в разделе 9.8, но при этом требуется $O(d^2)$ параметров размера шага, или в d раз больше параметров, чем мы обучаем. По этой причине мы даже не рассматриваем их для решения больших задач, в которых в основном аппроксимация функции и нужна.

Чтобы получить некоторое интуитивное представление о том, как задавать размер шага вручную, давайте ненадолго вернемся к табличному случаю. В этом случае мы понимаем, что размер шага $\alpha = 1$ приводит к полному устранению ошибки выборки после одного эксперимента (см. формулу (2.4) с размером шага 1). Как обсуждалось на стр. 242, обычно мы заинтересованы в меньшей скорости обучения. В табличном случае при размере шага $\alpha = 1/10$ понадобилось бы примерно 10 экспериментов для сходимости приблизительно к их средней цели, а если бы мы захотели обучиться на 100 экспериментах, то взяли бы $\alpha = 1/100$. Вообще, если $\alpha = 1/t$, то табличная оценка состояния после t экспериментов с этим состоянием будет близка к среднему целей, причем наибольшее влияние будут оказывать последние цели.

В общей теории аппроксимации нет такого очевидного понятия количества экспериментов с состоянием, поскольку каждое состояние может быть в разной степени похоже и непохоже на все остальные. Однако имеется общее правило, которое дает сходное поведение в случае линейной аппроксимации функций. Предположим, что мы хотим обучиться приблизительно за t экспериментов практически с одним и тем же вектором признаков. Тогда для задания размера шага в линейных СГС-методах можно следовать такому эвристическому правилу:

$$\alpha \doteq (t\mathbb{E}[\mathbf{x}^T\mathbf{x}])^{-1}, \quad (9.19)$$

где $\mathbf{x}$ – случайный вектор признаков, выбранный из того же распределения, что и входные векторы в СГС. Этот метод работает лучше всего, если длина векторов признаков слабо изменяется, в идеале $\mathbf{x}^T\mathbf{x}$ должно быть постоянно.

Упражнение 9.5. Предположим, что плиточное кодирование используется для преобразования семимерного непрерывного пространства состояний в бинарные векторы признаков с целью оценивания функции ценности состояния $\hat{v}_\pi(\mathbf{s}, \mathbf{w}) \approx v_\pi(\mathbf{s})$.

Есть основания полагать, что зависимость между измерениями слабая, поэтому мы решаем использовать восемь замощений по каждому измерению в отдельности (полосами), т. е. всего $7 \times 8 = 56$ замощений. Кроме того, на случай, если между измерениями имеются какие-то попарные зависимости, мы берем все $\binom{7}{2} = 21$ пар измерений и для каждой пары строим по два замощения прямоугольными плитками. Всего получается $21 \times 2 + 56 = 98$ замощений. Имея эти векторы признаков, мы все же подозреваем, что следует применить усреднение для устранения шума, поэтому решаем, что обучение должно быть постепенным, т. е. один и тот же вектор признаков должен предъявляться примерно 10 раз, прежде чем обучение выйдет на асимптоту. Как следует выбрать размер шага α ? Объясните свой ответ. □

9.7. НЕЛИНЕЙНАЯ АППРОКСИМАЦИЯ ФУНКЦИЙ: ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

Искусственные нейронные сети (ИНС) широко применяются для нелинейной аппроксимации функций. ИНС – это сеть взаимосвязанных блоков, обладающих некоторыми свойствами нейронов, основных компонентов нервной системы. ИНС имеют долгую историю, но последние достижения связаны с обучением глубоких ИНС с большим числом слоев (глубокое обучение). Именно они лежат в основе самых впечатляющих возможностей систем машинного обучения, в т. ч. и обучения с подкреплением. В главе 16 мы опишем несколько поражающих воображение примеров систем обучения с подкреплением, в которых ИНС применяются для аппроксимации функций.

На рис. 9.14 общая ИНС с прямым распространением. Это означает, что в сети нет циклов, т. е. таких путей, с помощью которых выход блока мог бы повлиять на его вход. В этой сети имеется выходной слой, состоящий из двух блоков, входной слой с четырьмя блоками и два «скрытых слоя», не являющихся ни входными, ни выходными. Каждой связи сопоставлен вещественный вес. Грубо говоря, вес – это аналог силы синаптической связи в настоящей нейронной сети (см. раздел 15.1). Если в графе связей АНС имеется хотя бы один цикл, то она называется рекуррентной, а не сетью прямого распространения. Хотя в обучении с подкреплением применяются как сети прямого распространения, так и рекуррентные, в этой книге мы будем рассматривать только первые, поскольку они проще.

Блоки (обозначены окружностями на рис. 9.14) обычно являются полулинейными, т. е. вычисляют взвешенную сумму входных сигналов, а затем применяют к результату нелинейную *функцию активации*, которая порождает выходной сигнал блока, или его активацию. Применяются различные функции активации, но обычно S-образные, или сигмоидные, например логистическая функция $f(x) = 1/(1 + e^{-x})$, хотя иногда в качестве нелинейности используется ректификатор $f(x) = \max(0, x)$. Ступенчатая функция $f(x) = 1$, если $x \geq \theta$, и 0 в противном случае, дает бинарный блок с порогом θ . Блоки входного слоя сети несколько отличаются, потому что их активации – полученные извне значения, являющиеся входными данными для функции, которую аппроксимирует сеть.

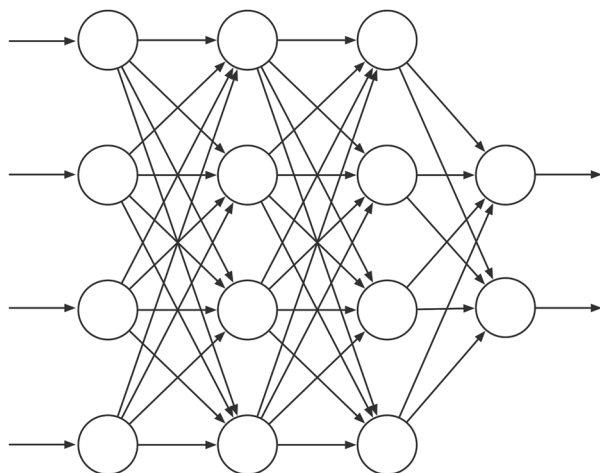


Рис. 9.14 ❖ ИНС прямого распространения
с четырьмя входными блоками, двумя выходными блоками
и двумя скрытыми слоями

Активация каждого выходного блока ИНС прямого распространения – нелинейная функция паттернов активации входных блоков сети. Функции параметризуются весами связей сети. ИНС без скрытых слоев способна представить только малую часть возможных функций, отображающих вход в выход. Однако ИНС с одним скрытым слоем, содержащим достаточное число сигмоидных блоков, может аппроксимировать любую непрерывную функцию на компактной области пространства входов сети с любой точностью (Cybenko, 1989). Это справедливо также для любой нелинейной функции активации, удовлетворяющей не слишком сильным условиям, но ее нелинейность существенна: если все блоки многослойной сети прямого распространения имеют линейные функции активации, то вся сеть эквивалентна сети без скрытых слоев (поскольку суперпозиция линейных функций сама является линейной функцией).

Несмотря на свойство «универсальной аппроксимации», присущее ИНС с одним скрытым слоем, теоретически и экспериментально показано, что для аппроксимации сложных функций, возникающих во многих задачах искусственного интеллекта, проще – а иногда обязательно – использовать абстракции, представляющие собой иерархические композиции большого количества слоев, соответствующих абстракциям низкого уровня; иными словами, абстракции, порождаемые глубокими архитектурами, каковыми являются ИНС со многими скрытыми слоями (см. подробный обзор Bengio, 2009). Последовательные слои глубокой ИНС вычисляют все более абстрактные представления «исходных» данных сети, в которых каждый блок дает признак, привносящий вклад в иерархическое представление всей функции сети, отображающей вход в выход.

Поэтому обучение скрытых слоев ИНС – способ автоматического создания признаков, подходящих для данной задачи, который позволяет порождать иерархические представления, не полагаясь исключительно на признаки, сконструированные вручную. Эта проблема стояла перед искусственным интеллектом давно, поэтому понятно, почему алгоритмы обучения ИНС со скрытыми слоями при-

влекли так много внимания в последние годы. Обычно ИНС обучают методом стохастического градиентного спуска (раздел 9.3). Каждый вес корректируется в направлении, которое сулит улучшение общего качества сети, измеряемого целевой функцией, подлежащей минимизации или максимизации. В наиболее распространенном случае обучения с учителем в роли целевой функции выступает математическое ожидание ошибки, или потери на множестве помеченных обучающих примеров. В обучении с подкреплением для обучения функций ценности можно использовать ИНС с TD-ошибкой в качестве целевой функции. Можно также поставить в качестве цели максимизацию ожидаемого дохода, как в градиентной задаче о бандите (раздел 2.9) или в алгоритме градиента стратегии (глава 13). Во всех этих случаях необходимо оценить, как вес каждой связи влияет на общее качество сети, иначе говоря, оценить частную производную целевой функции по каждому весу, зная текущие значения всех весов сети. Градиент – это вектор таких частных производных.

Самый успешный способ решить эту задачу для ИНС со скрытыми слоями (при условии что функции активации блоков дифференцируемы) дает алгоритм обратного распространения, который включает чередующиеся прямые и обратные проходы по сети. На каждом прямом проходе вычисляется активация каждого блока, зная текущие активации входных блоков сети. Затем на обратном проходе производится эффективное вычисление частных производных по каждому весу. (Как и в других стохастических градиентных алгоритмах обучения, вектор этих частных производных является оценкой истинного градиента.) В разделе 15.10 мы обсудим методы обучения ИНС со скрытыми слоями, в которых вместо обратного распространения используются принципы обучения с подкреплением. Эти методы не столь эффективны, как алгоритм обратного распространения, но могут быть ближе к тому, как обучаются настоящие нейронные сети.

Алгоритм обратного распространения может давать хорошие результаты для плоских сетей с 1 или 2 скрытыми слоями, но для более глубоких сетей не всегда работает хорошо. На самом деле обучение сети с $k + 1$ скрытыми слоями может дать результаты хуже, чем при обучении сети с k скрытыми слоями, пусть даже более глубокая сеть способна представить все функции, представимые более мелкой сетью (Bengio, 2009). Объяснить подобные факты нелегко, но важно несколько факторов. Во-первых, из-за большого количества весов в типичной глубокой ИНС трудно избежать проблемы переобучения, когда сеть не обобщается на примеры, которые не предъявлялись в процессе обучения. Во-вторых, обратное распространение плохо работает для глубоких ИНС, потому что частные производные, вычисляемые на обратных проходах, быстро убывают при приближении к входному слою, из-за чего обучение глубоких слоев происходит очень медленно. А бывает и так, что частные производные быстро возрастают при приближении к входному слою, и тогда обучение становится неустойчивым. Удачные способы решения этих проблем и позволили получить самые поразительные результаты, достигнутые системами на основе ИНС в последнее время.

Переобучение – проблема для любого метода аппроксимации функций, который корректирует функции с большим количеством степеней свободы, имея ограниченный объем обучающих данных. Проблема не так серьезна для онлайн-обучения с подкреплением, которое не опирается на ограниченные обучающие наборы, но все равно эффективное обобщение остается важным вопросом. Пере-

обучение составляет проблему для любых ИНС, но особенно для глубоких, потому что количество весов в них очень велико. Для борьбы с переобучением разработано много методов, например: остановка обучения, когда качество на контрольных данных, отличных от обучающих, начинает падать (перекрестный контроль); модификация целевой функции для уменьшения сложности аппроксимации (регуляризация) и введение зависимостей между весами для уменьшения числа степеней свободы (например, разделение весов).

Особенно эффективным методом борьбы с переобучением в глубоких ИНС является прореживание, предложенное в работе Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov (2014). В процессе обучения блоки случайным образом удаляются из сети (сеть прореживается) вместе со своими связями. Это можно интерпретировать как обучение большого числа «истонченных» сетей. Комбинирование результатов, полученных на этих истонченных сетях, на этапе тестирования позволяет улучшить качество обобщения. Метод прореживания, по сути, эффективно аппроксимирует эту комбинацию, умножая вес каждой выходной связи блока на вероятность сохранения этого блока в процессе обучения. Сривастава с соавторами обнаружили, что этот метод значительно повышает качество обобщения. Он понуждает отдельные скрытые блоки обучаться признакам, которые хорошо работают со случайными наборами других признаков. Это повышает универсальность признаков, найденных скрытыми блоками, избегая чрезмерной специализации сети на редко встречающихся случаях.

В работе Hinton, Osindero, and Teh (2006) предпринят важный шаг в направлении решения проблемы обучения глубоких слоев глубокой ИНС. Изученные ими сети глубокого доверия тесно связаны с обсуждаемыми здесь глубокими ИНС. В предложенном ими методе самые глубокие слои обучались по одному с применением алгоритма обучения без учителя. Поскольку обучение без учителя не опирается на общую целевую функцию, удастся выделить признаки, улавливающие статистические регулярности потока входных данных. Самый глубокий слой обучается первым, затем на данных, предоставляемых этим слоем, обучается следующий по глубине и т. д. В итоге веса во всех или во многих слоях сети становятся начальными значениями для обучения с учителем. После этого сеть подвергается тонкой настройке с помощью алгоритма обратного распространения – теперь уже с общей целевой функцией. Исследования показывают, что этот подход в общем случае работает гораздо лучше, чем обратное распространение со случайно инициализированными весами. Повышение качества сетей, при обучении которых веса были инициализированы описанным способом, можно объяснять по-разному, в частности есть идея, что этот метод помещает сеть в область пространства весов, из которой градиентный алгоритм может быстро сойтись к оптимуму.

Пакетная нормировка (Ioffe and Szegedy, 2015) – еще одна техника, упрощающая обучение глубоких сетей. Давно уже было известно, что обучение ИНС проходит проще, если входные данные нормированы, например так, что каждая входная величина имеет нулевое среднее и единичную дисперсию. Пакетная нормировка при обучении глубоких ИНС нормирует выход глубоких слоев перед подачей на вход следующего слоя. В работе Ioffe and Szegedy (2015) использовалась статистика по подмножествам, или «мини-пакетам», обучающих примеров с целью нормировать эти межслойные сигналы для повышения скорости обучения глубокой ИНС.

Еще одна техника, полезная для обучения глубоких ИНС, – *глубокое остаточное обучение* (residual learning) (He, Zhang, Ren, and Sun, 2016). Иногда проще обучиться отличиям функции от тождественной, чем самой функции. Тогда прибавление этой разности, или остаточной функции, к входу дает желаемую функцию. В глубоких ИНС можно обучить группу слоев остаточной функции, просто добавив прямые связи в обход этой группы. Эти связи добавляют вход группы к ее выходу, так что никаких дополнительных весов не требуется. В работе He et al. (2016) этот метод оценивался для глубоких сверточных сетей с прямыми связями в обход каждой пары соседних слоев, и был обнаружен значительный выигрыш по сравнению с сетями без прямых связей на эталонных задачах классификации изображений. И пакетная нормировка, и глубокое остаточное обучение нашли применение в приложении обучения с подкреплением к игре го, описанном в главе 16.

В различных приложениях, включая впечатляющие приложения обучения с подкреплением, большой успех сопутствовал *глубоким сверточным сетям*. Этот тип сетей предназначен специально для обработки данных высокой размерности, организованных в виде пространственных массивов, например изображений. В его основу легли знания о ранних этапах обработки зрительной информации мозгом (LeCun, Bottou, Bengio and Haffner, 1998). Благодаря специальной архитектуре глубокую сверточную сеть можно обучить методом обратного распространения, не прибегая к описанным выше методам обучения глубоких слоев.

На рис. 9.15 показана архитектура глубокой сверточной сети. Конкретно эта сеть, заимствованная из работы LeCun et al. (1998), проектировалась для распознавания рукописных символов. Она состоит из чередующихся сверточных слоев и слоев подвыборки, за которыми следует несколько полносвязных слоев. Каждый сверточный слой порождает ряд карт признаков. Карта признаков – это паттерн активности в массиве блоков, где каждый блок выполняет одну и ту же операцию над данными в своем рецептивном поле – «видимой» ему части данных предыдущего слоя (или внешних входных данных в случае первого сверточного слоя). Блоки в карте признаков совпадают во всем, кроме рецептивных полей, все они имеют одинаковые размер и форму, но расположены в разных позициях массива входных данных. Блоки в одной и той же карте признаков разделяют общие веса. Это означает, что карта признаков распознает данный признак вне зависимости от его положения во входном массиве. Например, в сети на рис. 9.15 первый сверточный слой порождает 6 карт признаков, каждая из которых содержит 28×28 блоков. Каждый блок в каждой карте признаков обладает рецептивным полем 5×5 , и эти поля перекрываются (в данном случае по четырем столбцам и четырем строкам). Следовательно, каждая из 6 карт признаков определяется всего 25 настраиваемыми весами.

Слои подвыборки глубокой сверточной сети уменьшают пространственное разрешение карт признаков. Каждая карта признаков в слое подвыборки состоит из блоков, которые производят усреднение по рецептивному полю блоков в картах признаков предыдущего сверточного уровня. Например, каждый блок в каждой из 6 карт признаков в первом слое подвыборки сети на рис. 9.15 производит усреднение по неперекрывающемуся рецептивному полю 2×2 одной из карт признаков, порожденных первым сверточным слоем, что дает шесть карт признаков размера 14×14 . Слои подвыборки уменьшают чувствительности сети к положению обнаруженных признаков в пространстве, т. е. способствуют пространствен-

ной инвариантности откликов сети. Это полезно, потому что признак, найденный в одном месте изображения, скорее всего, окажется полезен и в других местах.

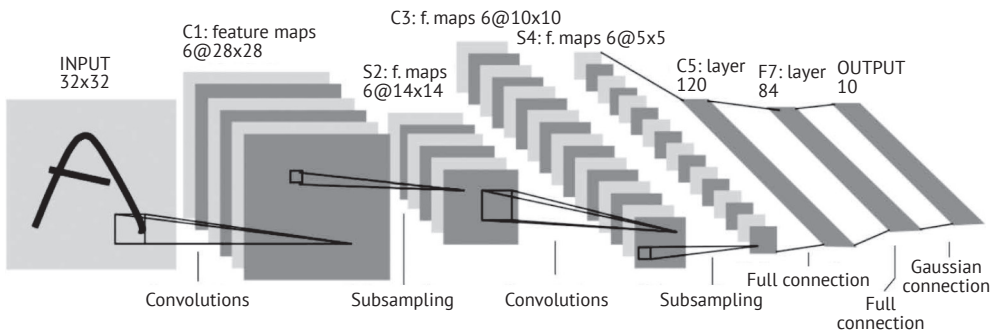


Рис. 9.15 ❖ Глубокая сверточная сеть. Воспроизведено с разрешения журнала «Proceedings of the IEEE» из статьи LeCun, Bottou, Bengio, and Haffner «Gradient-based learning applied to document recognition», том 86, 1998. Разрешение получено при посредничестве компании Copyright Clearance Center, Inc.

Достижения проектирования и обучения ИНС – а мы упомянули лишь малую их часть – имеют применение и в обучении с подкреплением. Хотя современная теория обучения с подкреплением ограничена в основном методами, в которых применяются табличные алгоритмы или линейная аппроксимация функций, самые известные приложения обучения с подкреплением своими впечатляющими успехами во многом обязаны нелинейной аппроксимации с помощью многослойных ИНС. Некоторые из этих приложений рассматриваются в главе 16.

9.8. АЛГОРИТМ TD НАИМЕНЬШИХ КВАДРАТОВ

Все рассмотренные в этой главе методы требовали на каждом шаге объема вычислений, пропорционального количеству параметров. Но если увеличить объем вычислений, то можно достигнуть большего. В этом разделе мы представим метод линейной аппроксимации функций, дающий, пожалуй, лучшее, чего можно достичь в этом случае.

Как было установлено в разделе 9.4, алгоритм TD(0) с линейной аппроксимацией асимптотически сходится (при надлежащем уменьшении размера шага) к неподвижной точке TD:

$$\mathbf{w}_{\text{TD}} = \mathbf{A}^{-1}\mathbf{b},$$

где

$$\mathbf{A} \doteq \mathbb{E}[\mathbf{x}_t(\mathbf{x}_t - \mathbf{x}_{t+1})^T] \quad \text{и} \quad \mathbf{b} \doteq \mathbb{E}[R_{t+1}\mathbf{x}_t].$$

Может возникнуть вопрос: почему мы должны вычислять это решение итеративно? Это же безумно расточительное использование данных! Не лучше ли будет вычислить оценки $\mathbf{A}$ и $\mathbf{b}$, а затем непосредственно вычислить неподвижную точ-

ку TD? Алгоритм TD наименьших квадратов, больше известный под названием LSTD, именно это и делает. Он вычисляет естественные оценки

$$\hat{\mathbf{A}}_t \doteq \sum_{k=0}^{t-1} \mathbf{x}_k (\mathbf{x}_k - \gamma \mathbf{x}_{k+1})^\top + \varepsilon \mathbf{I} \quad \text{и} \quad \hat{\mathbf{b}}_t \doteq \sum_{k=0}^{t-1} R_{k+1} \mathbf{x}_k, \quad (9.20)$$

где $\mathbf{I}$ – единичная матрица, а $\varepsilon \mathbf{I}$ для небольшого $\varepsilon > 0$ гарантирует, что матрица $\hat{\mathbf{A}}_t$ всегда обратима. Может показаться, что обе эти оценки следовало бы разделить на t , и это действительно так; в том виде, в котором мы их определили, они на самом деле оценивают $\mathbf{A}$, умноженное на t , и $\mathbf{b}$, умноженное на t . Однако дополнительные множители t сокращаются, когда LSTD используется для оценки неподвижной точки TD следующим образом:

$$\mathbf{w}_t \doteq \hat{\mathbf{A}}_t^{-1} \hat{\mathbf{b}}_t. \quad (9.21)$$

Этот алгоритм – форма линейного TD(0), в которой данные используются наиболее эффективно, но при этом он и самый дорогостоящий с точки зрения вычислений. Напомним, что в полугradientном TD(0) объем памяти и вычислений на каждом шаге составляет всего $O(d)$.

Насколько сложен алгоритм LSTD? В том виде, в каком мы его представили, сложность, похоже, возрастает с увеличением t , но обе аппроксимации в (9.20) можно было бы реализовать инкрементно, применяя рассмотренные выше (например, в главе 2) приемы, так что на каждом шаге будет затрачено постоянное время. Но все равно обновление $\hat{\mathbf{A}}_t$ включало бы вычисление внешнего произведения (вектора-столбца на вектор-строку) и, стало быть, являлось бы обновлением матрицы; его вычислительная сложность составляла бы $O(d^2)$, и, конечно, для хранения матрицы $\hat{\mathbf{A}}_t$ понадобилась бы память объемом $O(d^2)$.

Еще более серьезной потенциальной проблемой было бы то, что в окончательном вычислении по формуле (9.21) используется матрица, обратная к $\hat{\mathbf{A}}_t$, а вычислительная сложность обращения матрицы общего вида составляет $O(d^3)$. К счастью, нам нужно обратить матрицу специального вида – сумму внешних произведений, – а это можно сделать инкрементно со сложностью $O(d^2)$ следующим образом:

$$\begin{aligned} \hat{\mathbf{A}}_t^{-1} &= (\hat{\mathbf{A}}_{t-1} + \mathbf{x}_t (\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\top)^{-1} && \text{(в силу 9.20)} \\ &= \hat{\mathbf{A}}_{t-1}^{-1} - \frac{\hat{\mathbf{A}}_{t-1}^{-1} \mathbf{x}_t (\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\top \hat{\mathbf{A}}_{t-1}^{-1}}{1 + (\mathbf{x}_t - \gamma \mathbf{x}_{t+1})^\top \hat{\mathbf{A}}_{t-1}^{-1} \mathbf{x}_t} \end{aligned} \quad (9.22)$$

для $t > 0$, где $\hat{\mathbf{A}}_0 \doteq \varepsilon \mathbf{I}$. Хотя тождество (9.22), известное как *формула Шермана–Моррисона*, кажется очень сложным, на самом деле оно включает только умножение вектора на матрицу и вектора на вектор, а значит, имеет сложность всего $O(d^2)$. Таким образом, мы можем сохранить обратную матрицу $\hat{\mathbf{A}}_t^{-1}$, пересчитывать ее по формуле (9.22) и затем использовать в (9.21) – и на все это потребуется объем памяти и вычислений порядка $O(d^2)$ на каждом шаге. Полный псевдокод алгоритма приведен во врезке ниже.

Разумеется, $O(d^2)$ – это все еще гораздо дороже, чем полугradientный TD, имеющий сложность $O(d)$. Стоит ли более эффективное использование данных в алго-

ритме LSTD таких вычислительных расходов, зависит от того, насколько велико d , насколько важна скорость обучения, а также от затрат в других частях системы. Еще иногда обращают внимание на тот факт, что LSTD вообще не нуждается в размере шага, но это преимущество, пожалуй, переоценено. LSTD действительно не требует задавать размер шага, зато в нем есть ε ; если ε выбрано слишком малым, то последовательность обратных матриц может сильно флуктуировать, а если слишком большим, то обучение замедлится. Кроме того, отсутствие размера шага в алгоритме LSTD означает, что он ничего не забывает. Иногда это свойство желательно, но становится проблематичным, если целевая стратегия π изменяется, как то бывает в обучении с подкреплением и ОИС. В приложениях к управлению LSTD обычно приходится сочетать с каким-то другим механизмом, чтобы ввести забывание, что ставит под вопрос преимущество, заключающееся в отсутствии размера шага.

Алгоритм LSTD для оценивания $\hat{v} = w^T x(\cdot) \approx v_\pi$ (вариант со сложностью $O(d^2)$)

Вход: представление признаков $x: \mathcal{S}^+ \rightarrow \mathbb{R}^d$ такое, что $x(\text{terminal}) = 0$
 Параметр алгоритма: небольшое $\varepsilon > 0$

$$\widehat{A}^{-1} \leftarrow \varepsilon^{-1} I$$

$$\widehat{b} \leftarrow 0$$

Повторять для каждого эпизода:

Инициализировать S ; $x \leftarrow x(S)$

Повторять для каждого шага эпизода:

Выбрать и предпринять действие $A \sim \pi(\cdot | S)$, наблюдать R, S' ; $x' \leftarrow x(S')$

$$v \leftarrow \widehat{A}^{-1T}(x - \gamma x')$$

$$\widehat{A}^{-1} \leftarrow \widehat{A}^{-1} - (\widehat{A}^{-1} x) v^T / (1 + v^T x)$$

$$\widehat{b} \leftarrow \widehat{b} + R x$$

$$w \leftarrow \widehat{A}^{-1} \widehat{b}$$

$$S \leftarrow S'; x \leftarrow x'$$

пока S' не является заключительным

9.9. Аппроксимация функций с запоминанием

До сих пор мы обсуждали параметрический подход к аппроксимации функций ценности. При таком подходе алгоритм обучения корректирует параметры функциональной формы, предназначенной для аппроксимации функции ценности на всем пространстве состояний задачи. Каждое обновление $s \mapsto g$ является обучающим примером, который алгоритм обучения использует для изменения параметров с целью уменьшить ошибку аппроксимации. После обновления обучающий пример можно отбросить (но можно и сохранить для дальнейшего использования). Если понадобится приближенная ценность состояния (мы называем это *опросом состояния*), то функция просто вычисляется в этом состоянии с использованием последних значений параметров, вычисленных алгоритмом обучения.

Методы аппроксимации функций с запоминанием (memory-based function approximation) устроены совершенно иначе. Они просто сохраняют обучающие примеры (или хотя бы подмножество всех примеров) в памяти по мере поступления, не обновляя никаких параметров. Затем, когда возникает необходимость в оценке ценности состояния, множество примеров извлекается из памяти и используется для вычисления этой оценки. Иногда такой подход называют *ленивым обучением*, поскольку обработка обучающих примеров откладывается до того момента, когда систему попросят выдать результат.

Методы аппроксимации функций с запоминанием служат основными примерами непараметрических методов. В отличие от параметрических методов, форма аппроксимирующей функции не обязана принадлежать какому-то фиксированному параметрическому классу функций, скажем линейных или полиномиальных, но определяется самими обучающими примерами, а также средствами их комбинирования для получения оценок ценности опрашиваемых состояний. От параметрических методов ожидают, что чем больше обучающих примеров запомнено в памяти, тем более точной будет аппроксимация произвольной целевой функции.

Существует много разных методов с запоминанием, различающихся способами отбора запоминаемых обучающих примеров и их использования для ответа на запрос. Здесь мы ограничимся методами локального обучения, которые аппроксимируют функцию ценности только в окрестности текущего опрашиваемого состояния. Эти методы извлекают из памяти множество обучающих примеров, состояния которых признаны наиболее релевантными опрашиваемому состоянию, причем релевантность обычно определяется расстоянием между состояниями: чем ближе состояние обучающего примера к опрашиваемому, тем более релевантным оно считается. Само же понятие расстояния можно определять по-разному. После того как ценность опрашиваемого состояния вычислена, локальная аппроксимация отбрасывается.

Самый простой пример подхода с запоминанием – метод ближайшего соседа, который находит в памяти пример, состояние которого ближе всего к опрашиваемому, и возвращает ценность этого примера в качестве приближенной ценности опрашиваемого состояния. Иными словами, если опрашивается состояние s и $s' \mapsto g$ – хранящийся в памяти пример, для которого s' ближе всего к s , то в качестве приближенной ценности s возвращается g . Чуть сложнее *средневзвешенные методы*, которые извлекают множество ближайших соседних примеров и возвращают взвешенное среднее их целевых значений, причем вес состояния обычно убывает с увеличением расстояния между ним и опрашиваемым состоянием. *Локально взвешенная регрессия* аналогична, но подгоняет поверхность к ценностям множества ближайших состояний с помощью какого-нибудь метода параметрической аппроксимации, который минимизирует взвешенную ошибку типа (9.1), а веса зависят от расстояния до опрашиваемого состояния. Возвращенная ценность – результат вычисления локально прилегающей поверхности в опрашиваемом состоянии, после возврата эта поверхность отбрасывается.

Будучи непараметрическими, методы с запоминанием имеют над параметрическими то преимущество, что не ограничены заранее определенными формами функций. Поэтому точность может увеличиваться по мере накопления данных. У методов локальной аппроксимации с запоминанием есть и другие свойства,

благодаря которым они хорошо подходят для обучения с подкреплением. Поскольку в обучении с подкреплением огромное значение имеет траекторная выборка (см. раздел 8.6), локальные методы с запоминанием могут сфокусировать аппроксимацию функций на локальных окрестностях состояний (или пар состояние–действие), посещенных на реальных или имитированных траекториях. Глобальная аппроксимация вообще не всегда нужна, потому что многие области пространства состояний никогда (или почти никогда) не посещаются. Кроме того, в методах с запоминанием опыт агента может оказывать чуть ли не мгновенное влияние на оценки ценности в окрестности текущего состояния – в отличие от параметрических методов, которые должны постепенно корректировать параметры глобальной аппроксимации.

Отказ от глобальной аппроксимации – это также способ справиться с проклятием размерности. Например, для k -мерного пространства состояний табличный метод с хранением глобальной аппроксимации требует памяти, экспоненциально зависящей от k . С другой стороны, при хранении примеров в методе с запоминанием каждый пример требует памяти, пропорциональной k , а объем памяти, необходимый для хранения n примеров, линейно зависит от n . Здесь нет экспоненциальной зависимости ни от k , ни от n . Конечно, остается критически важный вопрос: может ли метод с запоминанием давать ответ достаточно быстро, чтобы быть полезным агенту? С этим связан и другой вопрос: как убывает скорость работы с ростом занятой памяти? Нахождение ближайших соседей в большой базе данных может занимать слишком много времени, так что во многих приложениях оказывается практически неприменимым.

Сторонники методов с запоминанием разработали способы ускорить поиск ближайших соседей. Один из подходов – использовать параллельные компьютеры или специализированное оборудование, другой – специальные многомерные структуры данных для хранения обучающих примеров. Одна из таких структур – k - d дерево (сокращение « k -мерного дерева»), которое рекурсивно разбивает k -мерное пространство на области, организованные как узлы двоичного дерева. В зависимости от количества данных и их распределения по пространству состояний поиск ближайшего соседа с помощью k - d деревьев способен быстро исключать из рассмотрения большие области пространства и в некоторых задачах дает приемлемые результаты там, где наивный поиск занял бы слишком много времени.

Локально взвешенная регрессия дополнительно нуждается в способах быстро вычисления локальной регрессии, поскольку эти вычисления повторяются для ответа на каждый запрос. Было разработано много способов решения данной проблемы, в т. ч. методы забывания элементов в определенном порядке, чтобы размер базы данных не превышал заданные границы. В библиографических и исторических замечаниях к этой главе имеются ссылки на литературу по этой теме, в т. ч. на подборку работ, в которых описываются приложения обучения с запоминанием к обучению с подкреплением.

9.10. Аппроксимация с помощью ядерных функций

Методы с запоминанием, такие как описанные выше методы взвешенного среднего и локально взвешенной регрессии, опираются на назначение весов приме-

рам $s' \mapsto g$ в базе данных, которые зависят от расстояния между s' и опрашиваемым состоянием s . Функция, назначающая эти веса, называется *ядерной функцией*, или просто *ядром*. Например, в методах взвешенного среднего и локально взвешенной регрессии ядерная функция $k: \mathbb{R} \rightarrow \mathbb{R}$ назначает веса расстояниям между состояниями. Вообще говоря, веса необязательно зависят от расстояний, они могут зависеть от какой-то другой меры сходства между состояниями. В таком случае $k: \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$, т. е. $k(s, s')$ – вес, приданный данным о том, как s' влияет на ответы на запросы о s .

Если взглянуть под другим углом зрения, то $k(s, s')$ измеряет силу обобщения s' на s . Ядерные функции численно выражают степень релевантности знаний об одном состоянии любому другому состоянию. Например, силе обобщения для плиточного кодирования, показанного на рис. 9.11, соответствуют различные ядерные функции, получающиеся из равномерного и асимметричного сдвига замощений. Хотя в плиточном кодировании ядерная функция явно не используется, обобщение производится согласно такой функции. На самом деле, как станет ясно из обсуждения ниже, силу обобщения при линейной параметрической аппроксимации функций всегда можно описать ядерной функцией.

Ядерная регрессия – это метод с запоминанием, который вычисляет ядерное взвешенное среднее целей всех хранящихся в памяти примеров и сопоставляет результат опрашиваемому состоянию. Если $\mathcal{D}$ – множество хранимых примеров, а $g(s')$ – цель для состояния s' в хранимом примере, то ядерная регрессия вычисляет аппроксимацию целевой функции, в данном случае функции ценности, зависящую от $\mathcal{D}$:

$$\hat{v}(s, \mathcal{D}) = \sum_{s' \in \mathcal{D}} k(s, s') g(s'). \quad (9.23)$$

Описанный выше метод взвешенного среднего – это частный случай, в котором $k(s, s')$ отлично от нуля тогда и только тогда, когда s и s' близки друг к другу, так что сумму необязательно вычислять по всему $\mathcal{D}$.

Часто в роли ядра выступает гауссова радиально-базисная функция (РБФ), используемая в РБФ-аппроксимации, описанной в разделе 9.5.5. Там РБФ были признаками, центр и ширина которых либо фиксированы с самого начала, так что центры предположительно сконцентрированы в областях, где ожидается много примеров, либо каким-то образом корректируются в процессе обучения. Если оставить в стороне методы с корректировкой центров и ширин, то это линейный параметрический метод, в котором параметрами являются веса каждой РБФ, обычно обучаемые методом стохастического градиентного или полуградиентного спуска. Аппроксимирующая функция имеет вид линейной комбинации предопределенных РБФ. Ядерная регрессия с РБФ-ядром отличается в двух отношениях. Во-первых, это метод с запоминанием: центры РБФ совпадают с состояниями хранимых примеров. Во-вторых, это непараметрический метод: никакие параметры не обучаются, а ответ на запрос вычисляется по формуле (9.23).

Разумеется, для практической реализации ядерной регрессии нужно разрешить немало вопросов, которые выходят за рамки нашего краткого обсуждения. Однако оказывается, что любой метод линейной параметрической регрессии типа описанных в разделе 9.4, в котором состояния представляются векторами признаков $\mathbf{x}(s) = (x_1(s), x_2(s), \dots, x_d(s))^T$, можно переформулировать как ядерную

регрессию, где $k(s, s')$ – скалярное произведение представлений векторов признаков s и s' , т. е.

$$k(s, s') = \mathbf{x}(s)^T \mathbf{x}(s'). \quad (9.24)$$

Ядерная регрессия с этим ядром дает такую же аппроксимацию, какую дал бы линейный параметрический метод, если бы он использовался с этими векторами признаков и обучался на тех же данных.

Мы опустим математическое обоснование этого утверждения, поскольку его можно найти в любом современном учебнике по машинному обучению, например в книге Bishop (2006), а просто отметим важное следствие. Вместо конструирования признаков для линейного параметрического аппроксиматора можно строить ядерные функции непосредственно, вообще не ссылаясь на векторы признаков. Не все ядерные функции можно выразить в виде скалярных произведений векторов признаков, как в формуле (9.24), но ядерная функция, которая может быть записана в таком виде, дает значительные преимущества над эквивалентным параметрическим методом. Для многих множеств векторов признаков у равенства (9.24) имеется компактная функциональная форма, которую можно вычислить, не производя вычислений в d -мерном пространстве признаков. В таких случаях ядерная регрессия гораздо проще прямого использования линейного параметрического метода с состояниями, представленными этими векторами признаков. Этот так называемый «ядерный трюк» позволяет эффективно организовать работу только со множеством хранимых обучающих примеров, когда размерность полного пространства признаков очень высока. Ядерный трюк лежит в основе многих методов машинного обучения, и показано, что иногда он может быть с пользой применен и к обучению с подкреплением.

9.11. БОЛЕЕ ГЛУБОКИЙ ВЗГЛЯД НА ОБУЧЕНИЕ С ЕДИНОЙ СТРАТЕГИЕЙ: ЗАИНТЕРЕСОВАННОСТЬ И ЗНАЧИМОСТЬ

В рассмотренных в этой главе алгоритмах все состояния трактовались единообразно, как если бы они были одинаково важны. Но иногда одни состояния интересуют нас больше, чем другие. Например, в эпизодических задачах с обесцениванием нас может больше интересовать точная оценка ценности ранних состояний в эпизоде, чем более поздних состояний, поскольку из-за обесценивания доход в последних гораздо менее важен для ценности стартового состояния. Другой пример – при обучении функции ценности действий не так важна точная оценка плохих действий, ценность которых гораздо меньше, чем у жадного действия. Ресурсы, доступные для аппроксимации функций, всегда ограничены, и если использовать их более целенаправленно, то, возможно, удастся добиться лучших результатов.

Одна из причин, почему мы рассматриваем все состояния одинаково, заключается в том, что впоследствии мы производим обновление согласно распределению с единой стратегией, для которого имеются более сильные теоретические результаты, относящиеся к полугradientным методам. Напомним, что распределение с единой стратегией было определено как распределение состояний, встре-

чающихся в МППР при следовании целевой стратегии. Теперь мы значительно обобщим эту концепцию. Вместо одного распределения с единой стратегией для МППР мы будем рассматривать много распределений. У всех них есть общая черта: это распределения состояний, встречающихся на траекториях, которые возникают при следовании целевой стратегии, а различаются они тем, как были инициализированы эти траектории.

Мы введем несколько новых понятий. Сначала определим неотрицательную скалярную меру – случайную величину I_t , называемую *заинтересованностью*, которая показывает, в какой мере нам интересна точная ценность состояния (или пары состояние–действие) в момент t . Если состояние нам вообще безразлично, то заинтересованность должна быть равна нулю; если же оно нам очень интересно, то заинтересованность может быть равна единице, хотя формально разрешено любое неотрицательное значение. Заинтересованность можно задавать любым причинно-обусловленным способом; например, она может зависеть от траектории до момента t или от обученных параметров в момент t . Распределение μ в выражении $\overline{VE}$ (9.1) тогда определяется как распределение состояний, встречающихся при следовании целевой стратегии, с весами, соответствующими заинтересованности. Кроме того, мы введем еще одну неотрицательную скалярную случайную величину M_t , называемую *значимостью* (emphasis). На этот скаляр умножается величина обновления в процедуре обучения, поэтому он увеличивает или уменьшает значимость обучения, произведенного в момент t . Тогда общее правило n -шагового обучения, заменяющее (9.15), принимает вид:

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha M_t [G_{t:t+n} - \hat{v}(S_t, \mathbf{w}_{t+n-1})] \nabla \hat{v}(S_t, \mathbf{w}_{t+n-1}), \quad 0 \leq t < T, \quad (9.25)$$

где n -шаговый доход описывается формулой (9.16), а значимость рекурсивно определяется по заинтересованности следующим образом:

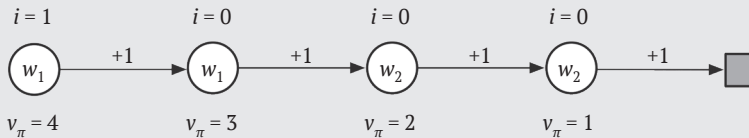
$$M_t = I_t + \gamma^n M_{t-n}, \quad 0 \leq t < T \quad (9.26)$$

и $M_t \doteq 0$ для всех $t < 0$. Эти уравнения включают в качестве частного случая метод Монте-Карло, для которого $G_{t:t+n} = G_t$, все обновления производятся в конце эпизода, $n = T - t$ и $M_t = I_t$.

Пример 9.5 иллюстрирует, как заинтересованность и значимость могут дать более точные оценки ценности.

Пример 9.5. Заинтересованность и значимость

Для демонстрации потенциальной выгоды, которую могут дать заинтересованность и значимость, рассмотрим марковский процесс вознаграждения с четырьмя состояниями, показанный ниже.



Эпизоды начинаются в самом левом состоянии, затем происходят переходы на одно состояние вправо с вознаграждением на каждом шаге, пока не будет достигнуто за-

ключительное состояние. Таким образом, истинная ценность первого состояния равна 4, второго – 3 и т. д. – ценность каждого состояния написана под ним. Это истинные ценности; оценки ценностей являются лишь приближением к ним, поскольку ограничены параметризацией. Вектор параметров $\mathbf{w} = (w_1, w_2)^T$ содержит два элемента, параметризация написана внутри каждого состояния. Оценки ценностей первых двух состояний определяются только элементом w_1 и потому должны быть одинаковы, пусть даже истинные их ценности различны. Аналогично оценки ценностей третьего и четвертого состояний определяются только элементом w_2 и, стало быть, должны быть одинаковы, несмотря на различие истинных ценностей. Предположим, что нас интересует точная ценность только самого левого состояния. Назначим ему заинтересованность 1, а всем остальным состояниям – заинтересованность 0; эти значения написаны над состояниями.

Сначала рассмотрим применение градиентных алгоритмов Монте-Карло к этой задаче. Алгоритмы, представленные выше в этой главе, не учитывающие ни заинтересованности, ни значимости (см. (9.7) и врезку на стр. 243), сходятся (при убывающем размере шага) к вектору параметров $\mathbf{w}_\infty = (3.5, 1.5)$, который назначает первому состоянию – единственному, в котором мы заинтересованы, – ценность 3.5 (расположенную между истинными ценностями первого и второго состояний). С другой стороны, описанные в этом разделе методы, учитывающие заинтересованность и значимость, обучаются точной ценности первого состояния; w_1 будет сходиться к 4, тогда как w_2 вообще никогда не обновляется, потому что значимость равна нулю во всех состояниях, кроме самого левого.

Теперь рассмотрим применение двухшаговых полуградиентных TD-методов. Методы, описанные ранее в этой главе, не учитывающие ни заинтересованности, ни значимости ((9.15), (9.16) и врезка на стр. 251), снова сходятся к $\mathbf{w}_\infty = (3.5, 1.5)$, тогда как методы, учитывающие заинтересованность и значимость, сходятся к $\mathbf{w}_\infty = (4, 2)$. Второй вектор дает точные значения ценности первого и третьего состояний (из которого первое получается бутстрэппингом), но второе и четвертое вообще не обновляются.

9.12. РЕЗЮМЕ

Система обучения с подкреплением должна быть способна к *обобщению*, если мы намереваемся применить ее к искусственному интеллекту или в большом техническом приложении. Для этого можно использовать широкий спектр существующих методов аппроксимации функций на основе *обучения с учителем* – достаточно просто рассматривать каждое обновление как обучающий пример.

Из методов обучения с учителем, пожалуй, лучше всего подходят те, в которых используется *параметрическая аппроксимация функций*, когда стратегия параметризована вектором весов $\mathbf{w}$. Хотя вектор весов насчитывает много элементов, пространство состояний все равно гораздо больше, поэтому приходится соглашаться на приближенное решение. Мы определили *среднеквадратическую ошибку ценности* $\overline{VE}(\mathbf{w})$ как меру ошибки ценностей $v_{\pi\mathbf{w}}(s)$ для вектора весов $\mathbf{w}$ при *распределении с единой стратегией* μ . $\overline{VE}$ дает понятный способ ранжировать различные аппроксимации функции ценности в случае единой стратегии.

Для нахождения хорошего вектора весов чаще всего применяются различные варианты *стохастического градиентного спуска* (СГС). В этой главе мы акцентировали внимание на случае *единой фиксированной стратегии*, который называется также оцениванием, или предсказанием, стратегии; в этом случае естественным

алгоритмом обучения является *n*-шаговый полугradientный TD-алгоритм, который включает gradientный алгоритм Монте-Карло и полугradientный алгоритм TD(0) в качестве частных случаев (когда $n = \infty$ и $n = 1$ соответственно). Полугradientные TD-методы не являются настоящими gradientными методами. В таких бутстрэппинговых методах (включая ДП) вектор весов входит в цель обновления, но не принимается во внимание при вычислении gradientа – потому-то они и называются полугradientными. А раз так, то к ним неприменимы классические результаты, относящиеся к СГС.

Тем не менее полугradientные методы позволяют получить хорошие результаты в частном случае *линейной* аппроксимации функций, когда в роли оценок ценностей выступают суммы произведений признаков на веса. Линейный случай лучше других изучен теоретически и хорошо работает на практике, если правильно выбрать признаки. Выбор признаков – один из самых важных способов добавления априорных знаний о предметной области в систему обучения с подкреплением. Можно выбирать полиномиальные признаки, но они плохо обобщаются в случае онлайн-обучения, который и представляет основной интерес в обучении с подкреплением. Лучше выбирать признаки в соответствии с базисом Фурье или какой-то формой грубого кодирования с разреженными перекрывающимися рецептивными полями. Радиально-базисные функции полезны в одномерных и двумерных задачах, когда важен гладкий отклик. LSTD – линейный TD-метод предсказания, в котором данные используются наиболее эффективно, но объем вычислений пропорционален квадрату количества весов, тогда как сложность всех остальных методов зависит от количества весов линейно. К нелинейным методам относятся искусственные нейронные сети, обучаемые методом обратного распространения и различными вариантами СГС; эти методы стали чрезвычайно популярны в последние годы под названием *глубокого обучения с подкреплением*.

Линейный полугradientный *n*-шаговый TD-алгоритм для всех *n* гарантированно сходится при стандартных условиях к решению с $\sqrt{E}$, находящейся в некоторой окрестности оптимальной ошибки (которая асимптотически достигается методами Монте-Карло). Радиус этой окрестности тем меньше, чем больше *n*, и стремится к нулю, когда $n \rightarrow \infty$. Но на практике при очень больших *n* обучение происходит слишком медленно, так что обычно предпочтительнее использовать бутстрэппинг ($n < \infty$), что мы и видели при сравнении с табличными *n*-шаговыми методами из главы 7 и табличными TD-методами и методами Монте-Карло из главы 6.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

Обобщение и аппроксимация функций всегда были неотъемлемыми частями обучения с подкреплением. В работах Bertsekas and Tsitsiklis (1996), Bertsekas (2012), и Sugiyama et al. (2013) описано современное состояние аппроксимации функций в обучении с подкреплением. Некоторые ранние работы по применению аппроксимации в обучении с подкреплением обсуждаются в конце этого раздела.

9.3 Методы gradientного спуска для минимизации среднеквадратической ошибки при обучении с учителем хорошо известны. В работе Widrow and Hoff (1960) впервые описан алгоритм минимальной среднеквадратической

ошибки (LMS), который является прототипом всех инкрементных алгоритмов градиентного спуска. Детали этого и связанных с ним алгоритмов можно найти во многих учебниках (например, Widrow and Stearns, 1985; Bishop, 1995; Duda and Hart, 1973). Полуградиентный метод TD(0) впервые исследован в работах Sutton (1984, 1988) как часть линейного алгоритма TD(λ), который мы будем подробно рассматривать в главе 12. Термин «полуградиентный» применительно к описанию этих бутстрэппинговых методов введен в употребление во втором издании данной книги.

Самым ранним примером использования агрегирования состояний в обучении с подкреплением, вероятно, является система BOXES Мичи и Чемберса (Michie and Chambers, 1968). Теория агрегирования состояний в обучении с подкреплением разработана в трудах Singh, Jaakkola, and Jordan (1995) и Tsitsiklis and Van Roy (1996). В динамическом программировании агрегирование состояний использовалось с самых первых дней (см., например, Bellman, 1957a).

- 9.4** В работе Sutton (1988) доказана сходимость в среднем линейного TD(0) к решению с минимальной $\overline{VE}$ для случая, когда векторы признаков $\{\mathbf{x}(s): s \in \mathcal{S}\}$ линейно независимы. Сходимость с вероятностью 1 была доказана несколькими исследователями примерно в то же время (Peng, 1993; Dayan and Sejnowski, 1994; Tsitsiklis, 1994; Gurvits, Lin, and Hanson, 1994). Дополнительно в работе Jaakkola, Jordan, and Singh (1994) доказана сходимость при онлайн-обновлении. Во всех этих работах предполагается, что векторы признаков линейно независимы, а это означает, что количество элементов $\mathbf{w}_t$ не меньше количества состояний. Сходимость для более важного общего случая линейно зависимых векторов признаков была впервые доказана в работе Dayan (1992). Важное обобщение и усиление результата Дайана было получено в работе Tsitsiklis and Van Roy (1997). Авторы доказали главный результат этого раздела – границу асимптотической ошибки линейных бутстрэппинговых методов.
- 9.5** Наше изложение возможностей линейной аппроксимации функций основано на подходе, принятом в работе Barto (1990).
- 9.5.2** В работе Konidaris, Osentoski, and Thomas (2011) базис Фурье описан в простой форме, пригодной для задач обучения с подкреплением с многомерным непрерывным пространством состояний и необязательно периодическими функциями.
- 9.5.3** Термин *грубое кодирование* ввел в обиход Хинтон (Hinton, 1984), а наш рис. 9.6 основан на одном из рисунков из его работы. В работе Waltz and Fu (1965) приведен один из первых примеров применения этого вида аппроксимации функций в системе обучения с подкреплением.
- 9.5.4** Плиточное кодирование, включая хеширование, впервые описано в работах Albus (1971, 1981). Он описал его в терминах «контроллера артикулятора модели мозжечка», или CMAC – именно так плиточное кодирование иногда называют в литературе. Термин «плиточное кодирование» впервые появился в первом издании данной книги, хотя идея описания CMAC в этих терминах встречалась в работе Watkins (1989). Плиточное кодирование ис-

пользовалось во многих системах обучения с подкреплением (например, Shewchuk and Dean, 1990; Lin and Kim, 1991; Miller, Scalera, and Kim, 1994; Sofge and White, 1992; Tham, 1994; Sutton, 1996; Watkins, 1989), а также в других типах систем управления обучением (например, Kraft and Campagna, 1990; Kraft, Miller, and Dietz, 1992). Этот раздел в значительной степени опирается на работу Miller and Glanz (1996). Общее программное обеспечение для плиточного кодирования доступно на нескольких языках (см., например, <http://incompleteideas.net/tiles/tiles3.html>).

- 9.5.5** Аппроксимация радиально-базисными функциями привлекала к себе внимание с того момента, как в работе Broomhead and Lowe (1988) была установлена ее связь с ИНС. В обзоре Powell (1987) описаны ранние применения РБФ, а в трудах Poggio and Girosi (1989, 1990) этот подход был значительно развит и получил новые применения.
- 9.6** К автоматическим методам подбора размера шага относятся RMSprop (Tieleman and Hinton, 2012), Adam (Kingma and Ba, 2015), методы стохастического метаспуска типа Delta-Bar-Delta (Jacobs, 1988), его инкрементное обобщение (Sutton, 1992b, c; Mahmood et al., 2012) и нелинейные обобщения (Schraudolph, 1999, 2002). Специально для обучения с подкреплением проектировались, в частности, методы AlphaBound (Dabney and Barto, 2012), SID и NOSID (Dabney, 2014), TIDBD (Kearney et al., готовится к печати) и применение стохастического метаспуска к обучению методами градиента стратегии (Schraudolph, Yu, and Aberdeen, 2006).
- 9.7** Введение блока пороговой логики как абстрактной модели нейрона в работе McCulloch and Pitts (1943) стало началом ИНС. История ИНС как методов обучения для задач классификации или регрессии насчитывает несколько этапов: этап применения перцептрона (Rosenblatt, 1962) и метода ADALINE (ADaptive LINear Element) (Widrow and Hoff, 1960) для обучения однослойных ИНС, этап алгоритма обратного распространения ошибки (LeCun, 1985; Rumelhart, Hinton, and Williams, 1986) для обучения многослойных ИНС и современный этап глубокого обучения с упором на обучение с подкреплением (см., например, Bengio, Courville, and Vincent, 2012; Goodfellow, Bengio, and Courville, 2016). Существует много книг на тему ИНС, упомянем лишь Haykin (1994), Bishop (1995) и Ripley (2007).

Применение ИНС как средства аппроксимации функций в интересах обучения с подкреплением восходит к ранней работе Фарли и Кларка (Farley and Clark, 1954), которые использовали обучение с подкреплением для модификации весов кусочно-линейных функций, представляющих стратегии. В работе Widrow, Gupta, and Maitra (1973) представлен нейроноподобный линейный пороговый блок, реализующий процесс обучения, который авторы назвали обучением с критиком, или избирательной адаптацией с бутстрэппингом; это был вариант алгоритма ADALINE для обучения с подкреплением. В работах Werbos (1987, 1994) разработан подход к предсказанию и управлению, в котором ИНС, обученная методом обратного распространения ошибки, использовалась для обучения стратегий и функций ценности с помощью TD-подобных алгоритмов. В работах Barto, Sutton, and Brouwer (1981) и Barto and Sutton (1981b) развита идея сети с ассоциатив-

ной памятью (см., например, Kohonen, 1977; Anderson, Silverstein, Ritz, and Jones, 1977) применительно к обучению с подкреплением. В статье Barto, Anderson, and Sutton (1982) двухуровневая ИНС использовалась для обучения нелинейной политики управления и была подчеркнута роль первого слоя в обучении подходящего представления. Хэмпсон (Hampson, 1983, 1989) одним из первых пропагандировал применение многослойных ИНС для обучения функций ценности. В работе Barto, Sutton, and Anderson (1983) представлен алгоритм исполнитель–критик в форме обучения ИНС для балансирования моделируемого стержня (см. разделы 15.7 и 15.8). В работе Barto and Anandan (1985) описана стохастическая версия алгоритма избирательной адаптации с бутстрэппингом (Widrow et al., 1973), названная алгоритмом *ассоциативного вознаграждения и штрафа* (A_{R-P}). В работах Barto (1985, 1986) и Barto and Jordan (1987) описаны многослойные ИНС, состоящие из A_{R-P} -блоков, обученных с помощью глобально распространяемого сигнала подкрепления, для обучения линейно неразделимых правил классификации. В работе Barto (1985) обсуждался этот подход к ИНС и вопрос о том, как данный тип правил обучения соотносится с другими, описанными в тогдашней литературе. (Дополнительные сведения об этом подходе к обучению многослойных ИНС см. в разделе 15.10.) В работах Anderson (1986, 1987, 1989) были подвергнуты оценке многочисленные методы обучения многослойных ИНС и показано, что алгоритм исполнитель–критик, в котором и исполнитель, и критик реализованы двухслойными ИНС, обученными методом обратного распространения ошибки, превосходит однослойные ИНС в решении задач о балансировании стержня и о ханойской башне. В работе Williams (1988) описано несколько способов комбинирования обратного распространения и обучения с подкреплением для обучения ИНС. В работах Gullapalli (1990) и Williams (1992) предложены алгоритмы обучения с подкреплением для нейроноподобных блоков с непрерывными, а не бинарными выходами. В работе Barto, Sutton, and Watkins (1990) доказывалось, что ИНС могут играть важную роль в аппроксимации функций, необходимой при решении последовательных задач принятия решений. В работе Williams (1992) установлена связь между правилами обучения REINFORCE (раздел 13.3) и методом обратного распространения ошибки применительно к обучению многослойных ИНС. Программа игры в нарды Тезауро (Tesauro 1992, 1994; раздел 16.1) убедительно продемонстрировала возможности алгоритма $TD(\lambda)$, в котором для аппроксимации функций использовалась многослойная ИНС применительно к обучению игре в нарды. В программах AlphaGo, AlphaGo Zero и AlphaZero, описанных в работах Silver et al. (2016, 2017a, b; раздел 16.6), обучение с подкреплением в сочетании с глубокой сверточной ИНС позволило достичь впечатляющих результатов в игре го. Работа Schmidhuber (2015) представляет собой обзор применения ИНС, в т. ч. рекуррентных, к обучению с подкреплением.

- 9.8** Алгоритм LSTD принадлежит Браттке и Барто (см. Bradtke, 1993, 1994; Bradtke and Barto, 1996; Bradtke, Ydstie, and Barto, 1994), а дальнейшее развитие получил в работах Boyan (1999, 2002), Nedić and Bertsekas (2003) и Yu (2010). Инкрементное обновление обратной матрицы известно по меньшей мере с 1949 года (Sherman and Morrison, 1949). Обобщение методов наи-

меньших квадратов на управление предложено в работах Lagoudakis and Parr (2003; Buşoniu, Lazaric, Ghavamzadeh, Munos, Babuška, and De Schutter, 2012).

- 9.9** Наше обсуждение аппроксимации функций с запоминанием основано преимущественно на обзоре локально взвешенного обучения Atkeson, Moore, and Schaal (1997). В работе Atkeson (1992) обсуждается применение локально взвешенной регрессии для обучения робота с запоминанием, там же имеется обширная библиографическая справка по истории идеи. В работе Stanfill and Waltz (1986) приведены авторитетные аргументы в пользу важности методов с запоминанием в искусственном интеллекте, особенно в свете параллельных архитектур (например, Connection Machine), которые тогда начали получать распространение. В работе Baird and Klopff (1993) предложен новаторский подход на основе запоминания, который использовался в качестве метода аппроксимации функций для Q-обучения применительно к задаче балансирования стержня. В работе Schaal and Atkeson (1994) локально взвешенная регрессия применена к задаче жонглирования роботом, где использовалась для обучения модели системы. В работе Peng (1995) задача о балансировании стержня использована в экспериментах с несколькими методами ближайших соседей для аппроксимации функций ценности, стратегий и моделей окружающей среды. В работе Tadepalli and Ok (1996) получены многообещающие результаты в области применения локально взвешенной линейной регрессии к обучению функции ценности для имитационной модели беспилотного транспортного средства. В работе Bottou and Vapnik (1992) продемонстрирована поразительная эффективность нескольких алгоритмов локального обучения по сравнению с нелокальными алгоритмами в некоторых задачах распознавания образов; там же обсуждается влияние локального обучения на обобщение.

В работе Bentley (1975) введены k - d деревья и сообщено о поиске ближайшего соседа по n записям за время $O(\log n)$ в среднем. В работе Friedman, Bentley, and Finkel (1977) уточнен алгоритм поиска ближайшего соседа с помощью k - d деревьев. В работе Omohundro (1987) обсуждается, какой выигрыш в эффективности возможен в результате применения иерархических структур данных типа k - d деревьев. В работе Moore, Schneider, and Deng (1997) предложено применение k - d деревьев для эффективного выполнения локально взвешенной регрессии.

- 9.10** В основе ядерной регрессии лежит *метод потенциальных функций*, описанный в работе Aizerman, Braverman, and Rozonoer (1964). Они рассматривали данные как распределенные в пространстве точечные электрические заряды разного знака и величины. Результирующий пространственный электрический потенциал, который получается суммированием потенциалов точечных зарядов, соответствовал интерполированной поверхности. При такой аналогии ядерная функция – это потенциал точечного заряда, который убывает обратно пропорционально расстоянию от заряда. В работе Connell and Utgoff (1987) метод исполнитель–критик применен к задаче балансирования стержня, в которой критик аппроксимировал функцию ценности с помощью ядерной регрессии с весами, обратно пропорциональ-

ными расстоянию. Поскольку эта работа была написана до возникновения широкого интереса к ядерной регрессии в машинном обучении, авторы не употребляют термин «ядро», а говорят о «методе Шепарда» (Shepard, 1968). Из других подходов к ядерным методам в обучении с подкреплением отметим работы Ormoneit and Sen (2002), Dietterich and Wang (2002), Xu, Xie, Hu, and Lu (2005), Taylor and Parr (2009), Barreto, Precup, and Pineau (2011), and Bhat, Farias, and Moallemi (2012).

9.11 Об эмфатических TD-методах см. библиографические замечания к разделу 11.8.

Самый ранний из известных нам примеров применения методов аппроксимации функций к обучению функций ценности – программа игры в шашки Сэмюэла (Samuel, 1959, 1967). Сэмюэл последовал предложению Шеннона (Shannon, 1950) о том, что функции ценности необязательно быть точной, чтобы приносить пользу при выборе ходов в игре и что ее можно аппроксимировать линейной комбинацией признаков. Помимо линейной аппроксимации, Сэмюэл экспериментировал с таблицами соответствия и иерархическими таблицами соответствия, называемыми таблицами сигнатур (Griffith, 1966, 1974; Page, 1977; Biermann, Fairfield, and Beres, 1982).

Примерно в одно время с работой Сэмюэла вышла статья Bellman and Dreyfus (1959), в которой авторы предложили использовать методы аппроксимации функций в ДП. (Возникает искушение подумать, что Беллман и Сэмюэл как-то повлияли друг на друга, но нам не известно ни о каких взаимных ссылках в их работах.) В настоящее время существует довольно обширная литература по методам аппроксимации функций и ДП, в т. ч. по многосеточным методам и методам с использованием сплайнов и ортогональных полиномов (см., например, Bellman and Dreyfus, 1959; Bellman, Kalaba, and Kotkin, 1973; Daniel, 1976; Whitt, 1978; Reetz, 1977; Schweitzer and Seidmann, 1985; Chow and Tsitsiklis, 1991; Kushner and Dupuis, 1992; Rust, 1996).

В системе классификаторов Холланда (Holland, 1986) техника избирательного сравнения признаков использовалась для обобщения информации об оценивании на новые пары состояние–действие. Каждый классификатор сравнивал подмножество состояний, имеющих заданные ценности, с некоторым подмножеством признаков, тогда как остальные признаки могли иметь произвольные значения. Затем эти подмножества использовались в традиционном подходе к аппроксимации функций на основе агрегирования состояний. Холланд хотел использовать генетический алгоритм для эволюции множества классификаторов, которые в совокупности реализовывали бы полезную функцию ценности действий. Идеи Холланда оказали влияние на ранние исследования по обучению с подкреплением, но мы акцентировали внимание на других подходах к аппроксимации. В роли аппроксиматоров функций классификаторы имеют ряд ограничений. Во-первых, это методы агрегирования состояний со всеми присущими им ограничениями в части масштабирования и эффективного представления гладких функций. Кроме того, правила сравнения классификаторов позволяют реализовать только границы агрегирования, параллельные осям признаков. Но, быть может, самое важное ограничение заключается в том, что классификаторы обучаются с помощью генетического алгоритма, относящегося к числу эволю-

ционных методов. В главе 1 мы говорили, что в процессе обучения информации о том, как производить обучение, гораздо больше, чем могут использовать эволюционные методы. И потому вместо них мы решили адаптировать к обучению с подкреплением методы обучения с учителем, а конкретно методы градиентного спуска и ИНС. Эти различия между подходом Холланда и нашим неудивительны, поскольку идеи Холланда появились в то время, когда ИНС считались слишком слабыми в вычислительном плане, чтобы быть полезными, тогда как наша работа относится к началу периода, в котором это устоявшееся мнение было подвергнуто большим сомнениям. Впрочем, остается много возможностей для комбинирования различных аспектов этих подходов.

В работе Christensen and Korf (1986) описаны эксперименты с регрессионными методами для модификации коэффициентов линейных аппроксимаций функции ценности для игры в шахматы. В работах Chapman and Kaelbling (1991) и Tan (1991) методы решающих деревьев адаптированы к обучению функций ценности. С той же целью были адаптированы методы обучения на основе объяснения, дающие компактные представления (Yee, Saxena, Utgoff, and Barto, 1990; Dietterich and Flann, 1995).

Глава 10

Управление с единой стратегией и аппроксимацией

В этой главе мы вернемся к задаче управления, но теперь с параметрической аппроксимацией функции ценности действий $\hat{q}(s, a, \mathbf{w}) \approx q_*(s, a)$, где $\mathbf{w} \in \mathbb{R}^d$ – конечномерный вектор весов. Мы снова ограничимся только случаем с единой стратегией, отложив рассмотрение методов с разделенной стратегией до главы 11. В настоящей главе рассматривается полугradientный алгоритм Sarsa, естественное обобщение полугradientного TD(0) (см. предыдущую главу) на ценности действий и управление с единой стратегией. В эпизодическом случае обобщение прямолинейно, но в непрерывном случае нам придется отступить на несколько шагов назад и заново проанализировать, как мы использовали обесценивание при определении оптимальной стратегии. Как это ни удивительно, но, располагая истинной аппроксимацией функций, мы должны будем отказаться от обесценивания и перейти к новой постановке задачи управления со «средним вознаграждением» и новыми «дифференциальными» функциями ценности.

Начав с эпизодического случая, мы распространим идеи аппроксимации функций, представленные в предыдущей главе, с ценности состояний на ценность действий. Затем мы распространим их на управление, следуя общему паттерну ОИС с единой стратегией, который предполагает ε -жадный выбор действий. Мы продемонстрируем результаты n -шагового линейного алгоритма Sarsa на примере задачи о машине на горе. Затем перейдем к непрерывному случаю и разовьем те же идеи применительно к среднему вознаграждению с дифференциальными ценностями.

10.1. ЭПИЗОДИЧЕСКОЕ ПОЛУГРАДИЕНТНОЕ УПРАВЛЕНИЕ

Обобщение полугradientных методов предсказания из главы 9 на ценности действий не вызывает трудностей. В данном случае мы имеем приближенную функцию ценности действий $\hat{q} \approx q_*$, представленную в параметрической форме с вектором весов $\mathbf{w}$. Если раньше мы рассматривали случайные обучающие примеры вида $S_t \mapsto U_t$, то теперь будем рассматривать примеры вида $S_t, A_t \mapsto U_t$. Целью об-

новления U_t может быть любая аппроксимация $q_\pi(S_t, A_t)$, включая такие привычные обновленные ценности, как доход Монте-Карло (G_t) или любой из n -шаговых доходов Sarsa (7.4). Обновление предсказания ценности действия по методу градиентного спуска в общем виде записывается так:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha[U_t - \hat{q}(S_t, A_t, \mathbf{w}_t)]\nabla \hat{q}(S_t, A_t, \mathbf{w}_t). \quad (10.1)$$

Например, обновление для одношагового метода Sarsa имеет вид:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha[R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)]\nabla \hat{q}(S_t, A_t, \mathbf{w}_t). \quad (10.2)$$

Мы называем этот метод *эпизодическим полугradientным одношаговым Sarsa*. Для постоянной стратегии этот метод сходится так же, как TD(0), и имеет такую же границу ошибки (9.14).

Для формирования методов управления мы должны объединить подобные методы предсказания ценности действий с методами улучшения стратегии и выбора действий. В области методов, применимых к непрерывным действиям или действиям, выбираемым из больших дискретных множеств, пока ведутся активные исследования и ясного решения еще не существует. С другой стороны, если множество действий дискретно и не слишком велико, то можно использовать методы, разработанные в предыдущих главах. Это означает, что для каждого действия a , доступного в текущем состоянии S_t , мы можем вычислить $\hat{q}(S_t, a, \mathbf{w}_t)$, а затем найти жадное действие $A_t^* = \arg \max_a \hat{q}(S_t, a, \mathbf{w}_t)$. После этого производится улучшение стратегии (в случае единой стратегии, рассматриваемом в этой главе), для чего мы переходим к мягкой аппроксимации жадной стратегии, например к выбору ε -жадной стратегии. Действия выбираются в соответствии с той же самой стратегией. Полный псевдокод алгоритма приведен во врезке ниже.

Эпизодический полугradientный Sarsa для оценивания $\hat{q} = q_*$

Вход: дифференцируемая параметризация функции ценности действий $\hat{q}$:
 $S \times \mathcal{A} \times \mathbb{R}^d \rightarrow \mathbb{R}$

Параметры алгоритма: размер шага $\alpha > 0$, небольшое $\varepsilon > 0$

Инициализировать веса функции ценности $\mathbf{w} \in \mathbb{R}^d$ произвольным образом (например, $\mathbf{w} = \mathbf{0}$)

Повторять для каждого эпизода:

$S, A \leftarrow$ начальное состояние и действие эпизода (например, ε -жадное)

Повторять для каждого шага эпизода:

Предпринять действие A , наблюдать R, S'

Если S' заключительное:

$\mathbf{w} \leftarrow \mathbf{w} + \alpha[R - \hat{q}(S, A, \mathbf{w})]\nabla \hat{q}(S, A, \mathbf{w})$

Перейти к следующему эпизоду

Выбрать A' как функцию от $\hat{q}(S, \cdot, \mathbf{w})$ (например, ε -жадное)

$\mathbf{w} \leftarrow \mathbf{w} + \alpha[R + \gamma \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})]\nabla \hat{q}(S, A, \mathbf{w})$

$S \leftarrow S'$;

$A \leftarrow A'$;

Пример 10.1. Задача о машине на горе. Рассмотрим задачу о движении машины с недостаточно мощным двигателем по крутой горной дороге, как показано в левой части рис. 10.1. Проблема в том, что гравитация сильнее двигателя и даже при полностью открытой дроссельной заслонке машина не может одолеть этот крутой склон. Единственное решение – сначала подняться на противоположный склон, а затем, выжав педаль газа до упора, набрать достаточную инерцию для преодоления подъема, несмотря на то что машина будет замедляться на протяжении всего пути. Это простой пример непрерывной задачи управления, в которой сначала дела идут в некотором смысле хуже (мы отдаляемся от цели), но затем все налаживается. Многие методики управления испытывают большие сложности, сталкиваясь с задачами такого типа, если только разработчик не подскажет решение явно.

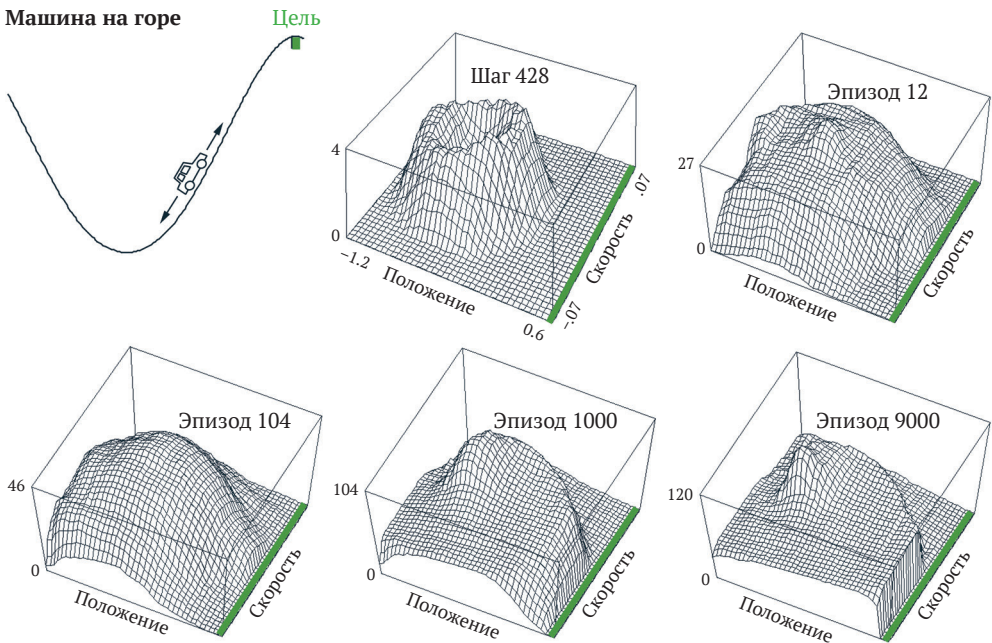


Рис. 10.1 ❖ Задача о машине на горе (слева сверху) и функция затрат на движение ($-\max_a \dot{q}(s, a, \mathbf{w})$), обученная за один прогон

В данной задаче вознаграждение равно -1 на каждом временном шаге до тех пор, пока машина не преодолеет подъем и не окажется на вершине горы. В этот момент эпизод заканчивается. Существует три возможных действия: полный газ вперед ($+1$), полный газ назад (-1) и закрытая дроссельная заслонка (0). Машина движется в соответствии с упрощенными законами физики. Ее положение x_t и скорость $\dot{x}_t$ обновляются следующим образом:

$$x_{t+1} \doteq \text{bound}[x_t + \dot{x}_{t+1}]$$

$$\dot{x}_{t+1} \doteq \text{bound}[\dot{x}_t + 0.001A_t - 0.0025\cos(3x_t)],$$

где операция *bound* гарантирует выполнение неравенств $-1.2 \leq x_{t+1} \leq 0.5$ и $-0.07 \leq \dot{x}_{t+1} \leq 0.07$. Кроме того, когда x_{t+1} достигало левой границы, $\dot{x}_{t+1}$ сбрасывалось в ноль. А когда x_{t+1} достигало правой границы, цель считалась достигнутой и эпизод заканчивался. Каждый эпизод начинался в случайном положении $x_t \in [-0.6, -0.4]$ с нулевой скоростью. Для преобразования непрерывных переменных состояния в бинарные признаки использовались сеточные замощения, как на рис. 9.9. Мы брали 8 замощений, так что каждая плитка покрывала $1/8$ расстояния (подвергнутого операции *bound*) в каждом направлении; при этом сдвиги были асимметричными, как описано в разделе 9.5.4¹. Затем для аппроксимации функции ценности действий строилась линейная комбинация вектора параметров и векторов признаков $\mathbf{x}(s, a)$, созданных в результате плиточного кодирования:

$$\hat{q}(s, a, \mathbf{w}) \doteq \mathbf{w}^T \mathbf{x}(s, a) = \sum_{i=1}^d w_i \cdot x_i(s, a), \quad (10.3)$$

для каждой пары, включающей состояние s и действие a .

На рис. 10.1 показана типичная ситуация, складывающаяся в процессе обучения для решения этой задачи при такой аппроксимации². Мы видим функцию ценности с обратным знаком (функцию затрат на движение), обученную за один прогон. Начальные ценности всех действий равны нулю, что является оптимистичным предположением (истинные ценности в этой задаче отрицательны), вызвавшим интенсивное исследование, несмотря на то что параметр исследования ϵ был равен 0. Это можно наблюдать в средней верхней части рисунка, обозначенной «Шаг 428». В этот момент не завершён даже первый эпизод, но машина совершает колебательное движение во впадине, описывая круговые траектории в пространстве состояний. Ценность всех часто посещаемых состояний хуже, чем неисследованных состояний, потому что фактические вознаграждения были хуже ожидаемых (без всяких на то оснований). Это постоянно уводит агента от уже исследованных состояний к новым, пока не будет найдено решение.

На рис. 10.2 показано несколько кривых обучения для полугradientного Sarsa с различными размерами шага.

¹ Конкретно мы использовали программу плиточного кодирования, размещённую по адресу <http://incompleteideas.net/tiles/tiles3.html>, положив `iht=INT(4096)` и `tiles(iht,8,[8*x/(0.5+1.2),8*xdot/(0.07+0.07)],A)`, чтобы получить индексы единиц в векторе признаков для состояния $(x, \dot{x})$ и действия A .

² В действительности эти данные взяты из алгоритма «полугradientный Sarsa(λ)», с которым мы познакомимся только в главе 12, но полугradientный Sarsa должен вести себя аналогично.

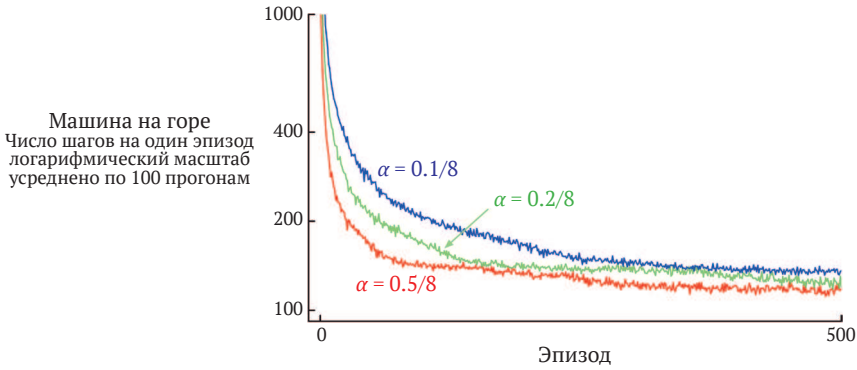


Рис. 10.2 ❖ Кривые обучения в задаче о машине на горе для полуградиентного метода Sarsa с аппроксимацией посредством плиточного кодирования и ϵ -жадным выбором действий

10.2. Полуградиентный n -шаговый Sarsa

Мы можем получить n -шаговый вариант эпизодического полуградиентного Sarsa, если воспользуемся n -шаговым доходом как целью обновления в уравнении обновления полуградиентного Sarsa (10.1). Обобщение табличной формы n -шагового дохода (7.4) на аппроксимацию функций не вызывает затруднений:

$$G_{t:t+n} \doteq R_{t+1} + R_{t+2} + \dots + \gamma^{n-1}R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}), \quad t+n < T, \quad (10.4)$$

где, как обычно, $G_{t:t+n} \doteq G_t$, если $t+n \geq T$. Уравнение n -шагового обновления имеет вид:

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha [G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})] \nabla \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}), \quad 0 \leq t < T. \quad (10.5)$$

Ниже приведен полный псевдокод.

Эпизодический полуградиентный n -шаговый алгоритм Sarsa для оценивания $\hat{q} \approx q_*$ или q_π

Вход: дифференцируемая параметризация функции ценности действий $\hat{q}$:
 $S \times \mathcal{A} \times \mathbb{R}^d \rightarrow \mathbb{R}$

Вход: стратегия π (если оценивается q_π)

Параметры алгоритма: размер шага $\alpha > 0$, небольшое $\epsilon > 0$, положительное целое число n

Произвольно инициализировать веса функции ценности $\mathbf{w} \in \mathbb{R}^d$ (например, $\mathbf{w} = 0$)

Все операции сохранения и доступа (S_t , A_t и R_t) принимают индекс по модулю $n+1$

Повторять для каждого эпизода:

Инициализировать и сохранить $S_0 \neq \text{terminal}$

Выбрать и сохранить действие $A_0 \sim \pi(\cdot | S_0)$ или ε -жадное относительно

$\hat{q}(S_0, \cdot, \mathbf{w})$

$T \leftarrow \infty$

Повторять для $t = 0, 1, 2, \dots$:

Если $t < T$, то:

Предпринять действие A_t

Наблюдать и сохранить следующее вознаграждение как R_{t+1}

и следующее состояние как S_{t+1}

Если S_{t+1} – заключительное состояние, то:

$T \leftarrow t + 1$

иначе:

Выбрать и сохранить действие $A_{t+1} \sim \pi(\cdot | S_{t+1})$ или ε -жадное относительно $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$

$\tau \leftarrow t - n + 1$ (τ – момент времени, для которого обновляется оценка)

Если $\tau \geq 0$:

$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$

Если $\tau + n < T$, то $G \leftarrow G + \gamma^n \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w})$ ($G_{\tau:\tau+n}$)

$\mathbf{w} \leftarrow \mathbf{w} + \alpha [G - \hat{q}(S_\tau, A_\tau, \mathbf{w})] \nabla \hat{q}(S_\tau, A_\tau, \mathbf{w})$

$S \leftarrow S'$

пока не $\tau = T - 1$

Как мы видели ранее, наилучшие результаты получаются при использовании промежуточного уровня бутстрэппинга, соответствующего $n > 1$. На рис. 10.3 показано, что в задаче о машине на горе этот алгоритм обучается быстрее и достигает лучших асимптотических результатов при $n = 8$, чем при $n = 1$. На рис. 10.4 показаны результаты более детального изучения влияния параметров α и n на скорость обучения в этой задаче.

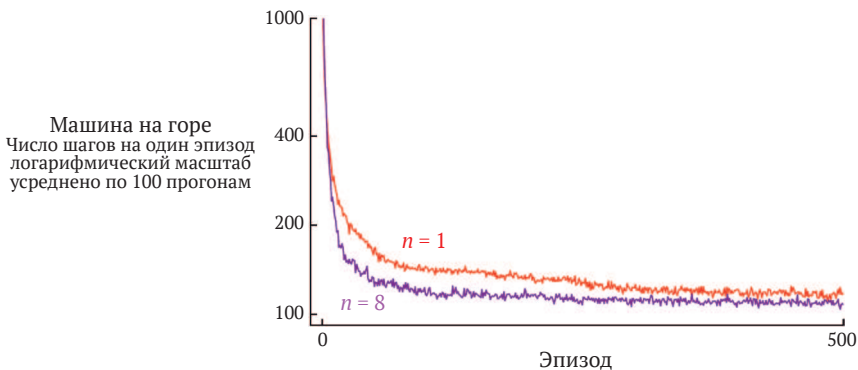


Рис. 10.3 ❖ Качество одношагового и 8-шагового полугradientного Sarsa в задаче о машине на горе. Использовались хорошие размеры шага: $\alpha = 0.5/8$ для $n = 1$ и $\alpha = 0.3/8$ для $n = 8$

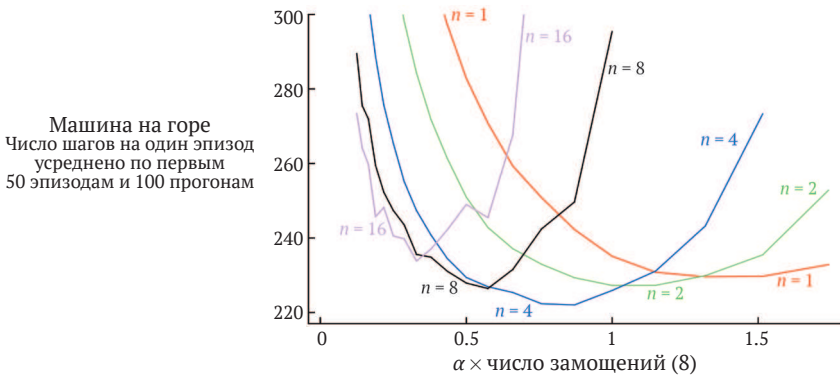


Рис. 10.4 ❖ Влияние α и n на качество n -шагового полугradientного Sarsa и аппроксимации функций посредством плиточного кодирования в задаче о машине на горе. Как обычно, наилучшее качество достигается при промежуточном уровне бутстрэппинга ($n = 4$). Результаты приведены для избранных значений α в логарифмическом масштабе, а затем соединены прямыми линиями. Стандартная ошибка варьировалась от 0.5 (меньше ширины линии) для $n = 1$ до примерно 4 для $n = 16$, поэтому все основные эффекты статистически значимы

Упражнение 10.1. В этой главе мы не рассматривали явно и не приводили псевдокод для методов Монте-Карло. На что они были бы похожи? Почему не имеет смысла приводить для них псевдокод? Какие результаты они дали бы в задаче о машине на горе? □

Упражнение 10.2. Напишите псевдокод полугradientного одношагового метода Expected Sarsa для задачи управления. □

Упражнение 10.3. Почему стандартная ошибка для результатов, показанных на рис. 10.4, выше при больших n , чем при малых? □

10.3. СРЕДНЕЕ ВОЗНАГРАЖДЕНИЕ: НОВАЯ ПОСТАНОВКА НЕПРЕРЫВНЫХ ЗАДАЧ

Теперь мы познакомимся с третьей классической постановкой задачи – наряду с эпизодической и с обесцениванием – для определения цели в марковских процессах принятия решений (МППР). Как и обесценивание, постановка со *средним вознаграждением* применима к непрерывным задачам, в которых взаимодействие между агентом и окружающей средой происходит бесконечно, не завершаясь и не имея начального состояния. Но обесценивание в этом случае отсутствует – отложенные вознаграждения интересны агенту так же, как и непосредственное. Постановка со средним вознаграждением – одна из основных в классическом динамическом программировании, но не так распространена в обучении с подкреплением. В следующем разделе мы увидим, что постановка с обесцениванием вызывает трудности при использовании аппроксимации функций, поэтому ее приходится заменить постановкой со средним вознаграждением.

В постановке со средним вознаграждением качество стратегии π определяется как средний темп вознаграждения или просто как *среднее вознаграждение* при следовании этой стратегии. Мы будем обозначать его $r(\pi)$:

$$r(\pi) \doteq \lim_{h \rightarrow \infty} \frac{1}{h} \sum_{t=1}^h \mathbb{E}[R_t | S_0, A_{0:t-1} \sim \pi] \quad (10.6)$$

$$\begin{aligned} &= \lim_{t \rightarrow \infty} \mathbb{E}[R_t | S_0, A_{0:t-1} \sim \pi], \quad (10.7) \\ &= \sum_s \mu_\pi(s) \sum_s \pi(a|s) \sum_{s', r} p(s', r | s, a) r, \end{aligned}$$

где математические ожидания обусловлены начальным состоянием S_0 и последующими действиями $A_0, A_1, \dots, A_{t-1}$, предпринимаемыми в соответствии со стратегией π . Обозначение μ_π относится к установившемуся распределению, $\mu_\pi(s) \doteq \lim_{t \rightarrow \infty} \Pr\{S_t = s | A_{0:t-1} \sim \pi\}$, которое, по предположению, существует для любой π и не зависит от S_0 . Это свойство МППР называется *эргодичностью*. Оно означает, что состояние, в котором начался процесс, а также решение, принятое агентом вначале, могут иметь только временный эффект; в конечном итоге математическое ожидание пребывания в некотором состоянии зависит только от стратегии и от вероятностей переходов в МППР. Эргодичность – достаточная гарантия существования пределов в приведенных выше равенствах.

Между различными видами оптимальности в непрерывном случае без обесценивания существуют тонкие различия. Тем не менее для большинства практических целей вполне достаточно просто упорядочить стратегии по их среднему вознаграждению на одном временном шаге, т. е. по $r(\pi)$. Эта величина, по существу, является средним вознаграждением при стратегии π , как следует из (10.7). В частности, мы считаем оптимальными все стратегии, для которых $r(\pi)$ достигает максимума.

Отметим, что установившееся распределение – это специальное распределение такое, что, выбирая действия согласно π , мы будем оставаться в том же самом распределении. То есть такое, для которого

$$\sum_s \mu_\pi(s) \sum_a \pi(a|s) p(s' | s, a) = \mu_\pi(s'). \quad (10.8)$$

В постановке со средним вознаграждением доходы определяются в терминах разностей между вознаграждениями и средним вознаграждением:

$$G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \dots \quad (10.9)$$

Это называется *дифференциальным доходом*, а соответствующие функции ценности – *дифференциальными функциями ценности*. Они определяются так же, как и раньше, и мы будем использовать для них такую же нотацию: $v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_t = s]$ и $q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$ (и аналогично для v_* и q_*). Для дифференциальных функций ценности также имеют место уравнения Беллмана, лишь немного отличающиеся от тех, что мы видели раньше. Нужно просто удалить все γs и заменить все вознаграждения разностью между вознаграждением и истинным средним вознаграждением:

$$\begin{aligned}
v_\pi(s) &= \sum_a \pi(a|s) \sum_{r,s'} p(s', r|s, a) [r - r(\pi) + v_\pi(s')], \\
q_\pi(s, a) &= \sum_{r,s'} p(s', r|s, a) \left[r - r(\pi) + \sum_{a'} \pi(a'|s') q_\pi(s', a') \right], \\
v_*(s) &= \max_a \sum_{r,s'} p(s', r|s, a) \left[r - \max_\pi r(\pi) + v_*(s') \right] \text{ и} \\
q_*(s, a) &= \sum_{r,s'} p(s', r|s, a) \left[r - \max_\pi r(\pi) + \max_{a'} q_*(s', a') \right]
\end{aligned}$$

(см. (3.14), упражнение 3.17, (3.19) и (3.20)).

Существует также дифференциальная форма обеих TD-ошибок:

$$\delta_t \doteq R_{t+1} - \bar{R}_{t+1} + \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t) \quad (10.10)$$

и

$$\delta_t \doteq R_{t+1} - \bar{R}_{t+1} + \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t), \quad (10.11)$$

где $\bar{R}_t$ – оценка в момент t среднего вознаграждения $r(\pi)$. При таких определениях большинство наших алгоритмов и многие теоретические результаты переносятся на постановку со средним вознаграждением вообще без изменений.

Например, вариант полугradientного Sarsa в постановке со средним вознаграждением определяется так же, как в (10.2), только используется дифференциальная версия TD-ошибки. То есть он описывается уравнением

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \delta_t \nabla \hat{q}(S_t, A_t, \mathbf{w}_t), \quad (10.12)$$

где δ_t определено, как в (10.11). Полный псевдокод алгоритма приведен во врезке ниже.

Упражнение 10.4. Напишите псевдокод дифференциальной версии полугradientного Q-обучения. □

Упражнение 10.5. Какие уравнения (кроме 10.10) необходимы для определения дифференциальной версии алгоритма TD(0)? □

Дифференциальный полугradientный Sarsa для оценивания $\hat{q} = q_*$

Вход: дифференцируемая параметризация функции ценности действий $\hat{q}: \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \rightarrow \mathbb{R}$

Параметры алгоритма: размеры шагов $\alpha, \beta > 0$

Инициализировать веса функции ценности $\mathbf{w} \in \mathbb{R}^d$ произвольным образом (например, $\mathbf{w} = \mathbf{0}$)

Инициализировать оценку среднего вознаграждения $\bar{R} \in \mathbb{R}$ произвольным образом (например, $\bar{R} = 0$)

Инициализировать состояние S и действие A

Повторять для каждого шага:

Предпринять действие A , наблюдать R, S'

Выбрать A' как функцию от $\hat{q}(S', \cdot, \mathbf{w})$ (например, ε -жадное)

$\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$

$\bar{R} \leftarrow \bar{R} + \beta \delta$

$\mathbf{w} \leftarrow \mathbf{w} + \alpha \delta \nabla \hat{q}(S, A, \mathbf{w})$

$S \leftarrow S'$

$A \leftarrow A'$

Упражнение 10.6. Рассмотрим марковский процесс вознаграждения, состоящий из трех состояний A, B, C ; переходы производятся детерминированно по кругу. Вознаграждение $+1$ начисляется по прибытии в состояние A , в остальных случаях оно равно 0 . Каковы дифференциальные ценности всех трех состояний? $\square$

Пример 10.2. Задача о контроле доступа с очередностью. В этой задаче речь идет о контроле доступа к группе из 10 серверов. Пользователи, имеющие один из четырех приоритетов, ставятся в одну очередь. Получив доступ к серверу, пользователь выплачивает серверу вознаграждение 1, 2, 4 или 8 в зависимости от своего приоритета. На каждом временном шаге пользователь в начале очереди либо принимается (ему назначается один из серверов), либо отвергается (удаляется из очереди с вознаграждением 0). В любом случае на следующем шаге рассматривается следующий пользователь в очереди. Очередь никогда не бывает пустой, и приоритеты пользователей в очереди случайны и равномерно распределены. Разумеется, пользователь не может быть обслужен, если нет свободного сервера, в таком случае пользователь безусловно отвергается. Занятый сервер становится свободным с вероятностью $p = 0.06$ на каждом временном шаге. Предположим, что статистика постановки в очередь и выбытия из нее неизвестна, хотя мы только что описали ее для определенности. Задача заключается в том, чтобы на каждом шаге решить, принять или отвергнуть следующего пользователя, если известен его приоритет и количество свободных серверов. При этом цель – максимизировать вознаграждение за длительный период времени без обесценивания.

Мы рассмотрим табличное решение этой задачи. Хотя обобщения между состояниями нет, мы все же можем рассматривать задачу как общую аппроксимацию функции, поскольку эта постановка обобщает табличную. Таким образом, мы имеем дифференциальную оценку ценности действий для каждой пары состояние–действие, где под состоянием понимается количество свободных серверов и приоритет пользователя в начале очереди, а действие может быть «принять» или «отвергнуть». На рис. 10.5 показано решение, найденное дифференциальным полугradientным методом Sarsa с параметрами $\alpha = 0.01, \beta = 0.01, \varepsilon = 0.1$. Начальные ценности действий и $\bar{R}$ были равны 0.

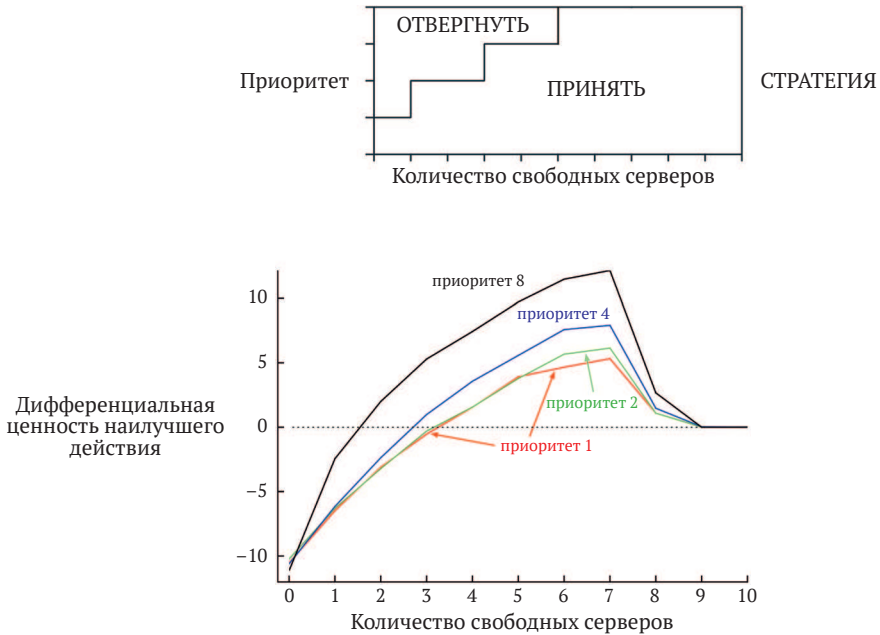


Рис. 10.5 ❖ Стратегия и функция ценности, найденные дифференциальным полу-градиентным одношаговым методом Sarsa в задаче о контроле доступа с очередностью после 2 млн шагов. Резкий спад в правой части графика, вероятно, объясняется недостаточным объемом данных; многие из этих состояний никогда не встречались. Обученное значение $\bar{R}$ равно приблизительно 2.31

Упражнение 10.7. Предположим, что имеется МППР, который при любой стратегии порождает детерминированную бесконечную последовательность вознаграждений $+1, 0, +1, 0, +1, 0, \dots$. Технически это недопустимо, потому что нарушается свойство эргодичности – не существует стационарного предельного распределения μ_π и предела (10.7). Тем не менее среднее вознаграждение (10.6) определено корректно. Чему оно равно? Теперь рассмотрим два состояния этого МППР. При старте из состояния А последовательность вознаграждений точно такая, как описано выше, т. е. начинается с $+1$, а при старте из В последовательность вознаграждений начинается с 0 и продолжается так: $+1, 0, +1, 0, \dots$. В этом случае дифференциальный доход (10.9) определен некорректно, потому что предела не существует. Чтобы исправить ситуацию, можно было бы определить ценность состояния по-другому:

$$v_\pi(s) \doteq \lim_{\gamma \rightarrow 1} \lim_{h \rightarrow \infty} \sum_{t=0}^h \gamma^t (\mathbb{E}_\pi[R_{t+1} | S_0 = s] - r(\pi)). \quad (10.13)$$

Каковы ценности состояний А и В при таком определении? □

Упражнение 10.8. Псевдокод во врезке на стр. 296 обновляет $\bar{R}_{t+1}$, используя в качестве ошибки δ_t , а не просто $R_{t+1} - \bar{R}_{t+1}$. Годятся оба определения ошибки, но δ_t дает лучшие результаты. Чтобы понять, почему, рассмотрим круговой МППР с тремя состояниями из упражнения 10.6. Оценка среднего вознаграждения долж-

на стремиться к своему истинному значению $1/3$. Предположим, что она уже принимала это значение и там и застряла. Какой тогда была бы последовательность $R_t - \bar{R}_t$? А какой была бы последовательность δ_t (вычисленная по формуле (10.10))? Какая последовательность давала бы более устойчивую оценку среднего вознаграждения, если бы оценке было разрешено изменяться в ответ на ошибки? Объясните свой ответ. $\square$

10.4. ВОЗРАЖЕНИЯ ПРОТИВ ПОСТАНОВКИ С ОБЕСЦЕНИВАНИЕМ

Непрерывная постановка задачи с обесцениванием была очень полезна в табличном случае, когда доход в каждом состоянии можно было идентифицировать и усреднить по отдельности. Но в приближенном случае использование этой постановки вызывает большие сомнения.

Чтобы разобраться, почему, рассмотрим бесконечную последовательность доходов без начала и конца, когда нет четко идентифицируемых состояний. Скажем, состояния могли бы быть представлены только векторами признаков, что не позволяет различить два состояния. В частности, все векторы признаков могут быть одинаковыми. Таким образом, имеется лишь последовательность вознаграждений (и действий), и только по ним и следует оценивать качество алгоритма. Как это сделать? Один из возможных подходов – усреднить вознаграждения за длительное время, это и есть идея постановки со средним вознаграждением. Как можно было бы использовать обесценивание? На каждом временном шаге мы могли бы измерять обесцененный доход. Одни доходы были бы малыми, другие большими, так что нам снова пришлось бы их усреднять за достаточно долгое время. В непрерывной постановке начальных и конечных состояний нет, как нет и специальных временных шагов, поэтому больше ничего сделать нельзя. Но если так поступить, то окажется, что средний обесцененный доход пропорционален среднему вознаграждению. На самом деле для стратегии π средний обесцененный доход всегда равен $r(\pi)/(1 - \gamma)$, т. е., по существу, не отличается от среднего вознаграждения $r(\pi)$. В частности, упорядочение стратегий в постановке со средним обесцененным доходом будет точно таким же, как в постановке со средним вознаграждением. Поэтому коэффициент обесценивания не оказывает никакого влияния на постановку задачи. Даже если бы он был равен 0, ранжирование не изменилось бы.

Этот удивительный факт доказан во врезке ниже, но основная идея понятна из соображений симметрии. Каждый временной шаг ничем не отличается от любого другого. В случае обесценивания каждое вознаграждение встречается ровно один раз в каждой позиции некоторого дохода. t -е вознаграждение окажется необесцененным в $(t - 1)$ -м доходе, обесцененным один раз в $(t - 2)$ -м доходе и обесцененным 999 раз в $(t - 1000)$ -м доходе. Таким образом, вес t -го вознаграждения равен $1 + \gamma + \gamma^2 + \gamma^3 + \dots = 1/(1 - \gamma)$. Поскольку все состояния одинаковы, каждому из них сопоставлен такой вес, и, следовательно, средний доход будет в такое число раз больше среднего вознаграждения, т. е. равен $r(\pi)/(1 - \gamma)$.

Этот пример и более общее рассуждение во врезке показывают, что если бы мы попытались оптимизировать обесцененную ценность по распределению с единой стратегией, то получили бы такой же результат, как при оптимизации

необесцененного среднего вознаграждения, а значение γ не играет никакой роли. Это сильный аргумент в пользу того, что обесценивание не играет роли в постановке задачи управления с аппроксимацией функций. Тем не менее обесценивание можно использовать в методах решения. Параметр обесценивания тогда становится не параметром задачи, а параметром метода решения! Однако в этом случае, к сожалению, нет никаких гарантий оптимизации среднего вознаграждения (или эквивалентной обесцененной ценности по распределению с единой стратегией).

Тщетность обесценивания в непрерывных задачах

Быть может, обесценивание удалось бы сохранить, выбрав целевую функцию, которая суммирует обесцененные ценности по распределению, согласно которому возникают состояния при следовании стратегии:

$$\begin{aligned}
 J(\pi) &= \sum_s \gamma_\pi(s) v_\pi^\gamma(s) && (\text{где } v_\pi^\gamma - \text{функция ценности с обесцениванием}) \\
 &= \sum_s \gamma_\pi(s) \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma v_\pi^\gamma(s')] && (\text{уравнение Беллмана}) \\
 &= r(\pi) + \sum_s \gamma_\pi(s) \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \gamma v_\pi^\gamma(s') && (\text{в силу (10.7)}) \\
 &= r(\pi) + \gamma \sum_{s'} v_\pi^\gamma(s') \sum_s \gamma_\pi(s) \sum_a \pi(a|s) p(s'|s, a) && (\text{в силу (3.4)}) \\
 &= r(\pi) + \gamma \sum_{s'} v_\pi^\gamma(s') \gamma_\pi(s') && (\text{в силу (10.8)}) \\
 &= r(\pi) + \gamma J(\pi) \\
 &= r(\pi) + \gamma r(\pi) + \gamma^2 J(\pi) \\
 &= r(\pi) + \gamma r(\pi) + \gamma^2 r(\pi) + \gamma^3 r(\pi) + \dots \\
 &= \frac{1}{1 - \gamma} r(\pi).
 \end{aligned}$$

Предложенная целевая функция с обесцениванием упорядочивает стратегии в точности так, как целевая функция без обесценивания (среднее вознаграждение). Коэффициент обесценивания не влияет на порядок!

Принципиальная причина трудностей при постановке задачи управления с обесцениванием состоит в том, что из-за аппроксимации функций перестает быть справедливой теорема об улучшении стратегии (раздел 4.2). Уже нельзя утверждать, что если мы изменим стратегию, так чтобы улучшить обесцененную ценность одного состояния, то и вся стратегия гарантированно станет лучше в некотором полезном смысле. Эта гарантия была ключом ко всей теории, стоящей за нашими методами обучения с подкреплением для задачи управления. А согласившись на аппроксимацию функций, мы ее потеряли!

На самом деле отсутствие теоремы об улучшении стратегии является также теоретическим пробелом в полностью эпизодической постановке и постановке

со средним вознаграждением. Допустив аппроксимацию функций, мы больше не можем гарантировать улучшение ни в какой постановке. В главе 13 мы познакомимся с альтернативным классом алгоритмов обучения с подкреплением, основанным на параметрических стратегиях, и вот там у нас будет теоретическая гарантия в виде «теоремы о градиенте стратегии», которая играет ту же роль, что теорема об улучшении стратегии. Но для методов, которые обучаются ценностям действий, в настоящее время, похоже, нет гарантий локального улучшения (быть может, подход, описанный в работе Perkins and Precup (2003), может дать частичный ответ на этот вопрос). Мы знаем, что переход к ε -жадности иногда приводит к худшей стратегии, поскольку возможно колебание между хорошими стратегиями вместо сходимости (Gordon, 1996a). В этой области много открытых теоретических вопросов.

10.5. ДИФФЕРЕНЦИАЛЬНЫЙ ПОЛУГРАДИЕНТНЫЙ n -ШАГОВЫЙ SARSA

Для обобщения n -шагового бутстрэппинга нам нужен n -шаговый вариант TD-ошибки. Начнем с обобщения n -шагового дохода (7.4). Вот как выглядит дифференциальная форма в случае аппроксимации функций:

$$G_{t:t+n} \doteq R_{t+1} - \bar{R}_{t+1} + \bar{R}_{t+2} - \bar{R}_{t+2} + \dots + R_{t+n} - \bar{R}_{t+n} + \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}), \quad (10.14)$$

где $\bar{R}$ — оценка $r(\pi)$, $n \geq 1$ и $t + n < T$. Если $t + n \geq T$, то мы, как обычно, определяем $G_{t:t+n} \doteq G_t$. Тогда n -шаговая TD-ошибка имеет вид:

$$\delta_t \doteq G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}), \quad (10.15)$$

после чего мы можем применить обновление (10.12) для обычного полугradientного Sarsa. Полный псевдокод алгоритма приведен во врезке ниже.

Дифференциальный полугradientный n -шаговый алгоритм Sarsa для оценивания $\hat{q} \approx q_*$ или q_π

Вход: дифференцируемая функция $\hat{q}: \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \rightarrow \mathbb{R}$, стратегия π
 Произвольно инициализировать веса функции ценности $\mathbf{w} \in \mathbb{R}^d$ (например, $\mathbf{w} = \mathbf{0}$)
 Произвольно инициализировать оценку среднего вознаграждения (например, $\bar{R} = 0$)
 Параметры алгоритма: размеры шагов $\alpha, \beta > 0$, положительное целое число n
 Все операции сохранения и доступа (S_t, A_t и R_t) принимают индекс по модулю $n + 1$

Инициализировать и сохранить S_0 и A_0
 Повторять для каждого шага $t = 0, 1, 2, \dots$:
 Предпринять действие A_t

Наблюдать и сохранить следующее вознаграждение как R_{t+1} и следующее состояние как S_{t+1}
 Выбрать и сохранить действие $A_{t+1} \sim \pi(\cdot | S_{t+1})$ или ε -жадное относительно $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
 $\tau \leftarrow t - n + 1$ (τ – момент времени, для которого обновляется оценка)
 Если $\tau \geq 0$:
 $\delta \leftarrow *114*(R_t - \bar{R}) + \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w}) - \hat{q}(\bar{S}, \bar{A}, \mathbf{w})$
 $\bar{R} \leftarrow \bar{R} + \beta \delta$
 $\mathbf{w} \leftarrow \mathbf{w} + \alpha \delta \nabla \hat{q}(S_\tau, A_\tau, \mathbf{w})$

Упражнение 10.9. В дифференциальном полугradientном n -шаговом алгоритме Sarsa размер шага для среднего вознаграждения β должен быть очень малым, чтобы величина $\bar{R}$ была хорошей оценкой среднего вознаграждения в долгосрочной перспективе. К сожалению, тогда $\bar{R}$ для многих шагов будет смещена на свое начальное значение, что может сделать обучение неэффективным. Альтернативно можно было бы использовать в качестве $\bar{R}$ выборочное среднее наблюдаемых вознаграждений. Тогда вначале адаптация происходила бы быстро, но в долгосрочной перспективе все равно замедлилась бы. При медленном изменении стратегии $\bar{R}$ также изменялась бы; из-за возможности такой долгосрочной нестационарности методы с выборочным средним не годятся. На самом деле размер шага для среднего вознаграждения – идеальное место для использования описанного в упражнении 2.7 приема с несмещенным постоянным размером шага. Опишите конкретные изменения, которые нужно внести в приведенный выше дифференциальный полугradientный n -шаговый алгоритм Sarsa с этой целью. $\square$

10.6. РЕЗЮМЕ

В этой главе мы обобщили введенные в предыдущей главе идеи параметрической аппроксимации функций и полугradientного спуска на задачу управления. Обобщение тривиально в эпизодическом случае, но в непрерывном случае нам пришлось ввести новую постановку задачи, основанную на максимизации среднего вознаграждения на каждом временном шаге. Удивительно, что при наличии аппроксимации постановка с обесцениванием не переносится на задачу управления. В этом случае большинство стратегий нельзя представить с помощью функции ценности. Оставшиеся произвольные стратегии необходимо ранжировать, и скалярное среднее вознаграждение $r(\pi)$ дает эффективный способ сделать это.

Постановка со средним вознаграждением несет с собой новые дифференциальные варианты функций ценности, уравнений Беллмана и TD-ошибок, но все они являются аналогами прежних, концептуальных изменений мало. Появляется также новый параллельный набор дифференциальных алгоритмов для случая среднего вознаграждения.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

- 10.1** Полугradientный алгоритм Sarsa с аппроксимацией функций впервые был изучен в работе Rummery and Niranjan (1994). Линейный полугradientный Sarsa с ε -жадным выбором действий не сходится в обычном смысле, но входит в ограниченную окрестность оптимального решения (Gordon, 1996a, 2001). В работе Precup and Perkins (2003) доказана сходимость в постановке с дифференциальным выбором действий. См. также работы Perkins and Pendrith (2002) и Melo, Meyn, and Ribeiro (2008). Пример с машиной на горе основан на похожей задаче, изученной в работе Moore (1990), но приведенная здесь формулировка взята из работы Sutton (1996).
- 10.2** Эпизодический n -шаговый полугradientный алгоритм Sarsa основан на прямом алгоритме Sarsa(λ) из работы van Seijen (2016). Приведенные здесь эмпирические результаты ранее не публиковались.
- 10.3** Постановка со средним вознаграждением была описана для динамического программирования (см., например, Puterman, 1994) и с точки зрения обучения с подкреплением (Mahadevan, 1996; Tadepalli and Ok, 1994; Bertsekas and Tsitsiklis, 1996; Tsitsiklis and Van Roy, 1999). Описанный здесь алгоритм – аналог с единой стратегией алгоритма «R-обучения» из работы Schwartz (1993). Название «R-обучение», вероятно, намекает на то, что это преемник Q-обучения, но мы предпочитаем считать, что это отсылка на обучение дифференциальных, или *относительных* (relative), ценностей. Пример управления доступом с очередностью навеян работой Carlström and Nordström (1997).
- 10.4** Ограничения обесценивания при постановке задачи обучения с подкреплением с аппроксимацией функций стали очевидны авторам вскоре после выхода первого издания этой книги. Работа Singh, Jaakkola, and Jordan (1994), по-видимому, стала первой печатной публикацией, где этот факт был отмечен.

Глава 11

*Методы с разделенной стратегией и аппроксимацией

Начиная с главы 5 мы рассматривали методы обучения с единой и разделенной стратегией в основном как два способа разрешить конфликт между исследованием и использованием, неотъемлемый от подхода к обучению на основе обобщенной итерации по стратегиям. В двух предыдущих главах рассматривался случай *единой* стратегии с аппроксимацией функций, а в этой мы рассмотрим случай *разделенной* стратегии. Его обобщение на аппроксимацию функций оказывается совсем иным и значительно более трудным, чем в случае обучения с единой стратегией. Табличные методы с разделенной стратегией, разработанные в главах 6 и 7, легко превратить в полугradientные алгоритмы, но эти алгоритмы не сходятся так же устойчиво, как при обучении с единой стратегией. В этой главе мы изучим проблемы сходимости, ближе познакомимся с теорией линейной аппроксимации, введем понятие обучаемости, а затем обсудим новые алгоритмы с более сильными гарантиями сходимости для случая с разделенной стратегией. В итоге мы будем иметь улучшенные методы, но теоретические результаты будут не такими сильными, а эмпирические – не такими удовлетворительными, как при обучении с единой стратегией. Попутно мы лучше поймем природу аппроксимации в обучении с подкреплением с единой и разделенной стратегией.

Напомним, что в обучении с подкреплением с разделенной стратегией наша цель – обучить функцию ценности для *целевой стратегии* π , располагая данными о другой *поведенческой* стратегии b . В случае задачи предсказания обе стратегии статические и заданы заранее, а требуется обучиться либо ценностям состояний $\hat{v} \approx v_\pi$, либо ценностям действий $\hat{q} \approx q_\pi$. В случае задачи управления требуется обучиться ценностям действий, а обе стратегии, как правило, изменяются в процессе обучения – π является жадной стратегией относительно $\hat{q}$, а b в большей степени исследовательской, например ϵ -жадной относительно $\hat{q}$.

Проблему обучения с разделенной стратегией можно разделить на две части, одна возникает в табличном случае, другая – только в случае аппроксимации функций. Первая часть касается цели обновления (не путать с целевой страте-

гией), вторая – распределения обновлений. Техники, связанные с выборкой по значимости, разработанные в главах 5 и 7, относятся к первой части; они могут увеличивать дисперсию, но необходимы во всех алгоритмах, претендующих на успех, будь то табличные или приближенные. Обобщение этих техник на случай аппроксимации функций кратко рассматривается в первом разделе этой главы.

Для той части проблемы обучения с разделенной стратегией, которая возникает в случае аппроксимации, нужно кое-что еще, поскольку обновления в случае с разделенной стратегией распределены не в соответствии с распределением с единой стратегией. Распределение с единой стратегией важно для устойчивости полугradientных методов. Для решения проблемы предложено два общих подхода. Один – снова воспользоваться методами выборки по значимости, но на этот раз деформировать распределение обновлений обратно в распределение с единой стратегией, так чтобы полугradientные методы гарантированно сходились (в линейном случае). Второй – разработать настоящие gradientные методы, которым для устойчивости не нужно какое-то специальное распределение. Мы представим методы, основанные как на первом, так и на втором подходе. Это область активных исследований, поэтому пока не ясно, какой подход окажется эффективнее на практике.

11.1. ПОЛУГРАДИЕНТНЫЕ МЕТОДЫ

Начнем с описания того, как методы, разработанные в предыдущих главах для случая разделенной стратегии, обобщаются на случай аппроксимации функций в виде полугradientных методов. Эти методы решают первую часть проблемы обучения с разделенной стратегией (изменение целей обновления), но не решают вторую часть (изменение распределения обновлений). Соответственно, в некоторых случаях они могут расходиться, и в этом смысле их нельзя назвать корректными, но зачастую практическое применение оказывается успешным. Напомним, что эти методы гарантированно устойчивы и асимптотически несмещенные в табличном случае, который является частным случаем аппроксимации функций. Поэтому, возможно, их все-таки можно объединить с методами отбора признаков таким образом, что получившаяся система будет устойчивой. Так или иначе, эти методы просты, поэтому послужат хорошей отправной точкой.

В главе 7 мы описали целый ряд табличных алгоритмов с разделенной стратегией. Чтобы преобразовать их в полугradientную форму, мы просто заменим обновление массива (V или Q) обновлением вектора весов (w), воспользовавшись приближенной функцией ценности ($\hat{v}$ или $\hat{q}$) и ее gradientом. Во многих из этих алгоритмов используется коэффициент выборки по значимости на одном шаге:

$$\rho_t \doteq \rho_{t:t} = \frac{\pi(A_t|S_t)}{b(A_t|S_t)}. \quad (11.1)$$

Например, одношаговый алгоритм ценности состояния является полугradientным TD(0) с разделенной стратегией, который похож на соответствующий алгоритм с единой стратегией (стр. 244) во всем, кроме добавления ρ_t :

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \rho_t \delta_t \nabla \hat{v}(S_t, \mathbf{w}_t), \quad (11.2)$$

где определение δ_t зависит от того, является ли задача эпизодической с обесцениванием или непрерывной без обесценивания со средним вознаграждением:

$$\delta_t \doteq R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t); \quad (11.3)$$

$$\delta_t \doteq R_{t+1} - R_t^- + \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t). \quad (11.4)$$

Для ценности действий одношаговым алгоритмом является полугradientный Expected Sarsa:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \delta_t \nabla \hat{q}(S_t, A_t, \mathbf{w}_t), \quad (11.5)$$

где

$$\delta_t \doteq R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) \hat{q}(S_{t+1}, a, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t) \quad \text{или} \quad (\text{эпизодический})$$

$$\delta_t \doteq R_{t+1} - \bar{R}_t + \sum_a \pi(a|S_{t+1}) \hat{q}(S_{t+1}, a, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t). \quad (\text{непрерывный})$$

Заметим, что в этом алгоритме не используется выборка по значимости. В табличном случае очевидно, что это правильно, потому что единственным выборочным действием является A_t , и, обучаясь его ценности, мы не обязаны рассматривать никакие другие действия. В случае аппроксимации функций это не так очевидно, поскольку мы, возможно, захотим назначать разным парам состояние–действие разные веса, при условии что все они вносят вклад в одну и ту же общую аппроксимацию. Чтобы правильно разрешить этот вопрос, необходимо более глубокое понимание теории аппроксимации функций в обучении с подкреплением, которого пока нет.

Многошаговые обобщения обоих видов алгоритмов – для ценности состояний и для ценности действий – включают выборку по значимости. Например, n -шаговый вариант полугradientного Expected Sarsa имеет вид:

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha \rho_{t+1} \cdots \rho_{t+n-1} [G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})] \nabla \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}), \quad (11.6)$$

где

$$G_{t:t+n} \doteq R_{t+1} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}) \quad \text{или} \quad (\text{эпизодический})$$

$$G_{t:t+n} \doteq R_{t+1} - \bar{R}_t + \cdots + R_{t+n} - \bar{R}_{t+n-1} + \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}), \quad (\text{непрерывный})$$

где мы допустили некоторую вольность в обработке концов эпизодов. В первом уравнении ρ_k для $k \geq T$ (T – последний временной шаг эпизода) следует положить равным 1, а $G_{t:n}$ должно быть равно G_t , если $t + n \geq T$.

Напомним, что в главе 7 мы также представили алгоритм с разделенной стратегией, в котором выборка по значимости не участвует вовсе: n -шаговый алгоритм обновления по дереву. Вот его полугradientный вариант:

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha [G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})] \nabla \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}); \quad (11.7)$$

$$G_{t:t+n} \doteq \hat{q}(S_t, A_t, \mathbf{w}_{t-1}) + \sum_{k=t}^{t+n-1} \delta_k \prod_{i=t+1}^k \gamma \pi(A_i | S_i), \quad (11.8)$$

где δ_t определено так же, как для Expected Sarsa выше. В главе 7 мы тоже определили алгоритм, унифицирующий все алгоритмы ценности действий: n -шаговый $Q(\sigma)$. Оставляем вывод полугradientной формы этого алгоритма, а также n -шагового алгоритма ценности состояний в качестве упражнений для читателя.

Упражнение 11.1. Преобразуйте уравнение (7.9) n -шагового TD-алгоритма с разделенной стратегией в полугradientную форму. Приведите также определения дохода в эпизодическом и непрерывном случаях. $\square$

**Упражнение 11.2.* Преобразуйте уравнения (7.11) и (7.18) n -шагового $Q(\sigma)$ в полугradientную форму. Приведите также определения, охватывающие как эпизодический, так и непрерывный случай. $\square$

11.2. ПРИМЕРЫ РАСХОДИМОСТИ В СЛУЧАЕ С РАЗДЕЛЕННОЙ СТРАТЕГИЕЙ

В этом разделе мы начинаем обсуждение второй части проблемы обучения с разделенной стратегией в случае аппроксимации функции – тот факт, что распределение обновлений не совпадает с распределением с единой стратегией. Мы опишем некоторые поучительные контрпримеры для обучения с разделенной стратегией – случаи, когда полугradientные и другие простые алгоритмы неустойчивы и расходятся.

Для развития интуиции лучше сначала рассмотреть простой пример. Предположим, что имеется два состояния (быть может, являющиеся частью более крупного МППР), для которых оценки ценностей выражены в функциональной форме w и $2w$, где векторы параметров w содержат всего один элемент w . Такое случается при линейной аппроксимации функций, если векторы признаков обоих состояний – просто числа (одномерные векторы), в данном случае 1 и 2. В первом состоянии возможно только одно действие, которое детерминированно переводит процесс во второе состояние с вознаграждением 0:



где выражения в обоих кружках обозначают ценности состояний.

Предположим, что в начальный момент $w = 10$. Затем произойдет переход из состояния с оценкой стоимости 10 в состояние с оценкой стоимости 20. Этот переход выглядит вполне нормально, и w будет увеличено, чтобы поднять оценку стоимости первого состояния. Если γ близко к 1, то TD-ошибка будет близка к 10, и если $\alpha = 0.1$, то w будет увеличено почти до 11 в попытке уменьшить TD-ошибку. Однако оценка ценности второго состояния также увеличится, почти до 22. Если произойдет еще один переход, то он уже будет из состояния с оценкой ценности ≈ 11 в состояние с оценкой ценности ≈ 22 , и, стало быть, TD-ошибка составит ≈ 11 – больше, а не меньше, чем прежде. Ситуация еще сильнее выглядит так, будто первое состояние недооценено, и его ценность снова повышается, теперь до ≈ 12.1 . Это плохо, и на самом деле в процессе последующих обновлений w будет стремиться к бесконечности.

Чтобы окончательно убедиться в этом, нужно более внимательно взглянуть на последовательность обновлений. TD-ошибка после перехода между двумя состояниями равна

$$\delta_t = R_{t+1} + \gamma \hat{V}(S_{t+1}, \mathbf{w}_t) - \hat{V}(S_t, \mathbf{w}_t) = 0 + \gamma 2w_t - w_t = (2\gamma - 1)w_t,$$

а обновление в полугradientном TD(0) с разделенной стратегией (в силу (11.2)) имеет вид:

$$w_{t+1} = w_t + \alpha \rho_t \delta_t \nabla \hat{V}(S_t, w_t) = w_t + \alpha \cdot 1 \cdot (2\gamma - 1)w_t \cdot 1 = (1 + \alpha(2\gamma - 1))w_t.$$

Заметим, что коэффициент выборки по значимости ρ_t на этом переходе равен 1, потому что в первом состоянии существует только одно действие, так что его вероятность что при целевой, что при поведенческой стратегии должна быть равна 1. В окончательном обновлении, показанном выше, новый параметр равен старому, умноженному на скалярную постоянную $1 + \alpha(2\gamma - 1)$. Если эта постоянная больше 1, то система неустойчива и w будет стремиться к положительной или отрицательной бесконечности в зависимости от начального значения. В данном случае эта постоянная больше 1, если $\gamma > 0.5$. Заметим, что устойчивость не зависит от размера шага, если только $\alpha > 0$. От размера шага зависит скорость стремления w к бесконечности, но не сам факт расходимости.

Главным в этом примере является то, что один и тот же переход происходит многократно без обновления w на других переходах. Это возможно при обучении с разделенной стратегией, поскольку поведенческая стратегия может выбирать на этих других переходах такие действия, которые целевая стратегия никогда не выбрала бы. Для этих переходов ρ_t было бы равно нулю, и никакие обновления не производились бы. Но при обучении с единой стратегией ρ_t всегда равно 1. Для каждого перехода из состояния w в состояние $2w$, увеличивающего w , должен был бы существовать переход из состояния $2w$. Этот переход должен был бы уменьшать w , если только ценность конечного состояния не выше $2w$ (из-за того, что $\gamma < 1$), а тогда за этим состоянием должно было бы следовать состояние с еще большей ценностью, иначе w все же было бы уменьшено. Каждое состояние может подкреплять предшествующее только путем создания большего математического ожидания. В конечном итоге обязательно настанет час расплаты. В случае единой стратегии необходимо выполнять обещание будущего вознаграждения, и система остается под контролем. Но если стратегия разделена, то можно дать обещание, а затем, предприняв действие, которое целевая стратегия никогда не выбрала бы, не сдержать его и не понести за это наказания.

Этот простой пример многое говорит о том, почему обучение с разделенной стратегией может приводить к расходимости, но он убеждает не до конца, потому что сам неполон – это лишь фрагмент полного МППР. А может ли существовать полная неустойчивая система? Простое подтверждение возможности расходимости дает контрпример Бэрда. Рассмотрим эпизодический МППР с семью состояниями и двумя действиями, показанный на рис. 11.1. Действие *dashed*, обозначенное штриховой линией, с одинаковой вероятностью переводит систему в одно из шести верхних состояний, а действие *solid*, обозначенное сплошной линией, – в седьмое состояние. Поведенческая стратегия b выбирает действия *dashed* и *solid* с вероятностями $6/7$ и $1/7$, так что при этой стратегии распределение следующих

состояний равномерное (вероятности всех незаключительных состояний одинаковы), каковым является и начальное распределение для каждого эпизода. Целевая стратегия π всегда выбирает действие `solid`, поэтому распределение с единой стратегией (а именно π) сконцентрировано в седьмом состоянии. За любой переход начисляется вознаграждение 0. Коэффициент обесценивания $\gamma = 0.99$.

Рассмотрим оценивание ценности состояний для линейной параметризации, показанной выражениями внутри окружностей, представляющих состояния. Например, оценка ценности самого левого состояния равна $2w_1 + w_8$, где нижний индекс выбирает элемент полного вектора весов $\mathbf{w} \in \mathbb{R}^8$; это означает, что вектор признаков для первого состояния равен $\mathbf{x}(1) = (2, 0, 0, 0, 0, 0, 0, 1)^T$. Вознаграждение на всех переходах нулевое, поэтому истинная функция ценности равна $v_\pi(s) = 0$ для всех s , и ее можно аппроксимировать точно, положив $\mathbf{w} = 0$. На самом деле существует много решений, поскольку количество элементов вектора весов (8) больше количества незаключительных состояний (7). Более того, множество векторов признаков $\{\mathbf{x}(s): s \in \mathcal{S}\}$ линейно независимо. С любой из этих точек зрения данная задача кажется подходящей для линейной аппроксимации функций.

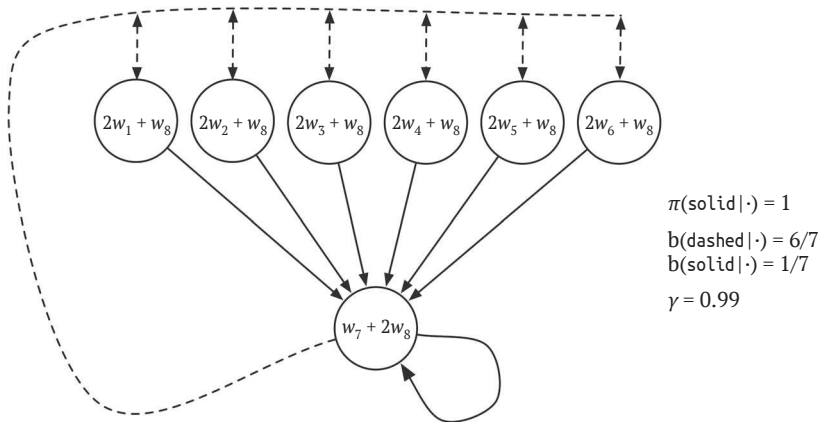


Рис. 11.1 ❖ Контрпример Бэрда. Приближенная функция ценности состояний для этого марковского процесса имеет вид, описываемый линейными выражениями внутри окружностей, представляющих состояния. Действие `solid` обычно переводит систему в седьмое состояние, а действие `dashed` – в одно из остальных шести состояний с одинаковыми вероятностями. Вознаграждение всегда равно нулю

Если применить к этой задаче полугradientный метод TD(0) (11.2), то веса будут стремиться к бесконечности, как показано на рис. 11.2 (слева). Неустойчивость возникает при любом сколь угодно малом, но положительном размере шага. На самом деле она возникает, даже если производится полное обновление, как в динамическом программировании (ДП), что показано на рис. 11.2 (справа). То есть если вектор весов $\mathbf{w}_k$ обновляется для всех состояний одновременно, как в полугradientных методах, с использованием цели ДП (основанной на математическом ожидании):

$$\mathbf{w}_{k+1} \doteq \mathbf{w}_k + \frac{\alpha}{|\mathcal{S}|} \sum_s (\mathbb{E}_\pi[R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_k) | S_t = s] - \hat{v}(s, \mathbf{w}_k)) \nabla \hat{v}(s, \mathbf{w}_k). \quad (11.9)$$

В таком случае нет ни случайности, ни асинхронности – точно так же, как при классическом обновлении в ДП. Это вполне традиционный метод, только в нем используется полугradientная аппроксимация функций. И все-таки система неустойчива.

Если мы изменим в контрпримере Бэрда только распределение ДП-обновлений, перейдя от равномерного к распределению с единой стратегией (для чего обычно требуется асинхронное обновление), то гарантируется сходимость к решению с ошибкой, ограниченной выражением (9.14). Этот пример поразителен, потому что использованные в нем методы TD и ДП считаются самыми простыми и хорошо понимаемыми бутстрэппинговыми методами, а линейный метод полугradientного спуска, наверное, является самым простым и теоретически понятным способом аппроксимации функций. Пример показывает, что даже простейшая комбинация бутстрэппинга и аппроксимации функций может оказаться неустойчивой, если обновления производятся не в соответствии с распределением с единой стратегией.

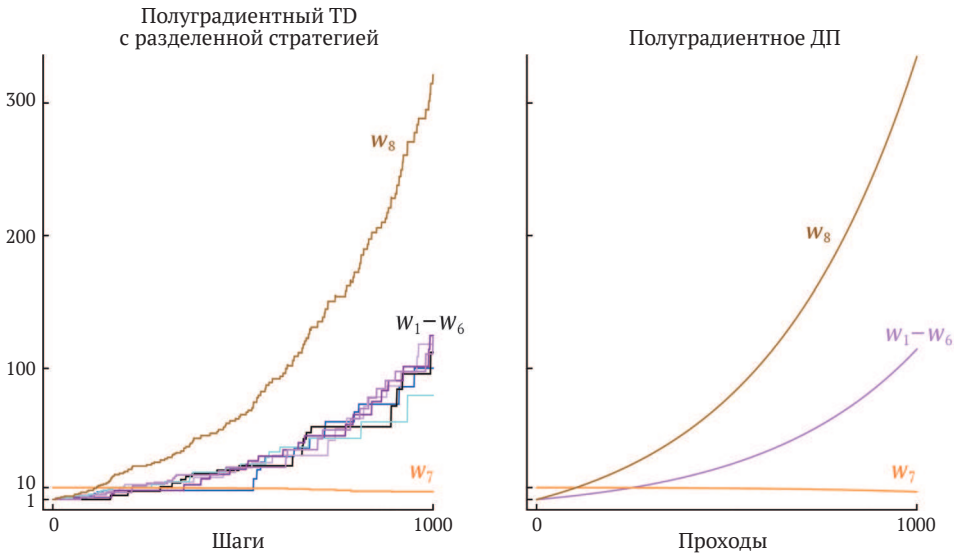


Рис. 11.2 ❖ Демонстрация неустойчивости в контрпримере Бэрда. Показана эволюция элементов вектора весов $\mathbf{w}$ для двух полугradientных алгоритмов. Размер шага $\alpha = 0.01$, а начальные веса $\mathbf{w} = (1, 1, 1, 1, 1, 1, 10, 1)^T$

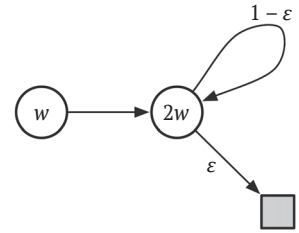
Существуют похожие контрпримеры, демонстрирующие расхождение в случае Q-обучения. Это настораживает, потому что вообще-то из всех методов управления Q-обучение имеет наилучшие гарантии сходимости. Были приложены большие усилия, чтобы найти решение этой проблемы или хотя бы получить более слабые, но все-таки работающие гарантии. Быть может, сходимость Q-обучения

можно гарантировать, если поведенческая стратегия достаточно близка к целевой, скажем является ε -жадной. Насколько нам известно, в этом случае Q-обучение всегда сходится, но теоретический анализ отсутствует. Далее в этом разделе мы изложим несколько других идей, которые были исследованы.

Предположим, что вместо шага в сторону ожидаемого одношагового дохода на каждой итерации, как в контрпримере Бэрда, мы все время изменяем функцию ценности, приближая ее к наилучшей аппроксимации наименьших квадратов. Решит ли это проблему неустойчивости? Конечно, решило бы, если бы векторы признаков $\{x(s): s \in \mathcal{S}\}$ образовывали линейно независимое множество, как в контрпримере Бэрда, потому что тогда на каждой итерации возможна точная аппроксимация, и метод сводится к стандартному табличному ДП. Но интересно рассмотреть случай, когда точное решение невозможно. В таком случае устойчивость не гарантируется, даже если на каждой итерации строить наилучшую аппроксимацию, как показывает следующий пример.

Пример 11.1. Контпример Цицкиса и ван Роя.

В этом примере демонстрируется, что линейная аппроксимация функций не работает с ДП, даже если на каждом шаге ищется решение по методу наименьших квадратов. Он строится путем дополнения примера с переходом из w в $2w$ (из предыдущего раздела) заключительным состоянием, как показано на рисунке справа. Как и раньше, оценка ценности первого состояния равна w , а второго состояния – $2w$. Вознаграждение на всех переходах нулевое, поэтому истинные ценности обоих состояний равны нулю, и это можно точно представить, положив $w = 0$. Если выбирать w_{k+1} на каждом шаге, так чтобы минимизировать величину ошибки $\overline{VE}$ между оценкой ценности и ожидаемым одношаговым доходом, то будем иметь:



$$\begin{aligned}
 w_{k+1} &= \operatorname{argmin}_{w \in \mathbb{R}} \sum_{s \in \mathcal{S}} \left(\hat{v}(s, w) - \mathbb{E}_{\pi} [R_{t+1} + \gamma \hat{v}(S_{t+1}, w_k) | S_t = s] \right)^2 \\
 &= \operatorname{argmin}_{w \in \mathbb{R}} (w - \gamma 2w_k)^2 + (2w - (1 - \varepsilon)\gamma 2w_k)^2 \\
 &= \frac{6 - 4\varepsilon}{5} \gamma w_k.
 \end{aligned} \tag{11.10}$$

Последовательность $\{w_k\}$ расходится, если $\gamma > 5/(6 - 4\varepsilon)$ и $w_0 \neq 0$. ■

Еще один способ предотвратить неустойчивость – пользоваться специальными методами аппроксимации функций. В частности, устойчивость гарантируется для методов, которые не экстраполируют наблюдаемые цели. К числу подобных методов, называемых *усреднителями*, относятся методы ближайших соседей и локально взвешенная регрессия, но не такие популярные методы, как плиточное кодирование и искусственные нейронные сети (ИНС).

Упражнение 11.3 (требуется программирование). Примените одношаговое полу-градиентное Q-обучение к контрпримеру Бэрда и эмпирически покажите, что веса в этом случае расходятся. □

11.3. СМЕРТЕЛЬНАЯ ТРИАДА

Всему сказанному выше можно подвести итог, констатируя, что опасность неустойчивости и расходимости возникает всякий раз, как сходятся вместе все три следующих элемента, называемых *смертельной триадой*:

- **аппроксимация функций.** Мощный и масштабируемый способ обобщения на пространство состояний, значительно превышающее объем доступной памяти и вычислительных ресурсов (например, линейная аппроксимация или ИНС);
- **бутстрэппинг.** Цели обновления, которые включают существующие оценки (как в динамическом программировании или TD-методах), а не полагаются исключительно на фактические вознаграждения и полные доходы (как в методах Монте-Карло);
- **обучение с разделенной стратегией.** Обучение на распределении переходов, отличным от порождаемого целевой стратегией. Проход по пространству состояний и равномерное обновление всех состояний, как в динамическом программировании, игнорируют целевую стратегию и могут служить примером обучения с разделенной стратегией.

Отметим, в частности, что опасность не связана ни с управлением, ни с обобщенной итерацией по стратегиям. Эти случаи труднее поддаются анализу, но неустойчивость возникает и в простой задаче предсказания, если она включает все три элемента смертельной триады. Опасность также не связана с обучением или недостоверностью информации об окружающей среде, поскольку она с таким же успехом возникает и в методах планирования, например в динамическом программировании, когда об окружающей среде известно всё.

Если присутствуют какие-либо два элемента смертельной триады, но не все три сразу, то неустойчивости можно избежать. А раз так, то естественно проанализировать все три и посмотреть, можно ли от какого-то отказаться.

Очевидно, что нельзя отказаться от аппроксимации функций. Нам нужны методы, которые масштабируются на большие задачи и обладают высокой выразительной способностью. По меньшей мере, нам нужна линейная аппроксимация с большим числом признаков и параметров. Агрегирование состояний или непараметрические методы, сложность которых возрастает с увеличением объема данных, оказываются слишком слабыми или слишком дорогостоящими. Методы наименьших квадратов типа LSTD имеют квадратичную сложность и потому непригодны для решения больших задач.

Обойтись без бутстрэппинга можно, пожертвовав вычислительной эффективностью и эффективным использованием данных. Пожалуй, потеря вычислительной эффективности важнее. Для методов Монте-Карло (без бутстрэппинга) требуется память для хранения всего, что происходит между каждым предсказанием и получением окончательного дохода, при этом все вычисления производятся после того, как окончательный доход получен. Такого рода вычислительные затраты не проявляются на последовательных компьютерах фон Неймана, но были бы заметны при использовании специализированного оборудования. Бутстрэппинговые методы и следы приемлемости (глава 12) позволяют работать с данными там и тогда, где и когда они генерируются, после чего необходимости в их ис-

пользовании уже не возникает. Экономия памяти и затрат на передачу данных, обеспечиваемая бутстрэппингом, очень значительна.

Потери в эффективности использования данных при отказе от бутстрэппинга также велики. Мы видели это не один раз, например в главе 7 (рис. 7.2) и в главе 9 (рис. 9.2), где даже небольшой бутстрэппинг позволил добиться гораздо лучших результатов в задаче предсказания случайного блуждания, чем методы Монте-Карло, и в главе 10, где то же самое наблюдалось в задаче об управлении машиной на горе (рис. 10.4). И во многих других задачах обучение происходит гораздо быстрее при наличии бутстрэппинга (см., например, рис. 12.14). Это связано с тем, что бутстрэппинг позволяет воспользоваться при обучении свойством состояния – способностью распознавать состояние при возвращении в него. С другой стороны, бутстрэппинг может помешать обучению в тех задачах, где представление состояний неудачно и ведет к плохой обобщаемости (например, так, похоже, обстоит дело с игрой Тетрис, см. Şimşek, Algórta, and Kothiyal, 2016). Неудачное представление состояния может также стать причиной смещения, а стало быть, худшей границы качества асимптотической аппроксимации (уравнение 9.14). С учетом всех аргументов способность к бутстрэппингу следует признать чрезвычайно ценной. Иногда от него отказываются, выбирая взамен длинные n -шаговые обновления (или большой параметр бутстрэппинга $\lambda \approx 1$, см. главу 12), но обычно бутстрэппинг резко повышает эффективность. Мы очень хотели бы сохранить эту возможность в своем арсенале.

И наконец, обучение с разделенной стратегией – можем ли мы отказаться от него? Методов с единой стратегией часто вполне достаточно. Для безмодельного обучения с подкреплением можно просто использовать алгоритм Sarsa вместо Q-обучения. Методы с разделенной стратегией делают поведение независимым от целевой стратегии. Это можно рассматривать как соблазнительное удобство, но необходимостью оно не является. Однако обучение с разделенной стратегией существенно в других ситуациях – мы их еще не упоминали в этой книге, но они могут быть важны для более широкой задачи создания мощного интеллектуального агента. В таких случаях агент обучается не одной функции ценности и одной стратегии, а большому их количеству параллельно. В психологии имеется много свидетельств в пользу того, что люди и животные обучаются предсказывать много разных событий, воздействующих на органы чувств, а не только вознаграждения. Необычные события могут удивить нас, так что мы скорректируем касающиеся их предсказания, даже если их валентность нейтральна (они не плохие и не хорошие). Такой вид предсказаний, по-видимому, лежит в основе прогностических моделей мира, в т. ч. тех, что мы используем при планировании. Мы предсказываем, что увидим после движения глаз, сколько времени займет дорога домой, вероятность успешного броска в прыжке в баскетболе и удовлетворенность от нового проекта. Во всех этих случаях события, которые мы хотели бы предсказать, зависят от наших действий. Чтобы обучиться всем им параллельно, нужно организовать обучение на одном потоке опыта. Существует много целевых стратегий, поэтому нельзя выбрать одну поведенческую стратегию, которая совпадала бы с каждой из них. И тем не менее параллельное обучение концептуально возможно, потому что поведенческая стратегия может частично перекрываться со многими целевыми. Но чтобы воспользоваться этим в полной мере, необходимо обучение с разделенной стратегией.

11.4. ГЕОМЕТРИЯ ЛИНЕЙНОЙ АППРОКСИМАЦИИ ФУНКЦИЙ ЦЕННОСТИ

Чтобы лучше разобраться в проблеме устойчивости обучения с разделенной стратегией, полезно поразмыслить об аппроксимации функции ценности на более абстрактном уровне и независимо от того, как производится обучение. Представим себе пространство всех возможных функций ценности состояний – всех функций, отображающих состояния в вещественные числа, $v: \mathcal{S} \rightarrow \mathbb{R}$. Большинству таких функций не соответствует никакая стратегия. Но нам важнее тот факт, что большинство их не представимо аппроксиматором функций, который, по построению, имеет меньше параметров, чем существует состояний.

Если представить пространство состояний в виде последовательности $\mathcal{S} = \{s_1, s_2, \dots, s_{|\mathcal{S}|}\}$, то любой функции ценности v будет соответствовать вектор, содержащий ценности каждого состояния $[v(s_1), v(s_2), \dots, v(s_{|\mathcal{S}|})]^T$. Это векторное представление функции ценности содержит столько элементов, сколько имеется состояний. В большинстве случаев, когда мы хотим воспользоваться аппроксимацией функции, элементов слишком много для явного представления вектора. Тем не менее сама идея вектора концептуально полезна. Далее мы считаем функцию ценности и ее векторное представление двумя сторонами одной медали.

Для развития интуиции рассмотрим случай, когда имеется три состояния $\mathcal{S} = \{s_1, s_2, s_3\}$ и два параметра $\mathbf{w} = (w_1, w_2)^T$. Тогда все функции ценности (векторы) можно рассматривать как точки в трехмерном пространстве. Параметры вводят альтернативную систему координат в двумерном подпространстве. Любой вектор весов $\mathbf{w} = (w_1, w_2)^T$ – точка в этом двумерном подпространстве и, следовательно, полноправная функция ценности $v_{\mathbf{w}}$, которая сопоставляет ценности всем трем состояниям. В случае аппроксимации функций общего вида связь между всем пространством и подпространством представимых функций может быть сложной, но в случае *линейной* аппроксимации функций ценности подпространством является обычная плоскость, как показано на рис. 11.3.

Теперь рассмотрим одну фиксированную стратегию π . Предполагается, что истинная функция ценности v_{π} слишком сложна и не может быть аппроксимирована точно. Поэтому v_{π} не принадлежит подпространству, на рисунке она находится выше плоскости представимых функций.

Если v_{π} нельзя представить точно, то какая представимая функция будет ближайшей к ней? Это тонкий вопрос, на который можно ответить по-разному. Для начала нам нужна мера расстояния между двумя функциями ценности. Если имеется две функции ценности v_1 и v_2 , то можно говорить о векторе их разности $v = v_1 - v_2$. Если v мал, то функции ценности близки. Но как измерить длину вектора разности? Традиционная евклидова норма не годится, потому что, как мы говорили в разделе 9.2, некоторые состояния важнее прочих, поскольку чаще встречаются или так как более интересны (раздел 9.11). Как и в разделе 9.2, будем считать, что распределение $\mu: \mathcal{S} \rightarrow [0, 1]$ описывает, в какой степени нам интересны точные ценности различных состояний (часто оно совпадает с распределением с единой стратегией). Тогда можно определить расстояние между функциями ценности с помощью нормы

$$\|v\|_{\mu}^2 \doteq \sum_{s \in \mathcal{S}} \mu(s) v(s)^2. \quad (11.11)$$



Рис. 11.3 ❖ Геометрия линейной аппроксимации функций ценности. Иллюстрируется на примере трехмерного пространства всех функций ценности трех состояний, в котором плоскость – подпространство всех функций ценности, представимых линейным аппроксиматором с параметром $\mathbf{w} = (w_1, w_2)^T$. Истинная функция ценности принадлежит более широкому пространству и может быть спроецирована (на подпространство с помощью оператора проецирования Π) в наилучшую ее аппроксимацию в смысле ошибки ценности (VE). Наилучшие аппроксимации в смысле беллановской ошибки (BE), спроецированной беллановской ошибки (PBE) и TD-ошибки (TDE) потенциально различны и показаны в правом нижнем углу рисунка (VE, BE и PBE рассматриваются на рисунке как соответствующие векторы). Оператор Беллмана отображает функцию ценности на плоскости в функцию ценности вне плоскости, которую можно спроецировать обратно. Если бы мы итеративно применяли оператор Беллмана вне подпространства (показано на рисунке серым цветом), то пришли бы к истинной функции ценности, как в традиционном динамическом программировании. Если бы вместо этого мы продолжили на каждом шаге проецировать на подпространство, как на нижнем шаге, изображенном серым цветом, то неподвижной точкой была бы точка, в которой PBE обращается в ноль

Отметим, что величину $\bar{VE}$ из раздела 9.2 можно записать в виде $\bar{VE}(\mathbf{w}) = \|v_w - v_\pi\|^2$. Для любой функции ценности v операция нахождения ближайшей функции в подпространстве представимых функций ценности – это проекция. Определим оператор проецирования Π , который отображает произвольную функцию ценности в представимую функцию, ближайшую к ней в смысле определенной выше нормы:

$$Pv \doteq v_w, \text{ где } \mathbf{w} = \operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^d} \|v - v_w\|_\mu^2. \quad (11.12)$$

Таким образом, представимой функцией, ближайшей к истинной функции ценности v_π , является ее проекция Pv_π , как показано на рис. 11.3. Это то решение, которое асимптотически находят методы Монте-Карло, правда, зачастую очень медленно. Оператор проецирования более подробно обсуждается во врезке ниже.

TD-методы находят другие решения. Чтобы понять их подоплеку, вспомним, что уравнение Беллмана для функции ценности v_π имеет вид:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v_\pi(s')], \quad \text{для всех } s \in \mathcal{S}. \quad (11.16)$$

Матрица проекции

В случае линейного аппроксиматора функций операция проецирования линейна, поэтому ее можно представить матрицей размера $|\mathcal{S}| \times |\mathcal{S}|$:

$$\Pi \doteq \mathbf{X}(\mathbf{X}^T \mathbf{D} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{D}, \quad (11.13)$$

где, как и в разделе 9.4, $\mathbf{D}$ обозначает диагональную матрицу $|\mathcal{S}| \times |\mathcal{S}|$ с элементами $\mu(s)$ на диагонали, а $\mathbf{X}$ – матрицу $|\mathcal{S}| \times d$, строками которой являются векторы признаков $\mathbf{x}(s)^T$, по одному для каждого состояния s . Если обратной матрицы не существует, то вместо нее подставляется псевдообратная. В терминах этих матриц норму вектора можно записать в виде:

$$\|v\|_\mu^2 = v^T \mathbf{D} v, \quad (11.14)$$

а приближенную линейную функцию ценности – в виде:

$$v_w = \mathbf{X} w. \quad (11.15)$$

Истинная функция ценности v_π – единственная функция ценности, являющаяся точным решением уравнения (11.16). Если вместо v_π подставить приближенную функцию ценности v_w , то разность между правой и левой частями модифицированного уравнения можно использовать как меру удаленности v_w от v_π . Мы называем ее *беллмановской ошибкой* в состоянии s :

$$\bar{\delta}_w(s) \doteq \left(\sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v_w(s')] \right) - v_w(s) \quad (11.17)$$

$$= \mathbb{E}_\pi [R_{t+1} + \gamma v_w(S_{t+1}) - v_w(S_t) | S_t = s, A_t \sim \pi]. \quad (11.18)$$

Это уравнение ясно показывает связь между беллмановской ошибкой и TD-ошибкой (11.3). Беллмановская ошибка – это математическое ожидание TD-ошибки.

Вектор беллмановских ошибок во всех состояниях $\bar{\delta}_w \in \mathbb{R}^{|\mathcal{S}|}$ называется *беллмановским вектором ошибок* (обозначен ВЕ на рис. 11.3). Длина этого вектора в смысле ранее определенной нормы является мерой полной ошибки аппроксимации функции ценности и называется *среднеквадратической беллмановской ошибкой*:

$$\overline{\text{BE}}(w) = \|\bar{\delta}_w\|_\mu^2. \quad (11.19)$$

Вообще говоря, невозможно уменьшить $\overline{\text{BE}}$ до нуля (в этой точке было бы $v_w = v_\pi$), но в случае линейной аппроксимации существует единственное значение w , в котором $\overline{\text{BE}}$ достигает минимума. Эта точка в подпространстве представимых функций (обозначена $\min \overline{\text{BE}}$ на рис. 11.3), вообще говоря, отлична от той, в кото-

рой достигает минимума $\overline{VE}$ (обозначена Πv_π). Методы минимизации $\overline{VE}$ обсуждаются в следующих двух разделах.

Беллмановский вектор ошибок показан на рис. 11.3 как результат применения оператора Беллмана $B_\pi: \mathbb{R}^{|\mathcal{S}|} \rightarrow \mathbb{R}^{|\mathcal{S}|}$ к приближенной функции ценности. Оператор Беллмана определяется следующим образом:

$$(B_\pi v)(s) \doteq \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v(s')] \quad (11.20)$$

для всех $s \in \mathcal{S}$ и $v: \mathcal{S} \rightarrow \mathbb{R}$. Беллмановский вектор ошибок для v можно записать в виде: $\bar{\delta}_w = B_\pi v_w - v_w$.

Применение оператора Беллмана к функции ценности в представимом подпространстве в общем случае дает функцию, не принадлежащую этому подпространству, как следует из рисунка. В динамическом программировании (без аппроксимации функций) этот оператор повторно применяется к точкам вне представимого подпространства, как показывают серые стрелки в верхней части рис. 11.3. В конечном итоге этот процесс сходится к истинной функции ценности v_π , единственной неподвижной точке оператора Беллмана, в которой

$$v_\pi = B_\pi v_\pi, \quad (11.21)$$

и это еще один способ записать уравнение Беллмана для π (11.16).

Но если аппроксимация функций присутствует, то промежуточные функции ценности, не принадлежащие подпространству, представить невозможно. Следовать по серым стрелкам в верхней части рис. 11.3 нельзя, потому что после первого обновления (черная линия) функцию ценности необходимо снова спроецировать в какую-нибудь представимую. Затем следующая итерация начинается внутри подпространства; оператор Беллмана снова выходит за пределы подпространства, и результат опять проецируется назад, как показывают серые стрелка и линия в нижней части рисунка. Следование этим стрелкам – ДП-подобный процесс при наличии аппроксимации.

В этом случае нас интересует проекция беллмановского вектора ошибок обратно на представимое подпространство. Этот спроецированный беллмановский вектор ошибок $\Pi \bar{\delta}_w$ обозначен на рис. 11.3 как РВЕ. Его длина по определенной выше норме – еще одна мера ошибки приближенной функции ценности. Для любой приближенной функции ценности v мы определим *среднеквадратическую спроецированную беллмановскую ошибку* $\overline{PBE}$ следующим образом:

$$\overline{PBE}(w) = \|\Pi \bar{\delta}_w\|_\mu^2. \quad (11.22)$$

В случае линейной аппроксимации всегда существует приближенная функция ценности (в представимом подпространстве) с нулевой РВЕ; это неподвижная точка TD, w_{TD} , определенная в разделе 9.4. Как мы видели, эта точка не всегда устойчива относительно полугradientных TD-методов и обучения с разделенной стратегией. На рисунке показано, что эта функция ценности, вообще говоря, отличается от тех, что доставляют минимум ошибкам $\overline{VE}$ или $\overline{BE}$. В разделах 11.7 и 11.8 обсуждаются методы, которые гарантированно сходятся к ней.

11.5. ГРАДИЕНТНЫЙ СПУСК ПО БЕЛЛМАНОВСКОЙ ОШИБКЕ

Лучше понимая, как устроена аппроксимация функции ценности и различные применяемые целевые функции, мы теперь вернемся к проблеме устойчивости в обучении с разделенной стратегией. Мы хотели бы применить стохастический градиентный спуск (СГС, раздел 9.3), при котором обновления производятся так, что математическое ожидание равно антиградиенту целевой функции. Эти методы всегда спускаются по графику целевой функции (в смысле математического ожидания) и потому обычно устойчивы и демонстрируют отличные свойства сходимости. Из алгоритмов, уже рассмотренных в этой книге, только методы Монте-Карло являются настоящими СГС-методами. Они устойчиво сходятся при обучении с единой и разделенной стратегией, а также для нелинейных (дифференцируемых) аппроксиматоров функций общего вида, хотя зачастую медленнее, чем полуградиентные методы с бутстрэппингом, не являющиеся СГС-методами. Но, как мы видели выше в этой главе, полуградиентные методы могут расходиться при обучении с разделенной стратегией и при аномальной нелинейной аппроксимации (Tsitsiklis and Van Roy, 1997). Для настоящего СГС-метода такая расходимость невозможна.

Притягательность СГС настолько велика, что было предпринято много усилий, чтобы найти практический способ его применения к обучению с подкреплением. Отправной точкой всех таких попыток является выбор ошибки или целевой функции, подлежащей оптимизации. В этом и следующем разделах мы рассмотрим истоки и ограничения самой популярной целевой функции, которая основана на *беллмановской* ошибке, введенной в предыдущем разделе. Хотя на протяжении некоторого времени это был широко распространенный и авторитетный подход, мы придем к выводу, что это был шаг в неверном направлении, который не дает хороших алгоритмов обучения. С другой стороны, причина провала данного подхода сама по себе интересна и подсказывает, какой подход мог бы оказаться более подходящим.

Для начала рассмотрим не беллмановскую ошибку, а что-нибудь попроще. Обучение на основе временных различий построено на основе TD-ошибки. Почему бы не выбрать в качестве целевой функции минимизацию математического ожидания квадрата TD-ошибки? В случае аппроксимации общего вида одношаговая TD-ошибка с обесцениванием равна

$$\delta_t = R_{t+1} + \gamma \hat{V}(S_{t+1}, \mathbf{w}_t) - \gamma \hat{V}(S_t, \mathbf{w}_t).$$

Тогда потенциальную целевую функцию можно было бы назвать *среднеквадратической TD-ошибкой*:

$$\begin{aligned} \overline{\text{TDE}}(\mathbf{w}) &= \sum_{s \in S} \mu(s) \mathbb{E}[\delta_t^2 | S_t = s, A_t \sim \pi] \\ &= \sum_{s \in S} \mu(s) \mathbb{E}[\rho_t \delta_t^2 | S_t = s, A_t \sim b] \\ &= \mathbb{E}_b[\rho_t \delta_t^2]. \end{aligned} \quad (\text{если } \mu - \text{распределение при следовании } b)$$

Последнее уравнение имеет вид, необходимый для применения СГС; оно выражает целевую функцию в виде математического ожидания, которое можно вы-

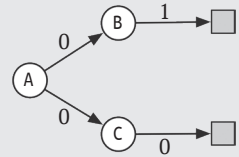
бирать из опыта (напомним, что опыт определяется поведенческой стратегией b). Таким образом, следуя стандартному подходу СГС, можно вывести обновление на одном шаге на основе выборки из этой ожидаемой ценности:

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t - \frac{1}{2} \alpha \nabla (\rho_t \delta_t^2) \\ &= \mathbf{w}_t - \alpha \rho_t \delta_t \nabla \delta_t \\ &= \mathbf{w}_t + \alpha \rho_t \delta_t (\nabla \hat{v}(S_t, \mathbf{w}_t) - \gamma \nabla \hat{v}(S_{t+1}, \mathbf{w}_t)). \end{aligned} \quad (11.23)$$

В этой формуле вы легко узнаете все тот же полуградиентный TD-алгоритм (11.2) с дополнительным последним членом. Этот член завершает градиент, в результате чего получается настоящий СГС-алгоритм с отличными гарантиями сходимости. Назовем этот алгоритм *наивным алгоритмом с остаточным градиентом* (вслед за Baird, 1995). Этот алгоритм сходится устойчиво, но необязательно к нужному месту.

Пример 11.2. Расщепление состояния А, иллюстрирующее наивность наивного алгоритма с остаточным градиентом

Рассмотрим эпизодический марковский процесс вознаграждения на рисунке *справа*. Эпизоды начинаются в состоянии А, а затем стохастически «расщепляются»: в половине случаев переходят в В (а затем детерминированно переходят в заключительное состояние с вознаграждением 1), а в половине – в С (и детерминированно завершаются с вознаграждением 0). Вознаграждение за первый переход, из состояния А, всегда равно 0, по какому бы пути ни развивался эпизод. Поскольку это эпизодическая задача, можно положить $\gamma = 1$. Предполагается также обучение с единой стратегией, так что ρ_t всегда равно 1, и табличная аппроксимация функций, так что алгоритмы обучения вправе назначить произвольные независимые ценности все трех состояний. Таким образом, задача обещает быть легкой.



Какими должны быть ценности? Доход в состоянии А в половине случаев равен 1, а в половине – 0; ценность А должна быть равна $\frac{1}{2}$. Доход в состоянии В всегда равен 1, поэтому его ценность должна быть равна 1. Аналогично доход в С всегда равен 0, поэтому его ценность должна быть равна 0. Это истинные ценности, и, поскольку это табличная задача, все ранее представленные методы должны сходиться к ним.

Однако наивный алгоритм с остаточным градиентом находит другие ценности В и С. Он сходится к ценности В, равной $\frac{3}{4}$, и к ценности С, равной $\frac{1}{4}$ (предельная ценность А равна $\frac{1}{2}$, что правильно). На самом деле это ценности, минимизирующие $\overline{TDE}$.

Вычислим $\overline{TDE}$ для этих ценностей. Первый переход в каждом эпизоде ведет либо вверх из А с ценностью $\frac{1}{2}$ в В с ценностью $\frac{3}{4}$ – изменение на $\frac{1}{4}$, либо вниз из А с ценностью $\frac{1}{2}$ в С с ценностью $\frac{1}{4}$ – изменение на $-\frac{1}{4}$. Поскольку вознаграждение на этих переходах равно 0 и $\gamma = 1$, эти изменения являются TD-ошибками, а значит, квадратичная TD-ошибка на первом переходе всегда равна $1/16$. Второй переход аналогичен: либо верхний из В с ценностью $\frac{3}{4}$ в заключительное состояние ценности 0 с вознаграждением 1, либо нижний из С с ценностью $\frac{1}{4}$ снова в заключительное состояние ценности 0 с вознаграждением 0. Таким образом, на втором шаге TD-ошибка всегда равна $\pm\frac{1}{4}$, а ее квадрат равен $1/16$. Итак, для этого набора ценностей $\overline{TDE}$ на обоих шагах равна $1/16$.

Теперь вычислим $\overline{TDE}$ для истинных ценностей (для В – 1, для С – 0, для А – $\frac{1}{2}$). Первый переход производится из состояния ценности $\frac{1}{2}$ либо в В ценности 1, либо в С ценности 0; в любом случае абсолютная величина ошибки равна $\frac{1}{2}$, а ее квадрат – $\frac{1}{4}$. Для

второго перехода ошибка равна 0, потому что начальная ценность – 1 или 0 в зависимости от того, производится переход из В или из С, всегда в точности равна непосредственному вознаграждению и доходу. Таким образом, квадрат TD-ошибки равен $\frac{1}{4}$ на первом переходе и 0 на втором, а среднее вознаграждение за оба перехода равно $\frac{1}{8}$. Поскольку $\frac{1}{8}$ больше $\frac{1}{16}$, это решение хуже с точки зрения TDE. В этой простой задаче у истинных ценностей TDE не наименьшая.

В примере с расщеплением состояния А используется табличное представление, так что истинные ценности можно представить точно, и тем не менее наивный алгоритм с остаточным градиентом находит другие значения ценностей, у которых TDE меньше, чем у истинных. Минимизация TDE производится наивно; из-за штрафования за все TD-ошибки достигается скорее сглаживание по времени, чем точное предсказание.

Похоже, правильнее было бы минимизировать беллмановскую ошибку. Если в результате обучения найдены точные значения, то беллмановская ошибка будет всюду равна нулю. Таким образом, алгоритм, минимизирующий беллмановскую ошибку, не должен столкнуться с проблемами в примере с расщеплением А. В общем случае нельзя ожидать получения нулевой беллмановской ошибки, поскольку для этого нужно было бы найти истинную функцию ценности, которая предположительно не принадлежит подпространству представимых функций ценности. Но подобраться ближе к идеалу – цель, которая кажется естественной. Как мы видели, беллмановская ошибка также тесно связана с TD-ошибкой. Беллмановская ошибка для некоторого состояния – это математическое ожидание TD-ошибки в данном состоянии. Поэтому повторим приведенный выше вывод, подставив математическое ожидание TD-ошибки (здесь все математические ожидания неявно обусловлены S_t):

$$\begin{aligned}
 \mathbf{w}_{t+1} &= \mathbf{w}_t - \frac{1}{2} \alpha \nabla (\mathbb{E}_\pi [\delta_t]) \\
 &= \mathbf{w}_t - \frac{1}{2} \alpha \nabla (\mathbb{E}_b [\rho_t \delta_t]^2) \\
 &= \mathbf{w}_t - \alpha \mathbb{E}_b [\rho_t \delta_t] \nabla \mathbb{E}_b [\rho_t \delta_t] \\
 &= \mathbf{w}_t - \alpha \mathbb{E}_b [\rho_t (R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w}))] \mathbb{E}_b [\rho_t \nabla \delta_t] \\
 &= \mathbf{w}_t + \alpha [\mathbb{E}_b [\rho_t (R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}))] - \hat{v}(S_t, \mathbf{w})] [\nabla \hat{v}(S_t, \mathbf{w}) - \gamma \mathbb{E}_b [\rho_t \nabla \hat{v}(S_{t+1}, \mathbf{w})]].
 \end{aligned}$$

Это обновление и различные способы выборки из него называются *алгоритмом с остаточным градиентом*. Если бы мы просто использовали выборочные ценности во всех математических ожиданиях, то это уравнение почти точно сводится к (11.23), наивному алгоритму с остаточным градиентом¹. Но это наивно, потому что операция выше включает следующее состояние, S_{t+1} , которое встречается в двух перемножаемых математических ожиданиях. Чтобы получить несмещенную выборку из произведения, необходимы два независимых образца следующе-

¹ Для ценностей состояний остается небольшое различие в обработке коэффициента выборки по значимости ρ_r . В аналогичном случае ценности действий (наиболее важно для алгоритмов управления) алгоритм с остаточным градиентом точно сводился бы к наивной версии.

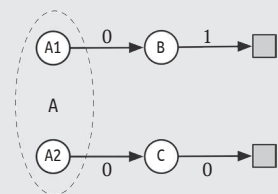
го состояния, но в процессе нормального взаимодействия с окружающей средой можно получить только один. Мы можем произвести выборку либо из одного, либо из другого математического ожидания, но не из обоих сразу.

Заставить алгоритм с остаточным градиентом работать можно двумя способами. Один применим в случае детерминированной окружающей среды. Если переход в следующее состояние детерминирован, то обе выборки по необходимости будут одинаковыми, и наивный алгоритм корректен. Другой способ – получить две независимые выборки состояния S_{t+1} , следующего за S_t , одну для первого математического ожидания, вторую для второго. При реальном взаимодействии с окружающей средой это невозможно, но при взаимодействии с имитированной средой – пожалуйста. Нужно просто откатиться к предыдущему состоянию и получить следующее, перед тем как продолжать работу с первым следующим состоянием. В обоих случаях алгоритм с остаточным градиентом гарантированно сходится к минимуму $\overline{BE}$ при обычных условиях на размер шага. Как и в настоящем СГС-методе, сходимость устойчивая как для линейных, так и для нелинейных аппроксиматоров. В линейном случае алгоритм сходится к единственному w , представляющему минимум $\overline{BE}$.

Однако остается по меньшей мере три причины, по которым сходимость алгоритма с остаточным градиентом может быть неудовлетворительна. Первая – то, что эмпирически она очень медленная, гораздо медленнее, чем для полугradientных методов. На самом деле сторонники этого метода предложили повысить его скорость, начав с какого-нибудь полугradientного метода, а затем постепенно переключаться на остаточный градиент, чтобы гарантировать сходимость (Baird and Moore, 1999). Вторая причина – тот факт, что алгоритм, похоже, сходится к неверным ценностям. Он действительно получает правильные ценности во всех табличных случаях, таких как пример с расщеплением A, поскольку для них уравнение Беллмана можно решить точно. Но если рассмотреть примеры с настоящей аппроксимацией функций, то окажется, что алгоритм с остаточным градиентом, да и сама целевая функция $\overline{BE}$, похоже, находят неправильные функции ценности. Один из самых убедительных примеров такого рода – вариация на тему расщепления A, известная как *предрасщепленное A*, показан ниже; в этом случае алгоритм с остаточным градиентом находит то же самое плохое решение, что и его наивная версия. Это пример демонстрирует, что минимизация $\overline{BE}$ (с которой алгоритм с остаточным градиентом, безусловно, справляется) – возможно, неподходящая цель.

Пример 11.3. Предрасщепленное A, контрпример к минимизации $\overline{BE}$

Рассмотрим эпизодический процесс марковского вознаграждения с тремя состояниями, показанный на рисунке справа. Эпизоды начинаются в состоянии A1 или A2 с одинаковой вероятностью. Для аппроксиматора оба состояния выглядят совершенно одинаково, как одно состояние A, представление которого с помощью признаков отличается и никак не связано с представлениями двух других состояний B и C, которые, в свою очередь, отличаются друг от друга. Точнее, вектор параметров аппроксиматора содержит три элемента: один дает ценность состояния B, другой – ценность состояния C, а третий – ценность обоих состояний A1 и A2. Если



не считать выбора начального состояния, система детерминированная. Если процесс начинается в состоянии A1, то переходит в состояние В с вознаграждением 0, а затем завершается с вознаграждением 1. Если же он начинается в состоянии A2, то переходит в состояние С и затем завершается, получая на обоих переходах вознаграждение 0.

Для алгоритма обучения, который видит только признаки, система выглядит точно так же, как в примере с расщеплением А. Он думает, что система всегда стартует в состоянии А, за которым с равной вероятностью следуют состояния В или С, после чего завершается с вознаграждением 1 или 0, которое детерминированно зависит от предыдущего состояния. Как и в примере с расщеплением А, истинные ценности В и С равны 1 и 0, а наилучшая общая ценность A1 и A2 равна $\frac{1}{2}$ в силу симметрии.

Поскольку эта задача внешне выглядит идентичной примеру с расщеплением А, мы уже знаем, какие ценности найдут алгоритмы. Полугradientный TD сходится к вышеупомянутым идеальным ценностям, а наивный алгоритм с остаточным градиентом – к ценностям $\frac{3}{4}$ и $\frac{1}{4}$ для В и С соответственно. Все переходы состояний детерминированны, поэтому ненаивный алгоритм с остаточным градиентом также сходится к этим ценностям (в данном случае это один и тот же алгоритм). Отсюда следует, что «наивное» решение также должно минимизировать $\overline{BE}$, и так оно и есть. В детерминированной задаче беллмановские и TD-ошибки совпадают, поэтому $\overline{BE}$ всегда то же самое, что $\overline{TDE}$. Оптимизация $\overline{BE}$ в этом примере заканчивается так же печально, как в случае наивного алгоритма с остаточным градиентом в примере с расщеплением А.

Третья причина, по которой сходимость алгоритма с остаточным градиентом неудовлетворительна, объясняется в следующем разделе. Как и вторая, она связана с самой целевой функцией $\overline{BE}$, а не с каким-то конкретным алгоритмом ее минимизации.

11.6. БЕЛЛМАНОВСКАЯ ОШИБКА НЕОБУЧАЕМА

Понятие обучаемости, которое мы введем в этом разделе, отличается от того, что традиционно используется в машинном обучении. Там говорят, что гипотеза «обучаемая», если она *эффективно* обучаемая, т. е. может быть обучена на полиномиальном, а не экспоненциальном количестве примеров. Здесь термин имеет более прямолинейный смысл и означает возможность обучения вообще, вне зависимости от объема опыта. Оказывается, что многие величины, представляющие очевидный интерес в обучении с подкреплением, невозможно обучить даже на бесконечном объеме опытных данных. Эти величины корректно определены, их можно вычислить, располагая информацией о внутренней структуре окружающей среды, но нельзя ни вычислить, ни оценить по наблюдаемой последовательности векторов признаков, действий и вознаграждений¹. Мы говорим, что они *не-*

¹ Их, конечно, можно было бы оценить, если бы наблюдалась последовательность состояний, а не только соответствующих им векторов признаков.

обучаемы. Оказывается, что целевая функция, равная беллмановской ошибке ($\overline{VE}$), рассмотренной в двух предыдущих разделах, в этом смысле необучаема. И тот факт, что беллмановскую ошибку невозможно обучить на наблюдаемых данных, – быть может, самый сильный аргумент против нее.

Чтобы прояснить концепцию обучаемости, начнем с простых примеров. Рассмотрим два марковских процесса вознаграждения (МПВ)¹, показанных на рисунке ниже.



Там, где из состояния выходят два ребра, предполагается, что оба перехода происходят с одинаковой вероятностью, а числа обозначают полученное вознаграждение. Все состояния выглядят одинаково; каждому соответствует один и тот же одномерный вектор признаков $x = 1$, и у всех приближенная ценность равна w . Таким образом, единственная варьирующаяся часть траектории данных – последовательность вознаграждений. Левый МПВ остается в одном состоянии и порождает бесконечный случайный поток чисел 0 и 2, каждое с вероятностью 0.5. Правый МПВ на каждом шаге либо остается в текущем состоянии, либо переходит в другое – с одинаковой вероятностью. Вознаграждение в этом МПВ детерминированное: при переходе из одного состояния равно 0, при переходе из другого – 2, но поскольку сами состояния на каждом этапе равновероятны, то наблюдается такой же бесконечный случайный поток чисел 0 и 2, как в левом МПВ. (Мы можем предполагать, что правый МПВ стартует в одном из двух состояний с равной вероятностью.) Таким образом, даже при наличии бесконечного объема данных невозможно сказать, какой из двух МПВ их сгенерировал. В частности, мы не можем сказать, имеет ли МПВ одно или два состояния, стохастический он или детерминированный. Узнать эти вещи в процессе обучения не представляется возможным.

Эта пара МПВ доказывает также, что целевая функция $\overline{VE}$ (9.1) необучаема. Если $\gamma = 0$, то истинные ценности трех состояний (в обоих МПВ) слева направо равны 1, 0 и 2. Предположим, что $w = 1$. Тогда $\overline{VE}$ равно 0 для левого МПВ и 1 для правого МПВ. Поскольку $\overline{VE}$ в двух задачах различна, а сгенерированные данные тем не менее имеют одинаковое распределение, то $\overline{VE}$ не может быть обучена. $\overline{VE}$ – неоднозначно определенная функция распределения данных. А раз ее нельзя обучить, то как она может быть полезна в качестве целевой функции обучения?

Если целевую функцию нельзя обучить, то ее полезность и вправду оказывается под вопросом. Однако в случае $\overline{VE}$ выход есть. Заметим, что одно и то же

¹ Все МПВ можно рассматривать как МППР с одним действием в каждом состоянии; все выводы, относящиеся к МПВ, применимы и к МППР.

решение, $w = 1$, оптимально для обоих вышеупомянутых МПВ (в предположении, что μ одинаково для обоих неразличимых состояний правого МПВ). Это случайное совпадение или действительно для всех МППР с одинаковым распределением данных оптимальные векторы параметров тоже одинаковы? Если это так – а далее мы покажем, что так и есть, – то $\overline{VE}$ остается полезной целевой функцией. $\overline{VE}$ необучаема, но можно обучить параметр, который ее оптимизирует!

Чтобы понять, в чем тут дело, полезно ввести в рассмотрение еще одну естественную целевую функцию, на этот раз, несомненно, обучаемую. Ошибка, которую всегда можно наблюдать, – это разность между оценкой в каждый момент времени и доходом, полученным после этого момента. Математическое ожидание квадрата данной ошибки при распределении μ называется *ошибкой среднеквадратического дохода* (Mean Square Return Error) и обозначается $\overline{RE}$. В случае единой стратегии $\overline{RE}$ можно записать в виде:

$$\begin{aligned}\overline{RE}(\mathbf{w}) &= \mathbb{E}[(G_t - \hat{v}(S_t, \mathbf{w}))^2] \\ &= \overline{VE}(\mathbf{w}) + \mathbb{E}[(G_t - v_\pi(S_t))^2].\end{aligned}\quad (11.24)$$

Таким образом, обе целевые функции одинаковы, за исключением члена дисперсии, который не зависит от вектора параметров. Поэтому оптимальный вектор параметров $\mathbf{w}^*$ для обеих должен быть одним и тем же. Связи между различными целевыми функциями показаны в левой части рис. 11.4.

**Упражнение 11.4.* Докажите тождество (11.24). Указание: запишите $\overline{RE}$ как математическое ожидание всех возможных состояний s математического ожидания квадрата ошибки при условии $S_t = s$. Затем прибавьте и вычтите истинную ценность s из ошибки (до возведения в квадрат), сгруппируйте вычтенную истинную ценность с доходом, а прибавленную истинную ценность с оценкой ценности. Тогда если раскрыть скобки, то самый сложный член обратится в ноль, а останется (11.24). $\square$

Но вернемся к $\overline{BE}$. $\overline{BE}$ похоже на $\overline{VE}$ тем, что может быть вычислена на основе информации о МППР, но не допускает обучения на данных. Однако она отличается от $\overline{VE}$ тем, что параметр, доставляющий ей минимум, необучаем. Во врезке ниже приведен контрпример – два МПВ, которые порождают одно и то же распределение данных, но при этом минимизирующие их параметры различны. Это доказывает, что оптимальный вектор параметров не является функцией данных и потому не может быть обучен на них. Другие рассмотренные нами бутстрэппинговые целевые функции, $\overline{PBE}$ и $\overline{TDE}$, могут быть найдены по данным (и обучены) и определяют оптимальные решения, в общем случае отличные друг от друга и от решения, доставляющего минимум $\overline{BE}$. Общий случай показан в правой части рис. 11.4.

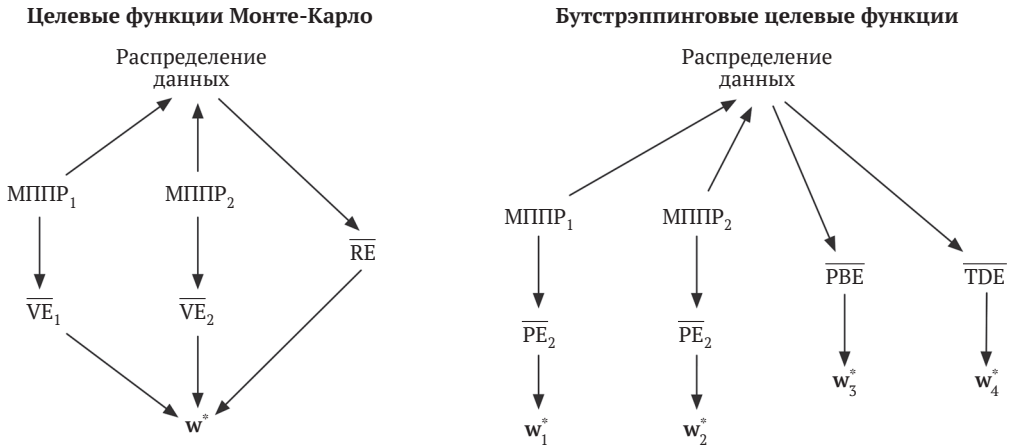
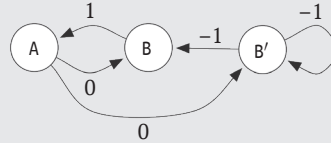
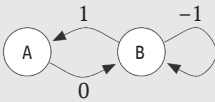


Рис. 11.4 ❖ Причинно-следственные связи между распределением данных, МППР и различными целевыми функциями. **Слева – целевые функции Монте-Карло:** два разных МППР могут порождать одно и то же распределение данных, но при этом иметь разные ошибки $\overline{VE}$, это доказывает, что целевую функцию $\overline{VE}$ нельзя определить по данным, а стало быть, она необучаема. Однако у всех таких $\overline{VE}$ должен быть один и тот же оптимальный вектор параметров w^* ! Более того, этот же вектор w^* можно найти с помощью другой целевой функции $\overline{RE}$, которая однозначно определяется по распределению данных. Таким образом, w^* и $\overline{RE}$ обучаемы, хотя $\overline{VE}$ – нет. **Справа – бутстрэппинговые целевые функции:** два разных МППР могут порождать одно и то же распределение данных и иметь разные минимизирующие векторы параметров; они необучаемы по распределению данных. Целевые функции $\overline{PBE}$ и $\overline{TDE}$ и их (различные) минимумы можно определить непосредственно по данным, т. е. они обучаемы

Таким образом, $\overline{BE}$ необучаема; ее нельзя оценить по векторам признаков и другим наблюдаемым данным. Это ограничивает применение $\overline{BE}$ задачами на основе модели. Не может существовать алгоритма, который минимизирует $\overline{BE}$ без доступа к информации о состояниях МППР сверх той, что дают векторы признаков. Алгоритм с остаточным градиентом способен минимизировать $\overline{BE}$ только потому, что ему разрешено дважды производить выборку из одного и того же состояния – не состояния с таким же вектором признаков, а истинного состояния процесса. И теперь мы видим, что обойти это никак нельзя. Минимизация $\overline{BE}$ требует такого рода доступа к номинальному истинному МППР. Это важное ограничение на $\overline{BE}$, помимо того, который мы выявили в примере с расщепленным состоянием A на стр. 321. Все это заставляет нас обратить больше внимания на $\overline{BE}$.

Пример 11.4. Контпример, доказывающий необучаемость беллмановской ошибки

Чтобы продемонстрировать полный спектр возможностей, нам понадобится чуть более сложная пара марковских процессов вознаграждения (VGD), чем рассмотренные выше.



Если из состояния выходят два ребра, то предполагается, что вероятности обоих переходов одинаковы, а числа обозначают начисленное вознаграждение. Левый МПВ имеет два состояния, представленных по отдельности. В правом МПВ три состояния, два из которых – B и B' – выглядят одинаково и должны иметь одинаковую приближенную ценность. Точнее, w содержит два элемента, и ценность состояния A задается первым из них, а ценность B и B' – вторым. Второй МПВ устроен так, что время, проведенное во всех трех состояниях, одинаково, так что можно положить $\mu(s) = 1/3$ для всех s.

Заметим, что наблюдаемое распределение данных одинаково для обоих МПВ. В обоих случаях агент будет видеть одно появление A, за которым следует 0, затем сколько-то B, за каждым из которых, кроме последнего, следует -1, а за последним -1, а затем все начнется сначала: одно A и 0 и т. д. Все статистические детали также одинаковы; в обоих МПВ вероятность цепочки из k вхождений B равна 2^{-k} .

Теперь предположим, что $w = 0$. В первом МПВ это точное решение, и $\overline{BE}$ равно нулю. Во втором МПВ это решение порождает для B и B' квадрат ошибки 1, так что $\overline{BE} = \mu(B)1 + \mu(B')1 = 2/3$. У этих двух МПВ, порождающих одно и то же распределение данных, $\overline{BE}$ различны, следовательно, $\overline{BE}$ необучаема.

Более того (в отличие от предыдущего примера с $\overline{BE}$), ценность, доставляющая минимум w, для этих МПВ различна. Для первого МПВ вектор $w = 0$ минимизирует $\overline{BE}$ для любого γ . Для второго МПВ минимизирующий w является сложной функцией от γ , но в пределе при $\gamma \rightarrow 1$ он равен $(-1/2, 0)^T$. Таким образом, решение, которое доставляет минимум $\overline{BE}$, нельзя оценить только на основе данных, необходима информация о МПВ сверх той, что содержится в данных. В этом смысле в принципе невозможно использовать $\overline{BE}$ в качестве целевой функции обучения.

Удивительно, что во втором МПВ ценность A, доставляющая минимум $\overline{BE}$, так далеко отстоит от нуля. Напомним, что A сопоставлен выделенный вес, и, стало быть, его ценность не ограничена аппроксимацией функции. За A следует вознаграждение 0 и переход в состояние с ценностью, почти равной 0, откуда следует, что $v_w(A)$ должно быть равно 0; так почему же его оптимальная ценность отрицательна, а не равна нулю? Ответ в том, что, делая $v_w(A)$ отрицательным, мы уменьшаем ошибку при переходе в A из B. Вознаграждение на этом детерминированном переходе равно 1, а значит, ценность B должна быть на 1 больше ценности A. Поскольку ценность B приблизительно равна 0, ценность A стремится к -1. Минимизирующая $\overline{BE}$ ценность A, приблизительно равная $-1/2$, – компромисс между уменьшением ошибки при входе в A и при выходе из A.

11.7. ГРАДИЕНТНЫЕ TD-МЕТОДЫ

Теперь мы рассмотрим СГС-методы для минимизации $\overline{\text{PBE}}$. Как и настоящие СГС-методы, эти градиентные TD-методы обладают свойствами устойчивой сходимости даже при обучении с разделенной стратегией и нелинейной аппроксимации функций. Напомним, что в линейном случае всегда существует точное решение, неподвижная точка TD, $\mathbf{w}_{\text{TD}}$, в которой $\overline{\text{PBE}}$ обращается в ноль. Это решение можно было бы найти методом наименьших квадратов (раздел 9.8), но только квадратов зависящих от количества параметров (сложность $O(d^2)$). А мы хотим найти СГС-метод, имеющий сложность $O(d)$ и обладающий свойствами устойчивой сходимости. Градиентные TD-методы приближаются к этой цели, но ценой удвоения вычислительной сложности. Чтобы построить СГС-метод для минимизации $\overline{\text{PBE}}$ (в предположении линейной аппроксимации), мы для начала перепишем целевую функцию (11.22) в матричном виде:

$$\begin{aligned}\overline{\text{PBE}}(\mathbf{w}) &= \|\Pi \bar{\delta}\|_{\mu}^2 \\ &= (\Pi \bar{\delta}_{\mathbf{w}})^T \mathbf{D} \Pi \bar{\delta}_{\mathbf{w}}^{-} \quad (\text{в силу 11.14})\end{aligned}$$

$$\begin{aligned}&= \bar{\delta}_{\mathbf{w}}^T \Pi^T \mathbf{D} \Pi \bar{\delta}_{\mathbf{w}} \\ &= \bar{\delta}_{\mathbf{w}}^T \mathbf{D} \mathbf{X} (\mathbf{X}^T \mathbf{D} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{D} \bar{\delta}_{\mathbf{w}}^{-} \quad (11.25)\end{aligned}$$

$$\begin{aligned}(\text{пользуясь (11.13) и тождеством } \Pi^T \mathbf{D} \Pi = \mathbf{D} \mathbf{X} (\mathbf{X}^T \mathbf{D} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{D}), \\ = (\mathbf{X}^T \mathbf{D} \bar{\delta}_{\mathbf{w}})^T (\mathbf{X}^T \mathbf{D} \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{D} \bar{\delta}_{\mathbf{w}}). \quad (11.26)\end{aligned}$$

Градиент по $\mathbf{w}$ равен

$$\nabla \overline{\text{PBE}}(\mathbf{w}) = 2 \nabla [\mathbf{X}^T \mathbf{D} \bar{\delta}_{\mathbf{w}}]^T (\mathbf{X}^T \mathbf{D} \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{D} \bar{\delta}_{\mathbf{w}}).$$

Чтобы преобразовать это в СГС-метод, необходимо производить на каждом шаге выборку, математическое ожидание которой равно этой величине. Пусть μ – распределение состояний, посещенных при следовании поведенческой стратегии. Тогда все три сомножителя в формуле выше можно записать в виде математических ожиданий при таком распределении. Например, последний сомножитель записывается в виде:

$$\mathbf{X}^T \mathbf{D} \bar{\delta}_{\mathbf{w}} = \sum_s \mu(s) \mathbf{x}(s) \bar{\delta}_{\mathbf{w}}(s) = \mathbb{E}[\rho_t \delta_t \mathbf{x}_t],$$

а это не что иное, как математическое ожидание обновления для полуградиентного TD(0) (11.2). Первый сомножитель – транспонированная матрица градиента этого обновления:

$$\begin{aligned}\nabla \mathbb{E}[\rho_t \delta_t \mathbf{x}_t]^T &= \mathbb{E}[\rho_t \nabla \delta_t^T \mathbf{x}_t^T] \\ &= \mathbb{E}[\rho_t \nabla (R_{t+1} + \gamma \mathbf{w}^T \mathbf{x}_{t+1} - \mathbf{w}^T \mathbf{x}_t)^T \mathbf{x}_t^T] \quad (\text{используя эпизодическое } \delta_t) \\ &= \mathbb{E}[\rho_t (\mathbf{x}_{t+1} - \mathbf{x}_t) \mathbf{x}_t^T].\end{aligned}$$

Наконец, средний сомножитель – обращение математического ожидания внешнего матричного произведения векторов признаков:

$$\mathbf{X}^T \mathbf{D} \mathbf{X} = \sum_s \mu(s) \mathbf{x}_s \mathbf{x}_s^T = \mathbb{E}[\mathbf{x}_t \mathbf{x}_t^T].$$

Подставляя эти математические ожидания вместо всех трех сомножителей в наше выражение для градиента $\overline{\text{PBE}}$, получаем:

$$\nabla \overline{\text{PBE}}(\mathbf{w}) = 2\mathbb{E}[\rho_t(\gamma \mathbf{x}_{t+1} - \mathbf{x}_t) \mathbf{x}_t^T] \mathbb{E}[\mathbf{x}_t \mathbf{x}_t^T]^{-1} \mathbb{E}[\rho_t \delta_t \mathbf{x}_t]. \quad (11.27)$$

На первый взгляд, не очевидно, зачем нужно было записывать градиент в такой форме. Это произведение трех выражений, из которых первое и последнее не являются независимыми. Оба они зависят от следующего вектора признаков $\mathbf{x}_{t+1}$; мы просто не можем произвести выборку из обоих математических ожиданий, а затем перемножить результаты. Это дало бы смещенную оценку градиента, как и в алгоритме с остаточным градиентом.

Другая идея – оценить все три математических ожидания по отдельности, а затем объединить их для получения несмещенной оценки градиента. Это возможно, но потребовало бы очень большого объема вычислений и памяти, в частности для хранения первых двух математических ожиданий, которые являются матрицами размера $d \times d$, и для обращения второй из них. Эту идею можно усовершенствовать. Если оценены и сохранены два из трех математических ожиданий, то из третьего можно было бы произвести выборку и использовать ее в сочетании с двумя сохраненными величинами. Например, можно было бы хранить оценки двух последних величин (применив технику инкрементного обращения-обновления из раздела 9.8), а затем произвести выборку из первого выражения. К сожалению, сложность всего алгоритма по-прежнему квадратичная (порядка $O(d^2)$).

Идея хранить какие-то оценки по отдельности, а затем объединять их с выборками хороша и нашла применение в градиентных TD-методах. В них оценивается и сохраняется произведение второго и третьего сомножителей в выражении (11.27). Эти сомножители представляют собой матрицу $d \times d$ и d -мерный вектор, поэтому их произведение – d -мерный вектор, как и сам $\mathbf{w}$. Обозначим этот второй обученный вектор $\mathbf{v}$:

$$\mathbf{v} \approx \mathbb{E}[\mathbf{x}_t \mathbf{x}_t^T]^{-1} \mathbb{E}[\rho_t \delta_t \mathbf{x}_t]. \quad (11.28)$$

Данная форма знакома тем, кто изучает линейные методы обучения с учителем. Это решение методом наименьших квадратов линейной задачи об аппроксимации $\rho_t \delta_t$ по признакам. Стандартный СГС-метод инкрементного нахождения вектора $\mathbf{v}$, минимизирующего математическое ожидание квадрата ошибки $(\mathbf{v}^T \mathbf{x}_t - \rho_t \delta_t)^2$, называется правилом наименьших средних квадратов (Least Mean Square – LMS) (здесь оно еще дополнено коэффициентом выборки по значимости):

$$\mathbf{v}_{t+1} \doteq \mathbf{v}_t + \beta \rho_t (\delta_t - \mathbf{v}_t^T \mathbf{x}_t) \mathbf{x}_t,$$

где $\beta > 0$ – еще один параметр размера шага. Мы можем использовать этот метод для эффективного нахождения оценки (11.28) с объемом потребляемой памяти и вычислений на каждом шаге $O(d)$.

Имея сохраненную оценку $\mathbf{v}_t$, аппроксимирующую (11.28), мы можем обновить главный вектор параметров $\mathbf{w}_t$ с помощью СГС-методов, основанных на формуле (11.27). Простейшее правило такого рода имеет вид:

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t - \frac{1}{2} \alpha \nabla \overline{\text{PBE}}(\mathbf{w}_t) && \text{(общее правило СГС)} \\ &= \mathbf{w}_t - \frac{1}{2} \alpha 2 \mathbb{E}[\rho_t (\gamma \mathbf{x}_{t+1} - \mathbf{x}_t) \mathbf{x}_t^\top] \mathbb{E}[\mathbf{x}_t \mathbf{x}_t^\top]^{-1} \mathbb{E}[\rho_t \delta_t \mathbf{x}_t] && \text{(в силу 11.27)} \\ &= \mathbf{w}_t + \alpha \mathbb{E}[\rho_t (\mathbf{x}_t - \gamma \mathbf{x}_{t+1}) \mathbf{x}_t^\top] \mathbb{E}[\mathbf{x}_t \mathbf{x}_t^\top]^{-1} \mathbb{E}[\rho_t \delta_t \mathbf{x}_t] && (11.29) \\ &\approx \mathbf{w}_t + \alpha \mathbb{E}[\rho_t (\mathbf{x}_t - \gamma \mathbf{x}_{t+1}) \mathbf{x}_t^\top] \mathbf{v}_t && \text{(на основе 11.28)} \\ &\approx \mathbf{w}_t + \alpha \rho_t (\mathbf{x}_t - \gamma \mathbf{x}_{t+1}) \mathbf{x}_t^\top \mathbf{v}_t. && \text{(выборка)}\end{aligned}$$

Этот алгоритм называется *GTD2*. Заметим, что если последнее скалярное произведение $(\mathbf{x}_t^\top \mathbf{v}_t)$ вычисляется первым, то сложность всего алгоритма составляет $O(d)$.

Чуть лучший алгоритм получается, если выполнить больше аналитических шагов до подстановки в $\mathbf{v}_t$. Продолжим формулу (11.29):

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha \mathbb{E}[\rho_t (\mathbf{x}_t - \gamma \mathbf{x}_{t+1}) \mathbf{x}_t^\top] \mathbb{E}[\mathbf{x}_t \mathbf{x}_t^\top]^{-1} \mathbb{E}[\rho_t \delta_t \mathbf{x}_t] \\ &= \mathbf{w}_t + \alpha (\mathbb{E}[\rho_t \mathbf{x}_t \mathbf{x}_t^\top] - \gamma \mathbb{E}[\rho_t \mathbf{x}_{t+1} \mathbf{x}_t^\top]) \mathbb{E}[\mathbf{x}_t \mathbf{x}_t^\top]^{-1} \mathbb{E}[\rho_t \delta_t \mathbf{x}_t] \\ &= \mathbf{w}_t + \alpha (\mathbb{E}[\mathbf{x}_t \rho_t \mathbf{x}_t^\top] - \gamma \mathbb{E}[\rho_t \mathbf{x}_{t+1} \mathbf{x}_t^\top]) \mathbb{E}[\mathbf{x}_t \mathbf{x}_t^\top]^{-1} \mathbb{E}[\rho_t \delta_t \mathbf{x}_t] \\ &\approx \mathbf{w}_t + \alpha (\mathbb{E}[\mathbf{x}_t \rho_t \delta_t] - \gamma \mathbb{E}[\rho_t \mathbf{x}_{t+1} \mathbf{x}_t^\top] \mathbf{v}_t) && \text{(на основе 11.28)} \\ &\approx \mathbf{w}_t + \alpha \rho_t (\delta_t \mathbf{x}_t - \gamma \mathbf{x}_{t+1} \mathbf{x}_t^\top \mathbf{v}_t), && \text{(выборка)}\end{aligned}$$

и сложность снова составляет $O(d)$, если последнее произведение $(\mathbf{x}_t^\top \mathbf{v}_t)$ вычисляется первым. Этот алгоритм называется либо *TD(0) с градиентной поправкой* (*TD(0) with gradient correction – TDC*), либо *GTD(0)*.

На рис. 11.5 показано выборочное и ожидаемое поведения TDC для контпримера Бэрда. Как и предполагалось, $\overline{\text{PBE}}$ падает до нуля, но заметим, что отдельные элементы вектора параметров не стремятся к нулю. На самом деле эти ценности по-прежнему далеки от оптимального решения $\hat{\mathbf{v}}(s) = 0$ для тех s , для которых $\mathbf{w}$ должен был бы быть пропорционален $(1, 1, 1, 1, 1, 4, -2)^\top$. После 1000 итераций мы все еще далеки от оптимального решения, что показывает ошибка $\overline{\text{VE}}$, остающаяся равной 3. Система все-таки сходится к оптимальному решению, но очень медленно, потому что $\overline{\text{PBE}}$ уже близка к нулю.

И GTD2, и TDC включают два процесса обучения: основной для $\mathbf{w}$ и вспомогательный для $\mathbf{v}$. Логика основного процесса обучения опирается на завершение вспомогательного, хотя бы приближенное, тогда как на протекание вспомогательного процесса обучения на основной никак не влияет. Такого рода асимметричную зависимость мы называем *каскадом*. В каскадах часто предполагается, что вспомогательный процесс обучения протекает быстрее и поэтому всегда достигает своего асимптотического значения, будучи готов помочь основному процессу. В доказательствах сходимости этих методов данное предположение часто делается явно. В таком случае мы имеем *доказательства с двумя временными шкалами*. Со вспомогательным процессом обучения связана быстрая временная шкала,

а с основным – более медленная. Если α – размер шага в основном процессе обучения, а β – во вспомогательном, то в этих доказательствах сходимости обычно требуется, чтобы $\beta \rightarrow 0$ и $\alpha/\beta \rightarrow 0$.

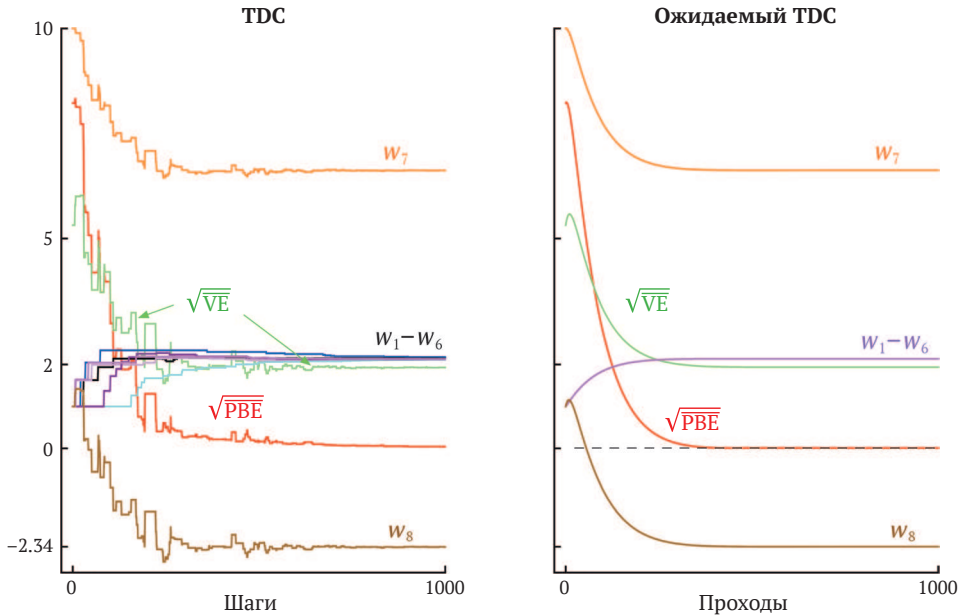


Рис. 11.5 ❖ Поведение алгоритма TDC для контрпримера Бэрда. Слева показан один типичный прогон, справа – ожидаемое поведение этого алгоритма, если обновления производятся синхронно (по аналогии с (11.9), за исключением обоих векторов параметров TDC). Размеры шагов были равны $\alpha = 0.005, \beta = 0.05$

Градиентные TD-методы – в настоящее время наиболее изученные и широко применяемые устойчивые методы обучения с разделенной стратегией. Существуют их варианты для вычисления ценности действий и управления (GQ, Maei et al., 2010), для следов приемлемости (GTD(λ) и GQ(λ), Maei, 2011; Maei and Sutton, 2010) и для нелинейной аппроксимации функций (Maei et al., 2009). Предложены также гибридные алгоритмы, занимающие место между полуградиентными и градиентными TD-методами (Hackman, 2012; White and White, 2016). Гибридные TD-алгоритмы ведут себя как градиентные TD-алгоритмы в тех состояниях, где целевая и поведенческая стратегии сильно различаются, и, как полуградиентные алгоритмы там, эти стратегии совпадают. Наконец, в результате объединения идеи градиентных TD-методов с идеями проксимальных методов и переменным управлением были созданы более эффективные методы (Mahadevan et al., 2014; Du et al., 2017).

11.8. ЭМФАТИЧЕСКИЕ TD-МЕТОДЫ

Теперь обратимся ко второму важному подходу, который всесторонне исследовался в надежде получить дешевый и эффективный метод обучения с разделенной стратегией и аппроксимацией функций. Напомним, что линейные

полуградиентные TD-методы эффективны и устойчиво сходятся при обучении с распределением с единой стратегией и что в разделе 9.4 было показано, что это напрямую связано с положительной определенностью матрицы A (9.11)¹ и соответствием между распределением состояний при единой стратегии μ_π с одной стороны, и вероятностями переходов состояний $p(s|s, a)$ при следовании целевой стратегии – с другой стороны. При обучении с разделенной стратегией мы переназначаем веса переходам состояний, применяя выборку по значимости, так чтобы они были пригодны для обучения целевой стратегии, но распределение состояний по-прежнему соответствует поведенческой стратегии. Это несоответствие. Естественно возникает идея как-то изменить веса состояний, увеличив роль одних состояний и уменьшив роль других, так чтобы вернуть распределение обновлений к распределению с единой стратегией. Тогда будет наблюдаться соответствие, и устойчивость и сходимость будут следовать из уже доказанных результатов. В этом и заключается идея эмфатических (с усилением) TD-методов, впервые введенная применительно к обучению с единой стратегией в разделе 9.11.

На самом деле понятие «распределения с единой стратегией» не совсем корректно, потому что таких распределений много, и любого из них достаточно для гарантии устойчивости. Рассмотрим эпизодическую задачу без обесценивания. Мы можем стартовать из состояния, близкого к заключительному, и до завершения эпизода посетить всего несколько состояний с высокой вероятностью. А можем стартовать далеко от заключительного состояния и до завершения успеть посетить много состояний. То и другое – распределения с единой стратегией, и обучение на любом из них линейным полуградиентным методом гарантированно устойчиво. Как бы процесс ни начался, получается распределение с единой стратегией, при условии что все встретившиеся состояния обновляются вплоть до завершения.

Если имеет место обесценивание, то в этом контексте его можно рассматривать как частичное или вероятностное завершение. Если $\gamma = 0.9$, то можно считать, что с вероятностью 0.1 процесс завершается на каждом временном шаге, а затем начинается заново в состоянии, в которое перешел. Задача с обесцениванием отличается тем, что постоянно завершается и перезапускается с вероятностью $1 - \gamma$ на каждом шаге. Такая интерпретация обесценивания – пример более общего понятия *псевдозавершения* – завершения, которое не влияет на последовательность переходов состояний, но влияет на процесс обучения и обученные величины. Подобное псевдозавершение важно для обучения с разделенной стратегией, потому что перезапуск необязателен – помните, что мы можем стартовать любым удобным нам способом, – и завершение освобождает нас от необходимости включать встретившиеся состояния в распределение с единой стратегией. То есть если мы рассматриваем новые состояния как перезапуск, то обесценивание легко дает нам ограниченное распределение с единой стратегией.

Одношаговый эмфатический TD-алгоритм для обучения ценностям состояний в эпизодической задаче определяется следующим образом:

$$\delta_t = R_{t+1} + \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t);$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha M_t \rho_t \delta_t \nabla \hat{v}(S_t, \mathbf{w}_t);$$

$$M_t = \gamma \rho_{t-1} M_{t-1} + I_t,$$

¹ В случае разделенной стратегии A определяется как $\mathbb{E}_{s \sim b}[\mathbf{x}(s)\mathbb{E}[\mathbf{x}(S_{t+1})^T | S_t = s; A_t \sim \pi]]$.

где заинтересованность I_t произвольна, а значимость M_t инициализирована, так что $M_{t-1} = 0$. Как этот алгоритм ведет себя на контрпримере Бэрда? На рис. 11.6 показана траектория математических ожиданий элементов вектора параметров (для случая, когда $I_t = 1$ для всех t). Имеют место колебания, но в конечном итоге все сходится и $\sqrt{VE}$ стремится к нулю. Эти траектории получены итеративно путем вычисления траектории математического ожидания вектора параметров без дисперсии, вызванной выборкой из множества переходов и вознаграждений. Мы не показываем результаты прямого применения эмфатического TD-алгоритма, поскольку его дисперсия на контрпримере Бэрда настолько высока, что почти невозможно получить достоверные результаты в численных экспериментах. Теоретически для этой задачи алгоритм сходится к оптимальному решению, но на практике такого не наблюдается. Мы вернемся к вопросу об уменьшении дисперсии этих алгоритмов в следующем разделе.

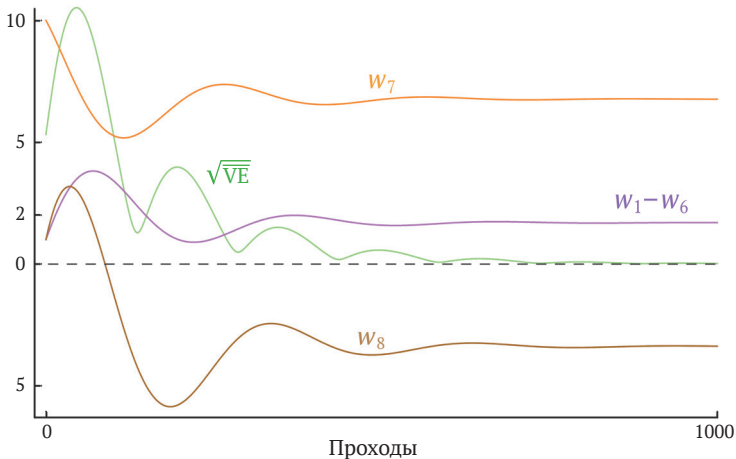


Рис. 11.6 ❖ Поведение математических ожиданий в одношаговом эмфатическом TD-алгоритме на контрпримере Бэрда. Размер шага $\alpha = 0.03$

11.9. УМЕНЬШЕНИЕ ДИСПЕРСИИ

Обучению с разделенной стратегией внутренне присуща более высокая дисперсия, чем обучению с единой стратегией. Оно и неудивительно: если мы получаем данные, не столь тесно связанные со стратегией, то следует ожидать, что в результате обучения узнаем меньше о самой стратегии. В предельном случае, возможно, не узнаем ничего. К примеру, странно было бы надеяться, что можно научиться водить машину, готовя обед. Только если целевая и поведенческая стратегии связаны между собой, если они посещают сходные состояния и предпринимают сходные действия, мы сможем добиться заметного прогресса в обучении с разделенной стратегией.

С другой стороны, для любой стратегии имеется много соседей – похожих стратегий, которые сильно перекрываются по посещенным состояниям и выбирае-

мым действиям, но все же не одинаковы. Смысл обучения с разделенной стратегией в том и состоит, чтобы обеспечить обобщение на это обширное множество связанных, но не идентичных стратегий. Но остается проблема – как оптимально воспользоваться опытом. Теперь, когда мы располагаем некоторыми методами, устойчиво сходящимися к ожидаемой ценности (если размеры шагов выбраны правильно), естественно обратить внимание на уменьшение дисперсии этих оценок. Идей в этой области много, и в данном учебнике вводного уровня мы сможем рассказать лишь о нескольких.

Почему контроль дисперсии особенно важен в методах обучения с разделенной стратегией, основанных на выборке по значимости? Как мы видели, выборка по значимости часто включает произведение отношения стратегий. Математические ожидания отношений всегда равны 1 (5.13), но фактические значения могут быть как очень велики, так и очень близки к нулю. Последовательные отношения не коррелируют, поэтому их произведения также равны 1 в смысле математического ожидания, но могут иметь очень большую дисперсию. Напомним, что в СГС-методах эти отношения умножаются на размер шага, поэтому высокая дисперсия означает выбор шагов, сильно различающихся по размеру. Это проблема для СГС, поскольку иногда могут появляться очень большие шаги. А они не должны быть настолько велики, чтобы перейти в область пространства параметров с совершенно другим градиентом. В основе СГС-методов лежит усреднение по большому числу шагов с целью получить хорошее представление о градиенте, и если метод совершает большие шаги в сторону от примеров, то результаты становятся ненадежными. Если размер шага выбран небольшим, чтобы предотвратить это, то может получиться, что математическое ожидание шага станет очень малым, и обучение сильно замедлится. Тут могут оказаться довольно полезными концепции импульса (Derthick, 1984), усреднения Поляка–Рупперта (Polyak, 1990; Ruppert, 1988; Polyak and Juditsky, 1992) и дальнейшее развитие этих идей. Не стоит забывать и о методах адаптивного выбора различных размеров шага для разных элементов вектора параметров (см., например, Jacobs, 1988; Sutton, 1992b, c), а также обновлениях «с учетом важности весов», описанных в работе Karampatziakis and Langford (2010).

В главе 5 мы видели, что взвешенная выборка по значимости ведет себя значительно лучше, т. е. дисперсия обновлений меньше, чем в случае обыкновенной выборки по значимости. Однако адаптация взвешенной выборки по значимости к аппроксимации функций – трудная задача, которая, вероятно, может быть решена только приближенно и имеет сложность $O(d)$ (Mahmood and Sutton, 2015).

Алгоритм обновления по дереву (раздел 7.5) показывает, что обучение с разделенной стратегией можно в какой-то мере выполнить и без выборки по значимости. Эта идея была обобщена на случай разделенной стратегии в работах Munos, Stepleton, Harutyunyan, and Bellemare (2016) и Mahmood, Yu and Sutton (2017), что позволило получить устойчивые и более эффективные методы.

Еще один, дополнительный подход заключается в том, чтобы целевая стратегия могла частично определяться поведенческой, но так, чтобы расхождения между ними никогда не становились настолько велики, что коэффициенты выборки по значимости окажутся большими. Например, целевую стратегию можно определить, сославшись на поведенческую, как в «распознавателях» (recognizer), предложенных в работе Precup et al. (2006).

11.10. РЕЗЮМЕ

Обучение с разделенной стратегией – интересная задача, бросающая вызов нашей изобретательности в проектировании устойчивых и эффективных алгоритмов обучения. Благодаря табличному Q-обучению создается впечатление, будто обучение с разделенной стратегией – это просто, и у него имеются естественные обобщения на алгоритмы Expected Sarsa и обновления по дереву. Но, как мы видели в этой главе, распространение данных идей на аппроксимацию функций, даже линейную, поднимает новые вопросы и вынуждает нас глубже разобраться в природе алгоритмов обучения с подкреплением.

А зачем такие усилия? Одна из причин, заставляющих нас прибегать к алгоритмам с разделенной стратегией, – обеспечить гибкий компромисс между исследованием и использованием. Другая – развязать поведение и обучение и избавиться от тирании целевой стратегии. TD-обучение, похоже, открывает возможность параллельно обучаться нескольким вещам, т. е. использовать один поток опыта для решения многих задач одновременно. Конечно, это можно делать в особых случаях, а не всегда, когда мы хотели бы, или так эффективно, как хотелось бы.

В этой главе мы разбили задачу об обучении с разделенной стратегией на две части. Первая часть, корректировка целей обучения для поведенческой стратегии, решается прямолинейно с использованием методов, разработанных ранее для табличного случая. Правда, достигается это ценой увеличения дисперсии обновлений, а значит, и замедления обучения. Высокая дисперсия, вероятно, всегда будет бичом обучения с разделенной стратегией.

Вторая часть проблемы обучения с разделенной стратегией проявляется как неустойчивость полугradientных TD-методов, включающих бутстрэппинг. Мы хотели бы получить мощную аппроксимацию функций, обучение с разделенной стратегией, а также эффективность и гибкость бутстрэппинговых TD-методов, но объединить все три члена этой *смертельной триады* в одном алгоритме, не рискуя внести неустойчивость, очень трудно. Было предпринято несколько попыток. Самый популярный подход – выполнять настоящий стохастический градиентный спуск (СГС) по беллмановской ошибке (или беллмановской невязке). Однако из нашего анализа следует, что эта цель во многих случаях вовсе не так привлекательна и что ее ни в коем случае невозможно достичь с помощью алгоритма обучения – градиент $\overline{BE}$ не допускает обучения на опыте, содержащем только векторы признаков, но не истинные состояния. Другой подход, градиентные TD-методы, подразумевает СГС по *спроецированной* беллмановской ошибке. Градиент $\overline{PBE}$ допускает обучение со сложностью $O(d)$, но ценой введения второго вектора параметров и второго размера шага. В самом современном семействе методов, эмфатических TD-методах, уточняется старая идея изменения весов обновлений с целью придания одним большей роли, чем другим. Таким образом, они позволяют восстановить специальные свойства, благодаря которым вычислительно простые полугradientные методы обеспечивают устойчивость обучения с единой стратегией.

Вся область обучения с разделенной стратегией относительно новая и неустоявшаяся. Какие методы считать лучшими или хотя бы приемлемыми, пока не ясно. Действительно ли сложности, присущие новым методам, описанным в конце этой главы, – неизбежность? Какие из них допускают эффективное ком-

бинирование с методами уменьшения дисперсии? Потенциал обучения с разделенной стратегией заманивает, как огоньки на болоте, а как его лучше раскрыть, остается тайной.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

- 11.1 Первым полугradientным методом был линейный алгоритм TD(λ) (Sutton, 1988). Название «полугradientный» родилось позже (Sutton, 2015a). Явная формулировка полугradientного TD(0) с разделенной стратегией с общим коэффициентом выборки по значимости, вероятно, появилась только в работе Sutton, Mahmood, and White (2016), но формы с ценностью действий были уже в работе Precup, Sutton, and Singh (2000), где также приведены формы этих алгоритмов со следами приемлемости (см. главу 12). Их непрерывные варианты без обесценивания пока еще изучены недостаточно. Описанные здесь n -шаговые варианты ранее не публиковались.
- 11.2 Самый ранний пример w -в- $2w$ встречается в работе Tsitsiklis and Van Roy (1996), где также опубликован контрпример, приведенный на стр. 311. Контрпример Бэрда опубликован в работе (1995), а мы включили несколько модифицированный вариант. Методы усреднения для аппроксимации функций осуществлены в работах Gordon (1995, 1996b). Другие примеры неустойчивости методов ДП с разделенной стратегией и более сложных методов аппроксимации приведены в работе Boyan and Moore (1995). В труде Bradtke (1993) приведен пример, в котором применение Q-обучения с линейной аппроксимацией функций к задаче линейно-квадратичного регулирования сходится к дестабилизирующей стратегии.
- 11.3 Смертельная триада впервые описана в работе Sutton (1995b) и всесторонне проанализирована в работе Tsitsiklis and Van Roy (1997). Название «смертельная триада» предложено в работе (2015a).
- 11.4 Этот вид линейного анализа, включая оператор динамического программирования, впервые рассмотрен в работах Tsitsiklis and Van Roy (1996; 1997). Диаграммы типа показанной на рис. 11.3 введены в работе Lagoudakis and Parr (2003).
- 11.5 Величина $\overline{BE}$ впервые предложена в качестве целевой функции для динамического программирования в работе Schweitzer and Seidmann (1985). Бэрд (Baird, 1995, 1999) распространил ее на TD-обучение, основанное на стохастическом градиентном спуске. В литературе минимизацию $\overline{BE}$ часто называют минимизацией беллмановской невязки.
Самый ранний пример с расщеплением состояния A встречается в работе Dayan (1992). Обе приведенные здесь формы взяты из работы Sutton et al. (2009a).
- 11.6 Материал этого раздела публикуется впервые.
- 11.7 Градиентные TD-методы впервые описаны в работе Sutton, Szepesvári, and Maei (2009b). Методы, представленные в этом разделе, взяты из работы Sutton et al. (2009a) и Mahmood et al. (2014). Основное обобщение на

проксимальные TD-методы предпринято в работе Mahadevan et al. (2014). Наиболее интересные на сегодняшний день эмпирические исследования градиентных TD-методов и родственных им содержатся в работах Geist and Scherrer (2014), Dann, Neumann, and Peters (2014), White (2015), and Ghiassian, White, White, and Sutton (готовится к печати). Недавние теоретические достижения в области градиентных TD представлены в работе Yu (2017).

- 11.8** Эмфатические TD-методы введены в работе Sutton, Mahmood, and White (2016). Полные доказательства сходимости и другие теоретические результаты впоследствии были приведены в работах Yu (2015; 2016; Yu, Mahmood, and Sutton, 2017), Hallak, Tamar, and Mannor (2015) и Hallak, Tamar, Munos, and Mannor (2016).

Следы приемлемости

Следы приемлемости – один из основных механизмов обучения с подкреплением. Например, в популярном алгоритме TD(λ) параметр λ указывает на использование следов приемлемости. Почти все методы обучения на основе временных различий (TD-методы), например Q-обучение или Sarsa, можно объединить со следами приемлемости и в результате получить более общий метод, реализующий обучение более эффективно.

Следы приемлемости унифицируют и обобщают TD-методы и методы Монте-Карло. Когда TD-методы дополняются следами приемлемости, образуется новое семейство методов, охватывающее широкий спектр подходов, на одном конце которого ($\lambda = 1$) находятся методы Монте-Карло, а на другом ($\lambda = 0$) – одношаговые TD-методы. Между ними располагаются промежуточные методы, которые часто оказываются лучше, чем крайние варианты. Кроме того, следы приемлемости предлагают способ реализации методов Монте-Карло в онлайн-режиме и для непрерывных задач без эпизодов.

Разумеется, мы уже видели один способ объединения TD-методов и методов Монте-Карло: n -шаговые TD-методы из главы 7. Сверх того следы приемлемости дают элегантный алгоритмический механизм, обладающий значительными вычислительными преимуществами. Конкретно, это временный вектор в памяти, *след приемлемости* $\mathbf{z}_t \in \mathbb{R}^d$, который дополняет постоянный вектор весов $\mathbf{w}_t \in \mathbb{R}^d$. В общих чертах идея состоит в том, что всякий раз, как элемент $\mathbf{w}_t$ участвует в вычислении оценки ценности, соответствующий элемент $\mathbf{z}_t$ резко увеличивается, а затем начинает медленно уменьшаться. Далее обучение в этом элементе $\mathbf{w}_t$ производится, если ненулевая TD-ошибка возникает до того, как след опустится до нуля. Параметр затухания $\lambda \in [0, 1]$ определяет скорость спадания следа.

Главное вычислительное преимущество следов приемлемости над n -шаговыми методами заключается в том, что нужно хранить только один вектор следа, а не n последних векторов признаков. Кроме того, обучение происходит непрерывно и равномерно во времени, а не откладывается до конца эпизода. Дополнительно обучение может иметь место и влиять на поведение сразу после обнаружения состояния, без задержки на n шагов.

Следы приемлемости иллюстрируют тот факт, что алгоритм обучения иногда можно реализовать по-другому ради получения вычислительных преимуществ. Многие алгоритмы наиболее естественно формулируются и интерпретируются как обновление ценности состояния, исходя из событий, которые последовали за этим состоянием на протяжении нескольких временных шагов. Например, в методах Монте-Карло (глава 5) состояние обновляется на основе всех будущих воз-

награждений, а в n -шаговых TD-методах (глава 7) – на основе следующих n вознаграждений и состояния, отстоящего на n шагов в будущем. Такие формулировки, основанные на заглядывании вперед из обновляемого состояния, называются *прямым представлением*. Прямые представления всегда несколько сложнее реализовать, потому что обновление зависит от того, что случится позднее, а в момент обновления неизвестно. Но, как мы покажем в данной главе, часто можно достичь почти такого же, а иногда и точно такого же обновления, с помощью алгоритма, в котором используется текущая TD-ошибка и производится оглядывание назад на недавно посещенные состояния с применением следа приемлемости. Эти альтернативные способы анализа состояния и реализации алгоритмов обучения называются *обратными представлениями*. Обратные представления, преобразования между прямыми и обратными представлениями и их эквивалентность – все это появилось вместе с обучением на основе временных различий, но основное развитие получило после 2014 года. Здесь мы представим в основных чертах современный взгляд на вещи.

Как обычно, сначала мы полностью изложим идеи, относящиеся к ценности состояний и предсказанию, а затем обобщим их на ценности действий и управление. Первым делом мы рассмотрим случай единой стратегии, а потом распространим наши идеи на обучение с разделенной стратегией. Мы будем обращать особое внимание на случай линейной аппроксимации функций, для которого результаты о следах приемлемости сильнее. Все полученные результаты применимы также к табличному случаю и агрегированию состояний, поскольку это частные случаи линейной аппроксимации.

12.1. λ -доход

В главе 7 мы определили n -шаговый доход как сумму первых n вознаграждений и оценки ценности состояния, достигнутого за n шагов, в которой каждое слагаемое соответствующим образом обесценивается (см. формулу 7.1). Общая форма этого определения для произвольного параметрического аппроксиматора функций имеет вид:

$$G_{t:t+n} \doteq R_{t+1} + R_{t+2} + \dots + \gamma^{n-1}R_{t+n} + \gamma^n \hat{v}(S_{t+n}, \mathbf{w}_{t+n-1}), \quad 0 \leq t \leq T - n, \quad (12.1)$$

где T – момент завершения эпизода, если он вообще завершается. В главе 7 мы отметили, что n -шаговый доход для любого n – допустимая цель обновления при табличном обучении, а также при обучении приближенным методом СГС, как в (9.7).

Теперь отметим, что допустимо обновление не только в сторону произвольного n -шагового дохода, но и в сторону любого среднего n -шаговых доходов для различных n . Например обновление можно производить в сторону цели, равной полусумме двух- и четырехшагового дохода $\frac{1}{2}G_{t:t+2} + \frac{1}{2}G_{t:t+4}$. Так можно усреднить любое множество n -шаговых доходов, даже бесконечное, при условии что веса складываемых доходов положительны и в сумме равны 1. Составной доход обла-

дает таким же свойством уменьшения ошибки (7.3), что и отдельные n -шаговые доходы, а потому его можно использовать для построения обновлений со свойствами гарантированной сходимости. Усреднение порождает существенно новый класс алгоритмов. Например, можно усреднить одношаговый и бесконечношаговый доход, получив еще один способ «поженить» TD-методы с методами Монте-Карло. В принципе, можно даже усреднять обновления, основанные на опыте, с обновлениями ДП, получив тем самым простую комбинацию методов, основанных на опыте и на модели (см. главу 8).

Обновление, являющееся результатом усреднения более простых обновлений, называется *составным обновлением*. Диаграмма предшествующих состояний для составного обновления состоит из диаграмм простых компонент с горизонтальной чертой наверху, под каждой из которых проставлен ее вес. Например, на рисунке справа показана диаграмма упомянутого в начале этого раздела составного обновления, равного полусумме двух- и четырехшагового дохода. Составное обновление можно выполнить только после завершения самой длинной его компоненты. Так, обновление справа для оценки, формируемой в момент t , можно было бы выполнить только в момент $t + 4$. В общем случае желательно ограничивать длину самой длинной компоненты обновления, чтобы не создавать задержек.

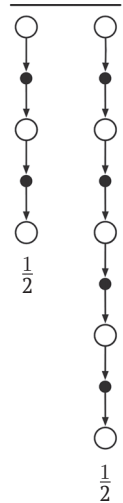
Алгоритм TD(λ) можно интерпретировать как конкретный способ усреднения n -шаговых обновлений. Это усреднение содержит все n -шаговые обновления, каждое с весом, пропорциональным λ^{n-1} (где $\lambda \in [0, 1]$). Нормировочный коэффициент $1 - \lambda$ гарантирует, что сумма весов равна 1 (рис. 12.1). Получающееся обновление стремится к так называемому λ -доходу, определяемому следующим образом:

$$G_t^\lambda \doteq (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n}. \quad (12.2)$$

На рис. 12.2 изображена схема взвешивания n -шаговых доходов, входящих в состав λ -дохода. Одношаговому доходу назначен наибольший вес $1 - \lambda$, двухшаговому – следующий по величине вес $(1 - \lambda)\lambda$, трехшаговому – вес $(1 - \lambda)\lambda^2$ и т. д. На каждом шаге вес уменьшается в λ раз. По достижении заключительного состояния все последующие n -шаговые доходы берутся равными традиционному доходу G_t . При желании можно отделить эти члены от основной суммы, записав

$$G_t^\lambda = (1 - \lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_t, \quad (12.3)$$

как показано на рисунках. Из этого уравнения становится понятнее, что происходит, когда $\lambda = 1$. В этом случае основная сумма обращается в ноль, а оставшийся член сводится к традиционному доходу. Таким образом, при $\lambda = 1$ обновление по формуле λ -дохода совпадает с алгоритмом Монте-Карло. С другой стороны, при $\lambda = 0$ λ -доход сводится к одношаговому TD-методу.



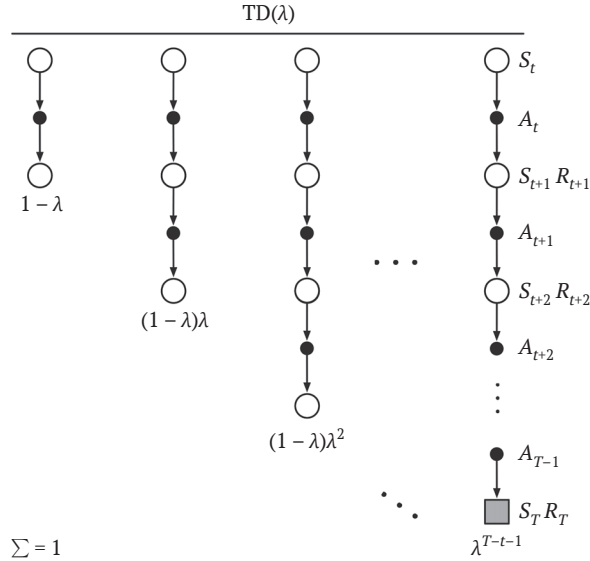


Рис. 12.1 ❖ Диаграмма предшествующих состояний для TD(λ). При $\lambda = 0$ полное обновление сводится к первой компоненте – одношаговому обновлению. А при $\lambda = 1$ полное обновление сводится к последней компоненте – обновлению Монте-Карло

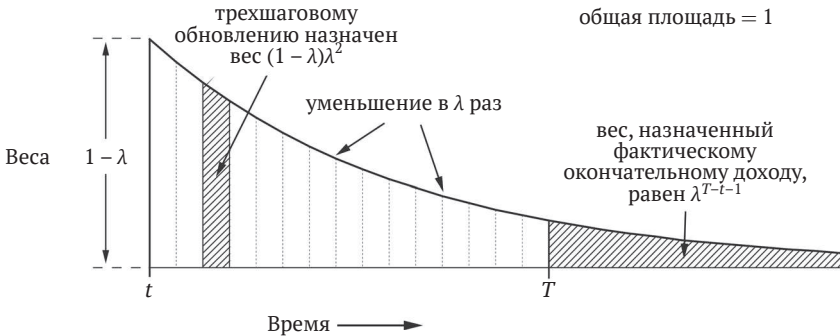


Рис. 12.2 ❖ Веса, назначаемые каждому n -шаговому доходу в λ -доходе

Упражнение 12.1. Доход можно выразить рекуррентно через первое вознаграждение и доход на следующем шаге (3.9), и то же самое справедливо в отношении λ -дохода. Выведите аналогичное рекуррентное соотношение из формул (12.2) и (12.1). □

Упражнение 12.2. Параметр λ характеризует скорость экспоненциального убывания весов на рис. 2.2, а значит, то, насколько далеко в будущее заглядывает λ -доходный алгоритм при вычислении обновления. Но коэффициенты типа λ – не всегда удачный способ описать скорость убывания. Иногда лучше задавать временную постоянную, или время полужизни. Напишите уравнение, связывающее λ и время полужизни T_λ , т. е. время, за которое вес снизится до половины начального значения. □

Теперь мы готовы определить наш первый алгоритм обучения на основе λ-дохода: *офлайнный λ-доходный алгоритм*. Будучи офлайнным, этот алгоритм не изменяет вектор весов на протяжении эпизода. А в конце эпизода выполняется вся последовательность офлайнных обновлений в соответствии с нашим обычным полугradientным правилом, где в качестве цели используется λ-доход:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha [G_t^\lambda - \hat{v}(S_t, \mathbf{w}_t)] \nabla \hat{v}(S_t, \mathbf{w}_t), \quad t = 0, \dots, T - 1. \quad (12.4)$$

λ-доход дает еще один способ плавного перехода от методов Монте-Карло к одношаговым TD-методам, который можно сравнить с n -шаговым бутстрэппингом, разработанным в главе 7. Там мы оценивали эффективность на примере задачи о случайном блуждании с 19 состояниями (пример 7.1). На рис. 12.3 показаны результаты офлайнного λ-доходного алгоритма и n -шаговых методов (воспроизведение рис. 7.2) на одной и той же задаче. Эксперимент проводился, как было описано выше, с тем отличием, что для λ-доходного алгоритма варьировалось не n , а λ. В качестве меры качества использовалась среднеквадратическая ошибка между правильной и оценочной ценностями каждого состояния, измеренная в конце эпизода, усредненная по первым 10 эпизодам и 19 состояниям. Заметим, что в целом качество офлайнных λ-доходных алгоритмов сопоставимо с качеством n -шаговых алгоритмов. В обоих случаях наилучшие результаты получаются при промежуточном значении параметра бутстрэппинга: n для n -шаговых методов и λ для офлайнных λ-доходных алгоритмов.

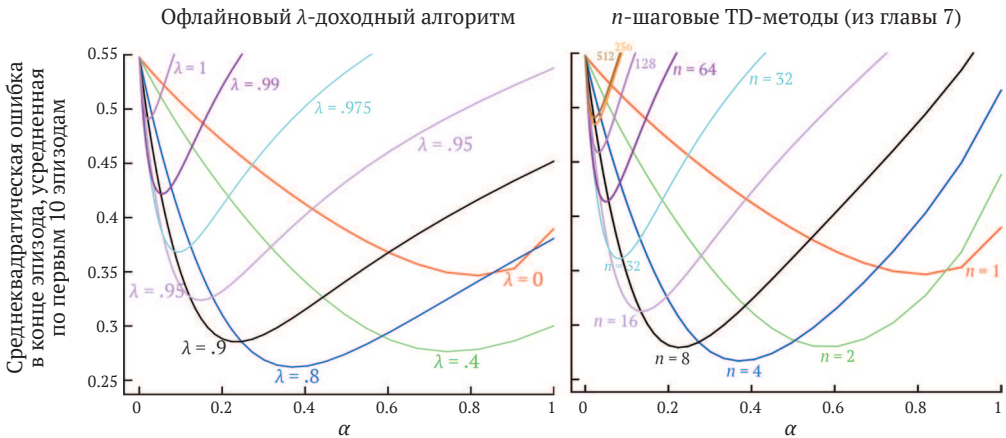


Рис. 12.3 ❖ Результаты случайного блуждания (пример 7.1): сравнение качества офлайнного λ-доходного алгоритма с n -шаговыми TD-методами. В обоих случаях наилучшие результаты получаются при промежуточном значении параметра бутстрэппинга (λ или n). Для офлайнного λ-доходного алгоритма результаты чуть лучше при оптимальных значениях α и λ и при больших α

Описанный выше подход мы называем теоретическим, или *прямым*, представлением алгоритма обучения. Для каждого посещенного состояния мы заглядываем вперед, т. е. просматриваем все будущие состояния и решаем, как их лучше скомбинировать. Можно представить себе, что мы оседлали поток состояний и из каждого состояния смотрим вперед, чтобы определить его обновление, как пока-

зано на рис. 12.4. После обновления одного состояния мы переходим к следующему и больше никогда не возвращаемся к предыдущему. В свою очередь, будущие состояния неоднократно просматриваются и обрабатываются, по одному разу из каждой предшествующей им наблюдательной позиции.

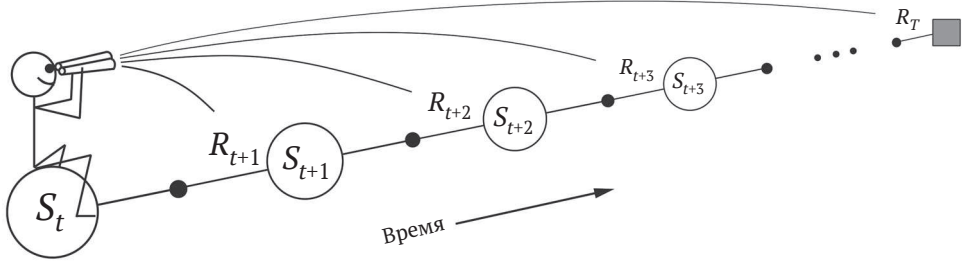


Рис. 12.4 ❖ Прямое представление. Мы решаем, как обновить каждое состояние, анализируя будущие вознаграждения и состояния

12.2. TD(λ)

TD(λ) – один из самых почтенных и широко распространенных алгоритмов обучения с подкреплением. Это первый алгоритм, для которого была установлена связь между двумя представлениями: прямым, носящим скорее теоретический характер, и обратным – более ориентированным на вычисления и использующим следы приемлемости. Здесь мы эмпирически покажем, что он аппроксимирует офлайн-λ-доходный алгоритм, описанный в предыдущем разделе.

TD(λ) улучшает офлайн-λ-доходный алгоритм в трех отношениях. Во-первых, вектор весов обновляется на каждом шаге эпизода, а не только в конце, поэтому лучшую оценку, возможно, удастся получить за меньшее время. Во-вторых, вычисления равномерно распределены во времени, а не выполняются разом в конце эпизода. И, в-третьих, алгоритм применим к непрерывным задачам, а не только к эпизодическим. В этом разделе мы представим полугradientный вариант TD(λ) с аппроксимацией функций.

В случае аппроксимации функций след приемлемости является вектором $\mathbf{z}_t \in \mathbb{R}^d$ той же размерности, что вектор весов $\mathbf{w}_t$. Но вектор весов хранится в памяти на протяжении всей жизни системы, а след приемлемости обычно меньше, чем длится один эпизод. Следы приемлемости способствуют процессу обучения; единственное их назначение – воздействовать на вектор весов, после чего уже сам вектор весов определяет оценку ценности.

В алгоритме TD(λ) вектор следа приемлемости инициализируется нулем в начале эпизода, увеличивается на каждом временном шаге на величину градиента ценности, а затем уменьшается на $\gamma\lambda$:

$$\begin{aligned} \mathbf{z}_{-1} &\doteq \mathbf{0}, \\ \mathbf{z}_t &\doteq \gamma\lambda\mathbf{z}_{t-1} + \nabla\hat{V}(S_t, \mathbf{w}_t), \quad 0 \leq t \leq T, \end{aligned} \tag{12.5}$$

где γ – коэффициент обесценивания, а λ – параметр, определенный в предыдущем разделе, который мы далее будем называть *скоростью затухания следа*. След приемлемости отслеживает, какие элементы вектора весов внесли вклад, положительный или отрицательный, в недавнее изменение оценки ценности состояния, причем смысл слова «недавно» определяется в терминах $\gamma\lambda$. (Напомним, что при линейной аппроксимации функций $\nabla\hat{v}(S_t, \mathbf{w}_t)$ – это просто вектор признаков $\mathbf{x}_t$, и в этом случае вектор следа приемлемости – сумма прошлых затухающих входных векторов.) Говорят, что след показывает степень приемлемости каждого элемента вектора весов для происходящих в процессе обучения изменений в случае, если произойдет подкрепляющее событие. Интересующие нас подкрепляющие события – это TD-ошибки на одном шаге. TD-ошибка при предсказании ценности состояния имеет вид:

$$\delta_t \doteq R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t). \quad (12.6)$$

В алгоритме TD(λ) вектор весов обновляется на каждом шаге пропорционально скалярной TD-ошибке и векторному следу приемлемости:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\delta_t\mathbf{z}_t. \quad (12.7)$$

Полугradientный TD(λ) для оценивания $\hat{v} \approx v_\pi$

Вход: подлежащая оценке стратегия π

Вход: дифференцируемая функция $\hat{v}: S^+ \times \mathbb{R}^d \rightarrow \mathbb{R}$ такая, что $\hat{v}(\text{terminal}, \cdot) = 0$

Параметры алгоритма: размер шага $\alpha > 0$, скорость затухания следа $\lambda \in [0, 1]$

Инициализировать веса функции ценности $\mathbf{w} \in \mathbb{R}^d$ произвольным образом (например, $\mathbf{w} = \mathbf{0}$)

Повторять для каждого эпизода:

Инициализировать S

$\mathbf{z} \leftarrow \mathbf{0}$

(d -мерный вектор)

Повторять для каждого шага эпизода:

Выбрать $A \sim \pi(\cdot, S)$

Предпринять действие A , наблюдать R, S'

$\mathbf{z} \leftarrow \gamma\lambda\mathbf{z} + \nabla\hat{v}(S, \mathbf{w})$

$\delta \leftarrow R + \gamma\hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$

$\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\mathbf{z}$

$S \leftarrow S'$

пока S' не является заключительным состоянием

TD(λ) ориентирован назад во времени. В каждый момент мы видим текущую TD-ошибку и применяем ее ко всем предыдущим состояниям в соответствии с вкладом каждого состояния в след приемлемости в тот момент времени. Можно представить себе, что мы оседлали поток состояний, вычисляем TD-ошибки и выкрикиваем их, донося до ранее посещенных состояний, как показано на рис. 12.5. Там, где TD-ошибки и следы встречаются, мы производим обновление по формуле (12.7), изменяя ценности прошлых состояний на случай, если они снова встретятся в будущем.

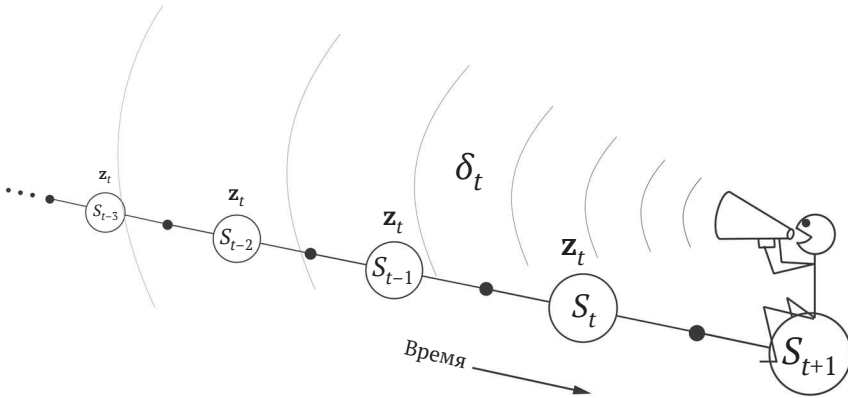


Рис. 12.5 ❖ Механистическое, или обратное, представление $TD(\lambda)$. Каждое обновление зависит от текущей TD -ошибки в сочетании с текущими следами приемлемости прошлых событий

Чтобы лучше разобраться в обратном представлении $TD(\lambda)$, посмотрим, что происходит при различных значениях λ . Если $\lambda = 0$, то, в силу (12.5), след в момент t в точности равен градиенту ценности, соответствующему S_t . Таким образом, обновление (12.7) $TD(\lambda)$ сводится к обновлению в одношаговом полуградиентном алгоритме TD , который рассматривался в главе 9 (а в табличном случае – к простому TD -правилу (6.2)). Потому-то этот алгоритм и называется (0). В терминах рис. 12.5 $TD(0)$ – это случай, когда TD -ошибка изменяет только одно состояние, предшествующее текущему. Если значение λ больше, но все-таки $\lambda < 1$, то изменяется больше предшествующих состояний, но чем дальше во времени отстоит состояние, тем меньше оно изменяется, потому что соответствующий ему след приемлемости меньше, что и показано на рисунке. Мы говорим, что более ранним состояниям назначается меньшее *поощрение* за TD -ошибку.

Если $\lambda = 1$, то поощрение, назначенное более ранним состояниям, уменьшается только на γ на каждом шаге. Это как раз то, что нужно для получения поведения, свойственного методам Монте-Карло. Напомним, к примеру, что TD -ошибка δ_t включает необесцененный член R_{t+1} . При возврате на k шагов его необходимо обесценить с коэффициентом γ^k , как любое вознаграждение, входящее в состав дохода, и именно этого эффекта достигают убывающие следы приемлемости. Если $\lambda = 1$ и $\gamma = 1$, то следы приемлемости со временем вообще не убывают. В этом случае метод ведет себя как метод Монте-Карло для эпизодической задачи без обесценивания. При $\lambda = 1$ алгоритм также называют $TD(1)$.

$TD(1)$ является более общим способом реализации алгоритмов Монте-Карло, чем ранее представленные, и значительно расширяет сферу их применимости. В то время как применение описанных выше методов Монте-Карло ограничено эпизодическими задачами, $TD(1)$ можно применять и к непрерывным задачам с обесцениванием. Более того, $TD(1)$ может работать инкрементно в онлайн-режиме. Одним из недостатков методов Монте-Карло является тот факт, что они не могут обучаться, пока эпизод не завершится. Например, если метод управления Монте-Карло предпринимает действие, приносящее очень плохое вознаграждение, но не завершающее эпизод, то это никак не уменьшит стремления агента

повторять это действие на протяжении эпизода. Напротив, онлайнный TD(1) обучается, как n -шаговый алгоритм TD, в течение всего эпизода (n – количество следующих шагов после текущего). Когда в эпизоде происходит что-то необычно хорошее или плохое, методы управления на основе TD(1) обучаются немедленно и изменяют свое поведение уже в этом эпизоде.

Интересно еще раз вернуться к примеру случайного блуждания с 19 состояниями (пример 7.1) и посмотреть, как TD(λ) справляется с аппроксимацией офлайнного λ -доходного алгоритма. Результаты обоих алгоритмов показаны на рис. 12.6. При каждом значении λ и оптимальном для него (или меньшем) значении α оба алгоритма дают практически одинаковые результаты. Но если α больше оптимального значения, то λ -доходный алгоритм работает лишь немного хуже, тогда как поведение TD(λ) резко ухудшается и даже может стать неустойчивым. Для применения TD(λ) к данной задаче это не катастрофа, поскольку мы в любом случае не хотим завышать эти параметры, но в других случаях может оказаться важной слабостью.

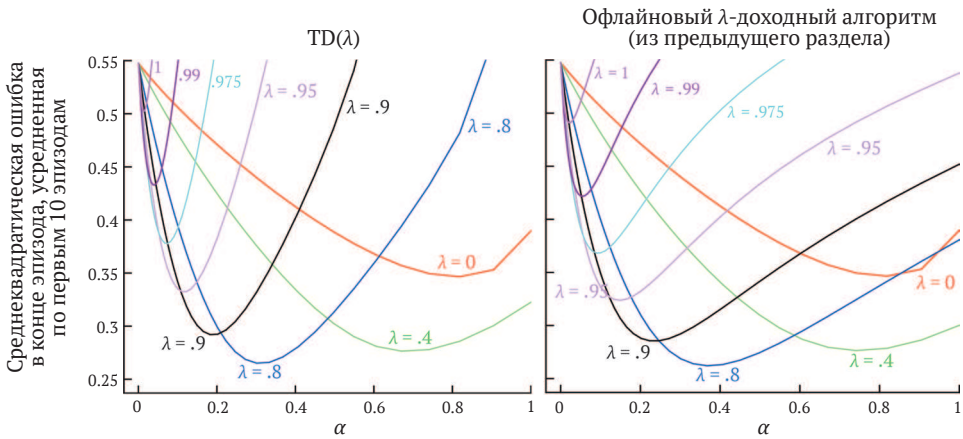


Рис. 12.6 ❖ Результаты для задачи о случайном блуждании с 19 состояниями (пример 7.1): показаны результаты TD(λ) и офлайнного λ -доходного алгоритма. Оба алгоритма работают практически одинаково при небольших (меньших оптимального) значениях α , но при больших α TD(λ) оказывается значительно хуже

Доказано, что линейный TD(λ) сходится в случае единой стратегии, если размер шага уменьшается со временем в соответствии с обычными условиями (2.7). Как было сказано в разделе 9.4, он сходится не к вектору весов, дающему минимальную ошибку, а к близко расположенному вектору, зависящему от λ . Верхнюю границу качества решения (9.14) теперь можно обобщить на любое λ . В непрерывном случае с обесцениванием имеем:

$$\overline{\text{VE}}(\mathbf{w}_\infty) \leq \frac{1 - \gamma\lambda}{1 - \gamma} \min_{\mathbf{w}} \overline{\text{VE}}(\mathbf{w}). \quad (12.8)$$

Это означает, что асимптотическая ошибка не более чем в $(1 - \gamma\lambda)/(1 - \gamma)$ раз превышает наименьшую возможную ошибку. Когда λ стремится к 1, верхняя граница стремится к минимальной ошибке (а самой слабой оказывается при $\lambda = 0$).

На практике, однако, $\lambda = 1$ – самый плохой выбор, что будет продемонстрировано ниже на рис. 12.14.

Упражнение 12.3. Получить более полное представление о том, насколько хорошо $TD(\lambda)$ может аппроксимировать офлайнный λ -доходный алгоритм, можно, заметив, что член ошибки в последнем (выражение в квадратных скобках в (12.4)) можно записать в виде суммы TD -ошибок (12.6) для одного фиксированного $\mathbf{w}$. Докажите это, следуя образцу (6.6) и воспользовавшись рекуррентным соотношением для λ -дохода, которое вы вывели, решая упражнение 12.1. $\square$

Упражнение 12.4. Воспользуйтесь результатом предыдущего упражнения, чтобы показать, что если бы обновления весов в эпизоде вычислялись на каждом шаге, но не использовались для изменения весов ($\mathbf{w}$ оставался бы фиксированным), то сумма обновлений весов в $TD(\lambda)$ была бы такой же, как сумма обновлений в офлайнном λ -доходном алгоритме. $\square$

12.3. n -ШАГОВЫЕ УСЕЧЕННЫЕ λ -ДОХОДНЫЕ МЕТОДЫ

Офлайнный λ -доходный алгоритм – важный идеал, но его полезность ограничена из-за использования λ -дохода (12.2), который остается неизвестным до конца эпизода. В непрерывном случае λ -доход, строго говоря, никогда не известен, поскольку зависит от n -шаговых доходов при сколь угодно больших n , а значит, от вознаграждений в сколь угодно далеком будущем. Однако зависимость от отложенных на долгий срок вознаграждений ослабевает, уменьшаясь на $\gamma \lambda$ за каждый шаг задержки. Раз так, то было бы естественно обрезать последовательность после некоторого количества шагов. Наше текущее понятие n -шагового дохода дает очевидный способ сделать это, заменив отсутствующие вознаграждения оценками.

В общем случае мы определяем *усеченный λ -доход* в момент t , располагая только данным вплоть до некоторого горизонта h , следующим образом:

$$G_{t:h}^{\lambda} \doteq (1 - \lambda) \sum_{n=1}^{h-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{h-t-1} G_{t:h}, \quad 0 \leq t < h \leq T. \quad (12.9)$$

Если сравнить это выражение с λ -доходом (12.3), то становится ясно, что горизонт h играет ту же роль, которую раньше играл T , момент завершения. В то время как в λ -доходе имеется остаточный вес, назначаемый традиционному доходу G_t , здесь он назначается самому длинному из доступных n -шаговых доходов, $G_{t:h}$ (рис. 12.2).

Понятие усеченного дохода сразу же порождает семейство n -шаговых λ -доходных алгоритмов, аналогичных n -шаговым методам из главы 7. Во всех этих алгоритмах обновления откладываются на n шагов и принимаются во внимание только первые n вознаграждений, но теперь включены все k -шаговые доходы для $1 \leq k \leq n$ (тогда как в прежних n -шаговых алгоритмах использовался только n -шаговый доход), которым сопоставлены геометрические веса, как показано на рис. 12.2.

В случае ценности состояний это семейство алгоритмов называется *усеченные TD(λ)*, или *TTD(λ)*. Составная диаграмма предшествующих состояний, изображенная на рис. 12.7, похожа на диаграмму для TD(λ) (рис. 12.1) с тем отличием, что обновление, описываемое самой длинной компонентой, не превышает n шагов, а не простирается до конца эпизода. Семейство TTD(λ) определяется следующим уравнением (сравните с (9.15)):

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha[G_{t:t+n}^\lambda - \hat{v}(S_t, \mathbf{w}_{t+n-1})]\nabla\hat{v}(S_t, \mathbf{w}_{t+n-1}), \quad 0 \leq t < T.$$

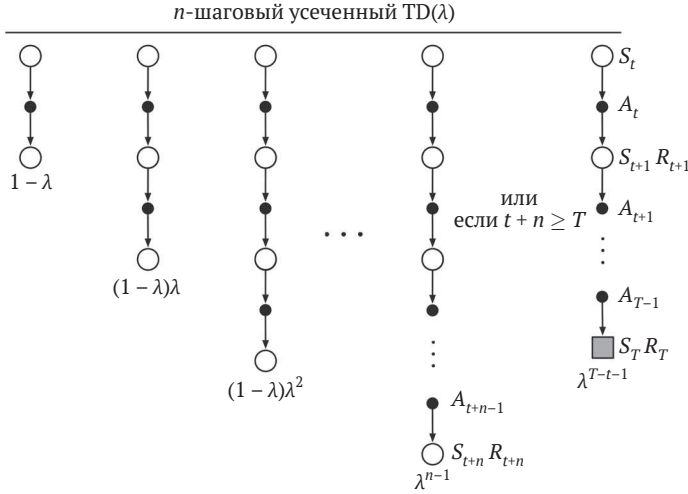


Рис. 12.7 ❖ Диаграмма предшествующих состояний для усеченного TD(λ)

Этот алгоритм можно реализовать эффективно, так что объем вычислений на одном шаге не будет расти с увеличением n (объем потребляемой памяти, конечно, будет расти). Как и в n -шаговых TD-методах, на первых $n - 1$ шагах не производится никаких обновлений, а $n - 1$ добавочных обновлений выполняются после завершения. В эффективной реализации используется тот факт, что k -шаговый λ -доход можно записать в виде:

$$G_{t:t+k}^\lambda = \hat{v}(S_t, \mathbf{w}_{t-1}) + \sum_{i=t}^{t+k-1} (\gamma\lambda)^{i-t} \delta'_i, \quad (12.10)$$

где

$$\delta'_t := R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_{t-1}).$$

Упражнение 12.5. Мы уже несколько раз (часто в упражнениях) устанавливали, что доход можно записать в виде суммы TD-ошибок, если функция ценности остается постоянной. Почему (12.10) является еще одним примером такого рода? Докажите тождество (12.10).

12.4. ПЕРЕСЧЕТ ОБНОВЛЕНИЙ: ОНЛАЙНОВЫЙ λ -ДОХОДНЫЙ АЛГОРИТМ

Выбор параметра усечения n в усеченном TD(λ) подразумевает компромисс. Параметр n должен быть достаточно большим, чтобы метод хорошо аппроксимировал офлайн- λ -доходный алгоритм, но достаточно малым, чтобы обновления производились как можно раньше и могли раньше повлиять на поведение. Можно ли получить и то, и другое? В принципе да, но ценой увеличения вычислительной сложности.

Идея заключается в том, чтобы на каждом временном шаге, получив новую порцию данных, отступить назад и заново произвести все обновления с момента начала текущего эпизода. Новые обновления будут лучше предыдущих, потому что учитывают новые данные. Иначе говоря, в качестве цели обновлений всегда выступает усеченный λ -доход, но всякий раз используется самый последний горизонт. При каждом проходе по эпизоду можно использовать несколько более длинный эпизод и получить несколько лучшие результаты. Напомним, что усеченный λ -доход определен формулой

$$G_{t:h}^{\lambda} \doteq (1 - \lambda) \sum_{n=1}^{h-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{h-t-1} G_{t:h}.$$

Давайте проанализируем, как можно было бы использовать эту цель, если бы мы не были ограничены вычислительной сложностью. Эпизод начинается с оценки в момент 0, для которой используются веса $\mathbf{w}_0$, вычисленные в конце предыдущего эпизода. Обучение начинается, когда горизонт данных сдвинулся на временной шаг 1. Целью для оценки на шаге 0 при наличии данных вплоть до горизонта 1 мог бы быть только одношаговый доход $G_{0:1}$, который включает R_1 и результаты бутстрэппинга от оценки $\hat{v}(S_1, \mathbf{w}_0)$. Заметим, что это в точности $G_{0:1}^{\lambda}$, поскольку сумма в первом члене выражения выродилась в 0. Используя эту цель обновления, мы конструируем $\mathbf{w}_1$. А что делать дальше, после сдвига горизонта данных на шаг 2? У нас есть новые данные в форме R_2 и S_2 , а также новый вектор $\mathbf{w}_1$, так что теперь мы можем построить лучшую цель обновления $G_{0:2}^{\lambda}$ для первого обновления из S_0 и лучшую цель обновления $G_{1:2}^{\lambda}$ для второго обновления из S_1 . Имея эти улучшенные цели, мы пересчитываем обновления в S_1 и S_2 , снова стартуя из $\mathbf{w}_0$, и таким образом получаем $\mathbf{w}_2$. Затем горизонт сдвигается на шаг 3 и все повторяется ради порождения трех новых целей – мы пересчитываем обновления, начав с исходного $\mathbf{w}_0$, и в итоге вычисляем $\mathbf{w}_3$. И так далее. Каждый раз, как горизонт сдвигается на один шаг, все обновления пересчитываются, начиная с $\mathbf{w}_0$, используя вектор весов, вычисленный на предыдущем горизонте.

Это концептуальное изложение алгоритма включает несколько проходов по эпизоду, по одному для каждого горизонта, на каждом из которых генерирует-

ся новая последовательность векторов весов. Чтобы формально описать его, мы должны различать векторы весов, вычисленные на разных горизонтах. Будем обозначать $\mathbf{w}_t^h$ веса, используемые для генерации ценности в момент t в последовательности, ограниченной горизонтом h . Первый вектор весов $\mathbf{w}_0^h$ в каждой последовательности унаследован от предыдущего эпизода (поэтому они одинаковы для всех h), а последний вектор $\mathbf{w}_h^h$ в каждой последовательности входит в итоговую последовательность векторов весов, вырабатываемую алгоритмом. На последнем горизонте $h = T$ мы получаем окончательные веса $\mathbf{w}_T^T$, которые передаются дальше, чтобы сформировать начальные веса для следующего эпизода. В этих обозначениях первые три последовательности, описанные в предыдущем абзаце, можно выписать явно:

$$h = 1: \quad \mathbf{w}_1^1 \doteq \mathbf{w}_0^1 + \alpha[G_{0:1}^\lambda - \hat{v}(S_0, \mathbf{w}_0^1)] \nabla \hat{v}(S_0, \mathbf{w}_0^1),$$

$$h = 2: \quad \begin{aligned} \mathbf{w}_1^2 &\doteq \mathbf{w}_0^2 + \alpha[G_{0:2}^\lambda - \hat{v}(S_0, \mathbf{w}_0^2)] \nabla \hat{v}(S_0, \mathbf{w}_0^2), \\ \mathbf{w}_2^2 &\doteq \mathbf{w}_1^2 + \alpha[G_{1:2}^\lambda - \hat{v}(S_1, \mathbf{w}_1^2)] \nabla \hat{v}(S_1, \mathbf{w}_1^2), \end{aligned}$$

$$h = 3: \quad \begin{aligned} \mathbf{w}_1^3 &\doteq \mathbf{w}_0^3 + \alpha[G_{0:3}^\lambda - \hat{v}(S_0, \mathbf{w}_0^3)] \nabla \hat{v}(S_0, \mathbf{w}_0^3), \\ \mathbf{w}_2^3 &\doteq \mathbf{w}_1^3 + \alpha[G_{1:3}^\lambda - \hat{v}(S_1, \mathbf{w}_1^3)] \nabla \hat{v}(S_1, \mathbf{w}_1^3), \\ \mathbf{w}_3^3 &\doteq \mathbf{w}_2^3 + \alpha[G_{2:3}^\lambda - \hat{v}(S_2, \mathbf{w}_2^3)] \nabla \hat{v}(S_2, \mathbf{w}_2^3). \end{aligned}$$

Общая форма обновления имеет вид:

$$\mathbf{w}_{t+1}^h \doteq \mathbf{w}_t^h + \alpha[G_{t:h}^\lambda - \hat{v}(S_t, \mathbf{w}_t^h)] \nabla \hat{v}(S_t, \mathbf{w}_t^h), \quad 0 \leq t < h \leq T.$$

Это обновление в сочетании с определением $\mathbf{w}_t \doteq \mathbf{w}_t^t$ описывает *онлайнный λ -доходный алгоритм*.

Онлайнный λ -доходный алгоритм действительно работает в онлайнном режиме, т. е. вычисляет новый вектор весов $\mathbf{w}_t$ на каждом шаге t эпизода, используя только информацию, доступную в момент t . Его главный недостаток – вычислительная сложность, поскольку он повторяет обработку уже пройденной части эпизода на каждом шаге. Заметим, что он строго сложнее офлайнного λ -доходного алгоритма, который проходит по всем шагам в момент завершения, но не производит никаких обновлений во время эпизода. Можно ожидать, что в качестве компенсации онлайнный алгоритм будет работать лучше офлайнного, не только внутри эпизода, когда он выполняет обновление, а офлайнный алгоритм – нет, но и в конце эпизода, потому что вектор весов, применяемый при бутстрэппинге (в $G_{t:h}^\lambda$), включил в себя больше информативных обновлений. Этот эффект можно заметить, внимательно взглянув на рис. 12.8, где оба алгоритма сравниваются на задаче о случайном блуждании с 19 состояниями.

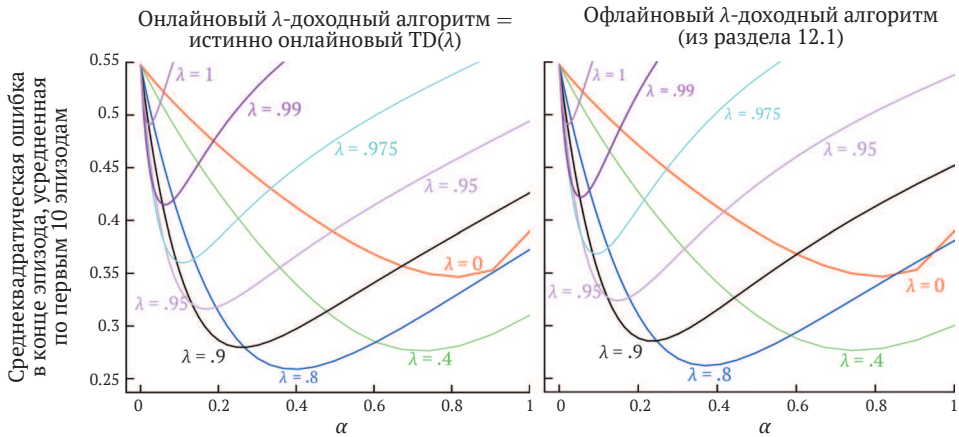


Рис. 12.8 ❖ Результаты для задачи о случайном блуждании с 19 состояниями (пример 7.1): показаны результаты онлайн- и офлайн- λ -доходных алгоритмов. В качестве меры качества здесь взята ошибка $\sqrt{E}$ в конце эпизода, которая должна показывать наилучшие результаты для офлайн-алгоритма. Тем не менее онлайн-алгоритм работает немного лучше. Для сравнения, кривая, соответствующая $\lambda = 0$, для обоих методов одинакова

12.5. Истинно онлайновый TD(λ)

Только что представленный онлайн- λ -доходный алгоритм в настоящее время является лучшим алгоритмом на основе временных различий. Это идеал, который онлайн- $\text{TD}(\lambda)$ всего лишь аппроксимирует. Однако в таком виде онлайн- λ -доходный алгоритм слишком сложен. Нет ли какого-нибудь способа, применив следы приемлемости, инвертировать этот алгоритм с прямым представлением и получить эффективный алгоритм с обратным представлением? Оказывается, существует вычислительно эффективная реализация онлайн- λ -доходного алгоритма для случая линейной аппроксимации функций. Эта реализация известна под названием «истинно онлайновый $\text{TD}(\lambda)$ », потому что она ближе к идеалу онлайн- λ -доходного алгоритма, чем сам алгоритм $\text{TD}(\lambda)$.

Вывод истинно онлайнового $\text{TD}(\lambda)$ слишком сложен для включения в эту книгу (см. следующий раздел и приложение к статье van Seijen et al., 2016), но сама идея проста. Последовательность векторов весов, порождаемую онлайн- λ -доходным алгоритмом, можно организовать в виде треугольника:

$$\begin{array}{ccccccc}
 \mathbf{w}_0^0 & & & & & & \\
 \mathbf{w}_0^1 & \mathbf{w}_1^1 & & & & & \\
 \mathbf{w}_0^2 & \mathbf{w}_1^2 & \mathbf{w}_2^2 & & & & \\
 \mathbf{w}_0^3 & \mathbf{w}_1^3 & \mathbf{w}_2^3 & \mathbf{w}_3^3 & & & \\
 \vdots & \vdots & \vdots & \vdots & \ddots & & \\
 \mathbf{w}_0^T & \mathbf{w}_1^T & \mathbf{w}_2^T & \mathbf{w}_3^T & \cdots & \mathbf{w}_T^T &
 \end{array}$$

На каждом временном шаге создается одна строка этого треугольника. Но в действительности нужны только векторы весов на диагонали, $\mathbf{w}_t^t$. Первый вектор $\mathbf{w}_0^0$ – это начальный вектор весов эпизода, последний $\mathbf{w}_T^T$ – окончательный вектор весов, а каждый вектор между ними, $\mathbf{w}_t^t$, играет роль в бутстрэппинге при вычислении n -шаговых доходов обновлений. В окончательном алгоритме диагональные векторы весов записываются без верхнего индекса – просто $\mathbf{w}_t \doteq \mathbf{w}_t^t$. Наша задача – найти компактный и эффективный способ вычисления каждого $\mathbf{w}_t^t$ по предыдущему. Если мы сумеем ее решить для линейного случая, в котором $\hat{v}(s, \mathbf{w}) = \mathbf{w}^T \mathbf{x}(s)$, то получим истинно онлайнный алгоритм TD(λ):

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \delta_t \mathbf{z}_t + \alpha (\mathbf{w}_t^T \mathbf{x}_t - \mathbf{w}_{t-1}^T \mathbf{x}_t) (\mathbf{z}_t - \mathbf{x}_t),$$

где использована сокращенная нотация $\mathbf{x}_t \doteq \mathbf{x}(S_t)$, δ_t определено, как в TD(λ) формулой (12.6), а $\mathbf{z}_t$ по определению равно

$$\mathbf{z}_t \doteq \gamma \lambda \mathbf{z}_{t-1} + (1 - \alpha \gamma \lambda \mathbf{z}_{t-1}^T \mathbf{x}_t) \mathbf{x}_t. \quad (12.11)$$

Доказано, что этот алгоритм порождает в точности ту же последовательность векторов весов $\mathbf{w}_t$, $0 \leq t \leq T$, что и онлайнный λ -доходный алгоритм (van Seijen et al. 2016). Таким образом, результаты для задачи о случайном блуждании в левой части рис. 12.8 также являются результатами этого алгоритма для той же задачи. Однако этот алгоритм гораздо менее затратный. Требования к памяти, предъявляемые истинно онлайнным TD(λ), совпадают с требованиями традиционного TD(λ), тогда как объем вычислений на одном шаге примерно на 50 % больше (в обновлении следа приемлемости есть одно дополнительное скалярное произведение). Общая вычислительная сложность на одном шаге по-прежнему имеет порядок $O(d)$, как и в TD(λ). Полный псевдокод алгоритма приведен во врезке ниже.

Истинно онлайнный TD(λ) для оценивания $\mathbf{w}^T \mathbf{x} \approx v_\pi$

Вход: подлежащая оценке стратегия π

Вход: функция признаков $\mathbf{x}: \mathcal{S}^+ \rightarrow \mathbb{R}^d$ такая, что $\mathbf{x}(\text{terminal}, \cdot) = \mathbf{0}$

Параметры алгоритма: размер шага $\alpha > 0$, скорость затухания следа $\lambda \in [0, 1]$

Инициализировать веса функции ценности $\mathbf{w} \in \mathbb{R}^d$ произвольным образом (например, $\mathbf{w} = \mathbf{0}$)

Повторять для каждого эпизода:

Инициализировать состояние и получить начальный вектор признаков $\mathbf{x}$
 $\mathbf{z} \leftarrow \mathbf{0}$ (d -мерный вектор)

$V_{old} \leftarrow 0$ (временная скалярная переменная)

Повторять для каждого шага эпизода:

Выбрать $A \sim \pi$

Предпринять действие A , наблюдать $R, \mathbf{x}'$ (вектор признаков следующего состояния)

$V \leftarrow \mathbf{w}^T \mathbf{x}$

$V' \leftarrow \mathbf{w}^T \mathbf{x}'$

$\delta \leftarrow R + \gamma V' - V$

$$\begin{aligned}
\mathbf{z} &\leftarrow \gamma \lambda \mathbf{z} + (1 - \alpha \gamma \lambda \mathbf{z}^T \mathbf{x}) \mathbf{x} \\
\mathbf{w} &\leftarrow \mathbf{w} + \alpha (\delta + V - V_{old}) \mathbf{z} - \alpha (V - V_{old}) \mathbf{x} \\
V_{old} &\leftarrow V \\
\mathbf{x} &\leftarrow \mathbf{x}' \\
&\text{пока не } \mathbf{x}' = 0 \text{ (сигнал о переходе в заключительное состояние)}
\end{aligned}$$

След приемлемости (12.11), используемый в истинно онлайн-овом $TD(\lambda)$, называется *голландским следом*, чтобы отличить его от следа (12.5), который используется в $TD(\lambda)$ и называется *накапливающимся*. В ранних работах часто использовался еще один вид следа, называемый *замещающим*, но он определен только для табличного случая или для бинарных признаков типа тех, что порождаются в результате плиточного кодирования. Замещающий след определяется поэлементно в зависимости от того, равен ли элемент вектора признаков 1 или 0:

$$z_{i,t} \doteq \begin{cases} 1 & \text{если } x_{i,t} = 1 \\ \gamma \lambda z_{i,t-1} & \text{в противном случае} \end{cases} \quad (12.12)$$

В настоящее время мы рассматриваем замещающие следы как грубую аппроксимацию голландских следов, которые повсеместно вытесняют их. Голландские следы обычно дают лучшие результаты, чем замещающие, и имеют более понятное теоретическое обоснование. Накапливающиеся следы остаются интересными для нелинейной аппроксимации функций, когда голландские следы неприменимы.

12.6. *ГОЛЛАНДСКИЕ СЛЕДЫ В ОБУЧЕНИИ МЕТОДАМИ МОНТЕ-КАРЛО

Хотя следы приемлемости исторически переплетаются с TD -обучением, на самом деле они не имеют с ним ничего общего. Следы приемлемости возникают даже при обучении методами Монте-Карло, что мы и увидим в этом разделе. Мы покажем, что линейный алгоритм МК (глава 9), рассматриваемый как прямое представление, можно использовать для вывода эквивалентного, но вычислительно более дешевого алгоритма с обратным представлением, в котором используются голландские следы. Это единственная эквивалентность прямых и обратных представлений, которая явно демонстрируется в этой книге. Данное рассуждение дает некое представление о доказательстве эквивалентности истинно онлайн-ового $TD(\lambda)$ и онлайн-ового λ -доходных алгоритмов, но гораздо проще.

В линейном варианте градиентного алгоритма предсказания Монте-Карло (стр. 243) производится следующая последовательность обновлений, по одному на каждом временном шаге эпизода:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha [G - \mathbf{w}_t^T \mathbf{x}_t] \mathbf{x}_t, \quad 0 \leq t < T. \quad (12.13)$$

Чтобы упростить пример, будем предполагать, что доход G состоит из единственного вознаграждения, полученного в конце эпизода (поэтому G и не снабжен нижним индексом), и что обесценивание не применяется. В таком случае обновление называют также правилом наименьших средних квадратов (LMS). Поскольку это алгоритм Монте-Карло, все обновления зависят от окончательного вознаграждения (дохода), поэтому их нельзя производить до завершения эпизода. Алгоритм МК является офлайновым, и мы не преследуем цели улучшить этот его аспект. Мы просто хотим найти его реализацию, обладающую вычислительными преимуществами. Мы по-прежнему обновляем вектор весов только в конце эпизода, но выполняем некоторые вычисления на каждом шаге, так чтобы в конце было меньше работы. Это дает более равномерное распределение вычислений – $O(d)$ на каждом шаге, – а также устраняет необходимость хранить векторы признаков на каждом шаге для последующего использования в конце эпизода. Вместо этого потребуется дополнительная память для векторного следа приемлемости, в котором будет храниться сводная информация обо всех встречавшихся векторах признаков. Этого будет достаточно, чтобы к концу эпизода эффективно повторить в точности то же самое общее обновление, которое достигается в результате последовательности обновлений Монте-Карло (12.13):

$$\begin{aligned} \mathbf{w}_T &= \mathbf{w}_{T-1} + \alpha(G - \mathbf{w}_{T-1}^\top \mathbf{x}_{T-1}) \mathbf{x}_{T-1} \\ &= \mathbf{w}_{T-1} + \alpha \mathbf{x}_{T-1} (-\mathbf{x}_{T-1}^\top \mathbf{w}_{T-1}) + \alpha G \mathbf{x}_{T-1} \\ &= (\mathbf{I} - \alpha \mathbf{x}_{T-1} \mathbf{x}_{T-1}^\top) \mathbf{w}_{T-1} + \alpha G \mathbf{x}_{T-1} \\ &= \mathbf{F}_{T-1} \mathbf{w}_{T-1} + \alpha G \mathbf{x}_{T-1}, \end{aligned}$$

где $\mathbf{F}_t \doteq \mathbf{I} - \alpha \mathbf{x}_t \mathbf{x}_t^\top$ – матрица *забывания*, или *затухания*. Теперь повторим это рекурсивно:

$$\begin{aligned} &= \mathbf{F}_{T-1} (\mathbf{F}_{T-2} \mathbf{w}_{T-2} + \alpha G \mathbf{x}_{T-2}) + \alpha G \mathbf{x}_{T-1} \\ &= \mathbf{F}_{T-1} \mathbf{F}_{T-2} \mathbf{w}_{T-2} + \alpha G (\mathbf{F}_{T-1} \mathbf{x}_{T-2} + \mathbf{x}_{T-1}) \\ &= \mathbf{F}_{T-1} \mathbf{F}_{T-2} (\mathbf{F}_{T-3} \mathbf{w}_{T-3} + \alpha G \mathbf{x}_{T-3}) + \alpha G (\mathbf{F}_{T-1} \mathbf{x}_{T-2} + \mathbf{x}_{T-1}) \\ &= \mathbf{F}_{T-1} \mathbf{F}_{T-2} \mathbf{F}_{T-3} \mathbf{w}_{T-3} + \alpha G (\mathbf{F}_{T-1} \mathbf{F}_{T-2} \mathbf{x}_{T-3}) + \mathbf{F}_{T-1} \mathbf{x}_{T-2} + \mathbf{x}_{T-1} \\ &\vdots \\ &= \underbrace{\mathbf{F}_{T-1} \mathbf{F}_{T-2} \cdots \mathbf{F}_0 \mathbf{w}_0}_{\mathbf{a}_{T-1}} + \alpha G \underbrace{\sum_{k=0}^{T-1} \mathbf{F}_{T-1} \mathbf{F}_{T-2} \cdots \mathbf{F}_{k+1} \mathbf{x}_k}_{\mathbf{z}_{T-1}} \\ &= \mathbf{a}_{T-1} + \alpha G \mathbf{z}_{T-1}, \end{aligned} \tag{12.14}$$

где $\mathbf{a}_{T-1}$ и $\mathbf{z}_{T-1}$ – значения в момент $T-1$ двух вспомогательных векторов, которые можно обновлять инкрементно, не зная G , со сложностью $O(d)$ на каждом временном шаге. Вектор $\mathbf{z}_t$ на самом деле является голландским следом приемлемости. Он инициализируется значением $\mathbf{z}_0 = \mathbf{x}_0$, а затем обновляется по следующему правилу:

$$\begin{aligned}
\mathbf{z}_t &\doteq \sum_{k=0}^t \mathbf{F}_t \mathbf{F}_{t-1} \cdots \mathbf{F}_{k+1} \mathbf{x}_k, \quad 1 \leq t < T \\
&= \sum_{k=0}^{t-1} \mathbf{F}_t \mathbf{F}_{t-1} \cdots \mathbf{F}_{k+1} \mathbf{x}_k + \mathbf{x}_t \\
&= \mathbf{F}_t \sum_{k=0}^{t-1} \mathbf{F}_{t-1} \cdots \mathbf{F}_{k+1} \mathbf{x}_k + \mathbf{x}_t \\
&= \mathbf{F}_t \mathbf{z}_{t-1} + \mathbf{x}_t \\
&= (\mathbf{I} - \alpha \mathbf{x}_t \mathbf{x}_t^\top) \mathbf{z}_{t-1} + \mathbf{x}_t \\
&= \mathbf{z}_{t-1} - \alpha \mathbf{x}_t \mathbf{x}_t^\top \mathbf{z}_{t-1} + \mathbf{x}_t \\
&= \mathbf{z}_{t-1} - \alpha (\mathbf{z}_{t-1}^\top \mathbf{x}_t) \mathbf{x}_t + \mathbf{x}_t \\
&= \mathbf{z}_{t-1} + (1 - \alpha \mathbf{z}_{t-1}^\top \mathbf{x}_t) \mathbf{x}_t,
\end{aligned}$$

т. е. является голландским следом для случая $\gamma\lambda = 1$ (см. уравнение 12.11). Вспомогательный вектор $\mathbf{a}_t$ инициализируется значением $\mathbf{a}_0 = \mathbf{w}_0$, а затем обновляется по следующему правилу:

$$\mathbf{a}_t \doteq \mathbf{F}_t \mathbf{F}_{t-1} \cdots \mathbf{F}_0 \mathbf{w}_0 = \mathbf{F}_t \mathbf{a}_{t-1} - \alpha \mathbf{x}_t \mathbf{x}_t^\top \mathbf{a}_{t-1}, \quad 1 \leq t < T.$$

Вспомогательные векторы $\mathbf{a}_t$ и $\mathbf{z}_t$ обновляются на каждом временном шаге $t < T$, а затем, в момент T , когда наблюдается G , используются в (12.14) для вычисления $\mathbf{w}_T$. Тем самым мы достигаем точно такого же окончательного результата, как в алгоритме МК/LMS, обладающем неудовлетворительными вычислительными свойствами (12.13), но на этот раз используем инкрементный алгоритм, сложность которого по времени и по памяти составляет $O(d)$ на каждом шаге. Это удивительный и интригующий результат, потому что понятие следа приемлемости (и голландского следа, в частности) возникло в постановке задачи, где никакого обучения на основе временных различий (TD) не было и в помине (в отличие от работы van Seijen and Sutton, 2014). Складывается впечатление, что следы приемлемости вообще не имеют прямого отношения к TD-обучению, они более фундаментальны. Необходимость в следах приемлемости, похоже, возникает всякий раз, как требуется эффективно обучиться давать долгосрочные прогнозы.

12.7. SARSA(λ)

В уже изложенные в этой главе идеи нужно внести совсем небольшие изменения, чтобы распространить следы приемлемости на методы вычисления ценности действий. Чтобы обучиться приближенным ценностям действий $\hat{q}(s, a, \mathbf{w})$, а не состояний $\hat{v}(s, \mathbf{w})$, необходимо использовать форму n -шагового дохода с ценностями действий из главы 10:

$$G_{t:t+n} \doteq R_{t+1} + \cdots + \gamma n^{-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}), \quad t+n < T,$$

где $G_{t:t+n} \doteq G_t$, если $t+n \geq T$. Применяя это определение, мы можем выписать форму усеченного λ -дохода для ценности действий, которая во всем остальном совпада-

Полный псевдокод Sarsa(λ) с линейной аппроксимацией функций, бинарными признаками и накапливающимися или замещающимися следами приведен во врезке ниже. В него включено несколько оптимизаций, возможных в частном случае бинарных признаков (принимающих значения «активен» (1) или «неактивен» (0)).

Пример 12.1. Следы в сеточном мире. Применение следов приемлемости может значительно повысить эффективность алгоритмов управления по сравнению с одношаговыми и даже n -шаговыми методами. Причина иллюстрируется приведенным ниже примером для сеточного мира.



На первом рисунке показан путь, выбранный агентом в одном эпизоде. Начальные оценки ценностей равны 0, а все вознаграждения нулевые, кроме положительного вознаграждения в целевой позиции, обозначенной буквой G. Стрелки на других рисунках показывают, ценности каких действий увеличивают различные алгоритмы по достижении цели и насколько. Одношаговый метод увеличит только ценность последнего действия, тогда как n -шаговый – ценности последних n действий, а метод следов приемлемости обновит ценности всех действий с начала эпизода, правда, в разной степени, затухающей с увеличением давности. Стратегия затухания часто оказывается наилучшей. ■

Sarsa(λ) с бинарными признаками и линейной аппроксимацией функций для оценивания $\mathbf{w}^T \mathbf{x} \approx q_\pi$ или q_*

Вход: функция $\mathcal{F}(s, a)$, возвращающая множество (индексов) активных признаков для пары s, a

Вход: стратегия π (если оценивается q_π)

Параметры алгоритма: размер шага $\alpha > 0$, скорость затухания следа $\lambda \in [0, 1]$

Инициализировать $\mathbf{w} = (w_1, \dots, w_d)^T \in \mathbb{R}^d$ (например, $\mathbf{w} = \mathbf{0}$), $\mathbf{z} = (z_1, \dots, z_d)^T \in \mathbb{R}^d$

Повторять для каждого эпизода:

Инициализировать S

Выбрать $A \sim \pi(\cdot | S)$ или ϵ -жадную относительно $\hat{q}(S, \cdot, \mathbf{w})$

$\mathbf{z} \leftarrow \mathbf{0}$

Повторять для каждого шага эпизода:

Предпринять действие A , наблюдать R, S'

$\delta \leftarrow R$

Повторять для i , принадлежащего $\mathcal{F}(S, A)$

$\delta \leftarrow \delta - w_i$

$z_i \leftarrow z_i + 1$

(накапливающиеся следы)

или $z_i \leftarrow 1$

(замещающие следы)

Если S' – заключительное состояние, то:

$\mathbf{w} \leftarrow \mathbf{w} + \alpha \delta \mathbf{z}$

Перейти к следующему эпизоду

Выбрать $A' \sim \pi(\cdot | S')$ или почти жадную относительно $\hat{q}(S', \cdot, \mathbf{w})$

Повторять для i , принадлежащего $\mathcal{F}(S', A')$: $\delta \leftarrow \delta + \gamma w_i$

$\mathbf{w} \leftarrow \mathbf{w} + \alpha \delta \mathbf{z}$

$\mathbf{z} \leftarrow \gamma \lambda \mathbf{z}$

$S \leftarrow S'; A \leftarrow A'$

Упражнение 12.6. Модифицируйте псевдокод алгоритма Sarsa(λ) так, чтобы использовать голландские следы (12.11) и никаких других особенностей истинно онлайн-алгоритма. Предполагайте, что аппроксимация функций линейна, а признаки бинарные. □

Пример 12.2. Применение Sarsa(λ) к задаче о машине на горе. На рис. 12.10 (слева) показаны результаты применения Sarsa(λ) к задаче о машине на горе, описанной в примере 10.1. Аппроксимация функций, выбор действий и детали окружающей среды точно такие же, как в главе 10, поэтому допустимо численное сравнение этих результатов с полученными в главе 10 для n -шагового Sarsa (на том же рисунке справа). Ранее мы изменяли длину обновления n , а в случае Sarsa(λ) изменяем параметр λ , играющий сходную роль. Подход Sarsa(λ), основанный на бутстрэппинге с затухающим следом, похоже, приводит к более эффективному обучению в этой задаче.

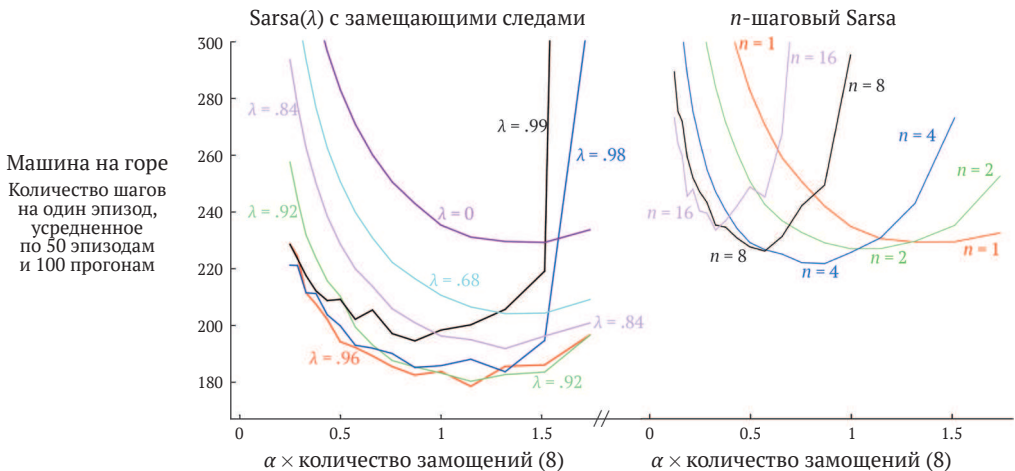


Рис. 12.10 ♦ Результаты работы двух алгоритмов для задачи о машине на горе – Sarsa(λ) с замещающими следами и n -шагового Sarsa (повторение рис. 10.4) – в виде функции от размера шага

Для ценности действий существует также вариант нашего идеального TD-метода, онлайн-ового λ -доходного алгоритма (раздел 12.4), и его эффективная реализация в виде истинно онлайн-ового TD(λ) (раздел 12.5). Все сказанное в разделе 12.4 проходит без изменений, если использовать форму n -шагового дохода с ценностями действий, приведенную в начале текущего раздела. Анализ, выполненный в разделах 12.5 и 12.6, также остается справедливым для ценностей действий; нужно только использовать векторы признаков состояние–действие $\mathbf{x}_t = \mathbf{x}(S_t, A_t)$ вместо векторов признаков состояний $\mathbf{x}_t = \mathbf{x}(S_t)$. Псевдокод получающегося эффективного алгоритма, называемого *истинно онлайн-овым Sarsa(λ)*, приведен во врезке ниже. На рис. 12.11 показаны результаты сравнения различных вариантов Sarsa(λ) для задачи о машине на горе. Наконец, существует усеченный вариант Sarsa(λ), называемый *прямым Sarsa(λ)* (van Seijen, 2016), который, похоже, особенно эффективен в качестве безмодельного метода управления, если использовать его в сочетании с многослойными нейронными сетями.

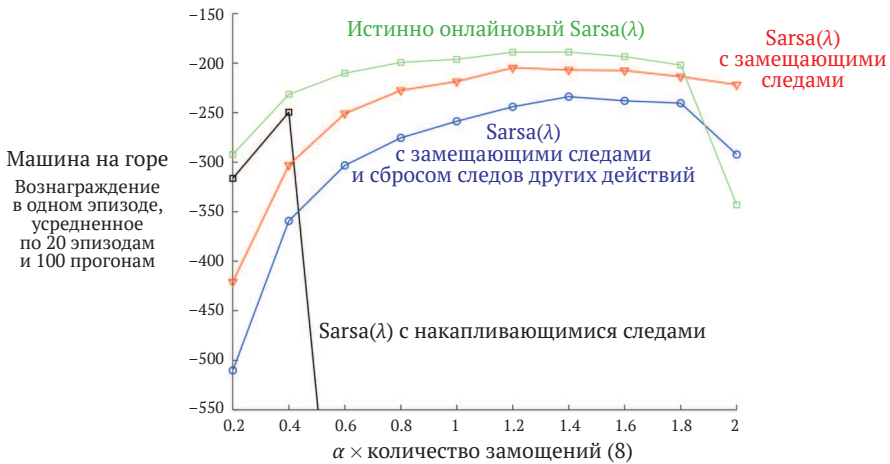


Рис. 12.11 ❖ Сравнение различных алгоритмов Sarsa(λ) для задачи о машине на горе. Истинно онлайн-овый Sarsa(λ) работал лучше, чем обычный Sarsa(λ) как с накапливающимися, так и с замещающими следами. Также включена версия Sarsa(λ) с замещающими следами, в которой на каждом временном шаге следы для состояния и невыбранных действий сбрасывались в 0

Истинно онлайн-овый Sarsa(λ) для оценивания $\mathbf{w}^\top \mathbf{x} \approx q_\pi$ или q_*

Вход: функция признаков $\mathbf{x}: \mathcal{S}^+ \times \mathcal{A} \rightarrow \mathbb{R}^d$ такая, что $\mathbf{x}(\text{terminal}, \cdot) = \mathbf{0}$

Вход: стратегия π (если оценивается q_π)

Параметры алгоритма: размер шага $\alpha > 0$, скорость затухания следа $\lambda \in [0, 1]$

Инициализировать $\mathbf{w} \in \mathbb{R}^d$ произвольным образом (например, $\mathbf{w} = \mathbf{0}$)

Повторять для каждого эпизода:

Инициализировать S

Выбрать действие $A \sim \pi(\cdot, S)$ или почти жадно из S с использованием $\mathbf{w}$

$$\mathbf{x} \leftarrow \mathbf{x}(S, A)$$

$$\mathbf{z} \leftarrow \mathbf{0}$$

$$Q_{old} \leftarrow 0$$

Повторять для каждого шага эпизода:

Предпринять действие A , наблюдать R, S'

Выбрать $A' \sim \pi(\cdot, S')$ или почти жадно из S' с использованием $\mathbf{w}$

$$\mathbf{x}' \leftarrow \mathbf{x}(S', A')$$

$$Q \leftarrow \mathbf{w}^T \mathbf{x}$$

$$Q' \leftarrow \mathbf{w}^T \mathbf{x}'$$

$$\delta \leftarrow R + \gamma Q' - Q$$

$$\mathbf{z} \leftarrow \gamma \lambda \mathbf{z} + (1 - \alpha \gamma \lambda \mathbf{z}^T \mathbf{x}) \mathbf{x}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha(\delta + Q - Q_{old})\mathbf{z} - \alpha(Q - Q_{old})\mathbf{x}$$

$$Q_{old} \leftarrow Q$$

$$\mathbf{x} \leftarrow \mathbf{x}'$$

$$A \leftarrow A'$$

пока S' не является заключительным состоянием

12.8. ПЕРЕМЕННЫЕ λ И γ

Вот мы и приближаемся к концу разработки фундаментальных алгоритмов ТД-обучения. Чтобы представить окончательные алгоритмы в самой общей форме, полезно не ограничиваться постоянным количеством шагов бутстрэппинга и коэффициентом обесценивания, а считать, что это функции, которые потенциально могут зависеть от состояния и действия. Иначе говоря, на каждом временном шаге мы будем иметь различные λ и γ , обозначаемые λ_t и γ_t . Теперь $\lambda: \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ – функция, отображающая пару состояние–действие в число из единичного отрезка и $\lambda_t \doteq \lambda(S_t, A_t)$. И аналогично $\gamma: \mathcal{S} \rightarrow [0, 1]$ – функция, отображающая состояние в число из единичного отрезка и $\gamma_t \doteq \gamma(S_t)$.

Введение функции завершения γ особенно важно, потому что это изменяет определение дохода, фундаментальной случайной величины, математическое ожидание которой мы и стремимся оценить. Теперь доход определяется в более общем виде:

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma_{t+1} G_{t+1} \\ &= R_{t+1} + \gamma_{t+1} R_{t+2} + \gamma_{t+1} \gamma_{t+2} R_{t+3} + \gamma_{t+1} \gamma_{t+2} \gamma_{t+3} R_{t+4} + \dots \\ &= \sum_{k=t}^{\infty} R_{k+1} \prod_{i=t+1}^k \gamma_i, \end{aligned} \tag{12.17}$$

где для гарантии конечности сумм мы требуем, чтобы $\prod_{k=t}^{\infty} \gamma_k = 0$ с вероятностью 1 для всех t . У этого определения есть удобная особенность – оно позволяет представить эпизодическую постановку и все алгоритмы для нее в терминах единого потока опыта, без специальных заключительных состояний, начальных распределений и моментов завершения. То, что раньше было заключительным состоянием, становится состоянием, в котором $\gamma(s) = 0$, а все переходы ведут в начальное распределение. Таким образом (и полагая $\gamma(\cdot)$ постоянной во всех остальных со-

стояниях) мы можем воспроизвести эпизодическую постановку как частный случай. Зависимость завершения от состояния допускает и другие сценарии предсказания, например *псевдозавершение*, когда мы хотим предсказать некоторую величину, не изменяя поток марковского процесса. Обесцененные доходы можно рассматривать как такую величину, и в этом случае зависящее от состояния завершение унифицирует эпизодическую постановку и непрерывную постановку с обесцениванием. (Непрерывная постановка без обесценивания по-прежнему требует отдельного рассмотрения.)

Обобщение на переменное количество шагов бутстрэппинга является не изменением в постановке задачи, как в случае обесценивания, а изменением в подходе к решению. Обобщение затрагивает λ -доходы для состояний и действий. Новый λ -доход для состояний можно определить рекуррентным соотношением

$$G_t^{\lambda s} \doteq R_{t+1} + \gamma_{t+1}((1 - \lambda_{t+1})\hat{v}(S_{t+1}, \mathbf{w}_t) + \lambda_{t+1}G_{t+1}^{\lambda s}), \quad (12.18)$$

в котором к верхнему индексу λ добавлено «s» как напоминание о том, что это доход на основе бутстрэппинга по ценностям состояний, а не доход на основе бутстрэппинга по ценностям действий, который ниже обозначен путем добавления «a» к верхнему индексу. Это выражение говорит, что λ -доход является первым вознаграждением, необесцененным и не подверженным влиянию бутстрэппинга, плюс, возможно, второй член в тех случаях, когда мы выполняем обесценивание в следующем состоянии (т. е. в зависимости от γ_{t+1} ; напомним, что эта величина равна нулю, если следующее состояние заключительное). В тех случаях, когда мы не завершаемся в следующем состоянии, имеется второй член, который сам разбивается на два случая в зависимости от степени бутстрэппинга в этом состоянии. Если бутстрэппинг присутствует, то этот член является оценкой ценности в данном состоянии, а если нет, то это λ -доход для следующего временного шага. Зависящий от действий λ -доход определяется либо как в Sarsa

$$G_t^{\lambda a} \doteq R_{t+1} + \gamma_{t+1}((1 - \lambda_{t+1})\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) + \lambda_{t+1}G_{t+1}^{\lambda a}), \quad (12.19)$$

либо как в Expected Sarsa

$$G_t^{\lambda a} \doteq R_{t+1} + \gamma_{t+1}((1 - \lambda_{t+1})\bar{V}(S_{t+1}) + \lambda_{t+1}G_{t+1}^{\lambda a}), \quad (12.20)$$

где (7.8) обобщается на аппроксимацию функций следующим образом:

$$\bar{V}_t(s) \doteq \sum_a \pi(a|s)\hat{q}(s, a, \mathbf{w}_t). \quad (12.21)$$

Упражнение 12.7. Обобщите все три приведенных выше рекуррентных соотношения на усеченные версии, определив $G_{t:h}^{\lambda s}$ и $G_{t:h}^{\lambda a}$. $\square$

12.9. СЛЕДЫ С РАЗДЕЛЕННОЙ СТРАТЕГИЕЙ И ПЕРЕМЕННЫМ УПРАВЛЕНИЕМ

Последний шаг – включить выборку по значимости. В отличие от n -шаговых методов, для полных неусеченных λ -доходов нет никаких практически полезных ситуаций, когда выборка по значимости производится вне целевого дохода. Вместо этого мы переходим непосредственно к обобщению на бутстрэппинг выборки по значимости на одном шаге с переменным управлением (раздел 7.4). В случае состояний наше окончательное определение λ -дохода обобщает формулу (12.18) по образцу (7.13):

$$G_t^{\lambda s} \doteq \rho_t(R_{t+1} + \gamma_{t+1}((1 - \lambda_{t+1})\hat{v}(S_{t+1}, \mathbf{w}_t) + \lambda_{t+1}G_{t+1}^{\lambda s}) + (1 - \rho_t)\hat{v}(S_t, \mathbf{w}_t)), \quad (12.22)$$

где $\rho_t = \pi(A_t|S_t)/b(A_t|S_t)$ – обычный коэффициент выборки по значимости на одном шаге. Как и в случае других вариантов дохода, встречавшихся в этой книге, усеченную версию данного дохода можно аппроксимировать просто в терминах сумм основанных на состоянии TD-ошибок

$$\delta_t^s \doteq R_{t+1} + \gamma_{t+1}\hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t), \quad (12.23)$$

в виде:

$$G_t^{\lambda s} \approx \hat{v}(S_t, \mathbf{w}_t) + \rho_t \sum_{k=t}^{\infty} \delta_k^s \prod_{i=t+1}^k \gamma_i \lambda_i \rho_i, \quad (12.24)$$

причем аппроксимация становится точной, если приближенная функция ценности не изменяется.

Упражнение 12.8. Докажите, что приближенное равенство (12.24) становится точным, если функция ценности не изменяется. Для сокращения записи рассмотрите случай $t = 0$ и воспользуйтесь нотацией $V_k \doteq \hat{v}(S_k, \mathbf{w})$. $\square$

Упражнение 12.9. Усеченный вариант общего дохода с разделенной стратегией обозначается $G_{t,h}^{\lambda s}$. Попробуйте угадать его правильное выражение, взяв за основу (12.24). $\square$

Приведенная выше форма λ -дохода (12.24) удобна для использования в обновлении прямого представления:

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha(G_t^{\lambda s} - \hat{v}(S_t, \mathbf{w}_t))\nabla \hat{v}(S_t, \mathbf{w}_t) \\ &\approx \mathbf{w}_t + \alpha\rho_t \left(\sum_{k=t}^{\infty} \delta_k^s \prod_{i=t+1}^k \gamma_i \lambda_i \rho_i \right) \nabla \hat{v}(S_t, \mathbf{w}_t), \end{aligned}$$

в котором опытный глаз увидит сходство с основанным на приемлемости TD-обновлением – произведение похоже на след приемлемости и умножается на TD-ошибки. Но это лишь один временной шаг прямого представления. А интересующая нас связь заключается в том, что обновление прямого представления,

просуммированное по времени, приближенно равно обновлению обратного представления, просуммированному по времени (это соотношение лишь приближенное, потому что мы снова игнорируем изменения функции ценности). Сумма обновлений прямого представления по времени равна

$$\begin{aligned}
 \sum_{t=1}^{\infty} (\mathbf{w}_{t+1} - \mathbf{w}_t) &\approx \sum_{t=1}^{\infty} \sum_{k=t}^{\infty} \alpha \rho_t \delta_k^s \nabla \hat{v}(S_t, \mathbf{w}_t) \prod_{i=t+1}^k \gamma_i \lambda_i \rho_i \\
 &= \sum_{k=1}^{\infty} \sum_{t=t}^k \alpha \rho_t \nabla \hat{v}(S_t, \mathbf{w}_t) \delta_k^s \prod_{i=t+1}^k \gamma_i \lambda_i \rho_i \\
 &\quad (\text{используется правило суммирования } \sum_{t=x}^y \sum_{k=t}^y = \sum_{k=x}^y \sum_{t=x}^k) \\
 &= \sum_{k=1}^{\infty} \alpha \delta_k^s \sum_{t=1}^k \rho_t \nabla \hat{v}(S_t, \mathbf{w}_t) \prod_{i=t+1}^k \gamma_i \lambda_i \rho_i,
 \end{aligned}$$

и эта сумма имела бы вид суммы TD-обновлений обратного представления, если бы всю внутреннюю сумму можно было записать и обновлять инкрементно, как след приемлемости. И сейчас мы покажем, как это можно сделать. То есть мы покажем, что если бы это выражение было следом в момент k , то мы могли бы обновить его, зная значение в момент $k - 1$, следующим образом:

$$\begin{aligned}
 \mathbf{z}_k &= \sum_{t=1}^k \rho_t \nabla \hat{v}(S_t, \mathbf{w}_t) \prod_{i=t+1}^k \gamma_i \lambda_i \rho_i \\
 &= \sum_{t=1}^{k-1} \rho_t \nabla \hat{v}(S_t, \mathbf{w}_t) \prod_{i=t+1}^k \gamma_i \lambda_i \rho_i + \rho_k \nabla \hat{v}(S_k, \mathbf{w}_k) \\
 &= \underbrace{\gamma_k \lambda_k \rho_k \sum_{t=1}^{k-1} \rho_t \nabla \hat{v}(S_t, \mathbf{w}_t) \prod_{i=t+1}^{k-1} \gamma_i \lambda_i \rho_i}_{\mathbf{z}_{k-1}} + \rho_k \nabla \hat{v}(S_k, \mathbf{w}_k) \\
 &= \rho_k (\gamma_k \lambda_k \mathbf{z}_{k-1} + \nabla \hat{v}(S_k, \mathbf{w}_k)),
 \end{aligned}$$

что после переименования индекса k в t является общей формой обновления накапливающегося следа для ценностей состояний:

$$\mathbf{z}_t \doteq \rho_t (\gamma_t \lambda_t \mathbf{z}_{t-1} + \nabla \hat{v}(S_t, \mathbf{w}_t)). \quad (12.25)$$

Этот след приемлемости в сочетании с обычным полугradientным правилом обновления параметров для TD(λ) (12.7) образует общий алгоритм TD(λ), применимый к данным как единой, так и разделенной стратегии. В случае единой стратегии алгоритм в точности совпадает с TD(λ), потому что ρ_t всегда равно 1 и (12.25) становится обычным накапливающимся следом (12.5) (обобщенным на переменные λ и γ). В случае с разделенной стратегией алгоритм часто работает хорошо, но, поскольку это полугradientный метод, устойчивая сходимости не гарантируется. В нескольких следующих разделах мы рассмотрим его обобщения, которые все-таки гарантируют устойчивость.

Очень похожая цепочка шагов приводит к выводу следов приемлемости с разделенной стратегией для методов вычисления ценности действий и соответствующих общих алгоритмов Sarsa(λ). Можно было бы начать с любой рекуррентной формы общего λ -дохода на основе действий (12.19) или (12.20), но последняя (Ex-

pected Sarsa) оказывается проще. Обобщив (12.20) на случай разделенной стратегии по образцу (7.14), мы получим

$$\begin{aligned} G_t^{\lambda a} &\doteq R_{t+1} + \gamma_{t+1} \left((1 - \lambda_{t+1}) \bar{V}_t(S_{t+1}) + \lambda_{t+1} [\rho_{t+1} G_{t+1}^{\lambda a} + \bar{V}_t(S_{t+1}) \right. \\ &\quad \left. - \rho_{t+1} \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) \right] \\ &= R_{t+1} + \gamma_{t+1} \left(\bar{V}_t(S_{t+1}) + \lambda_{t+1} \rho_{t+1} [G_{t+1}^{\lambda a} - \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t)] \right), \end{aligned} \quad (12.26)$$

где $\bar{V}_t(S_{t+1})$ определяется как в (12.21). И снова λ -доход можно приближенно записать в виде суммы TD-ошибок:

$$G_t^{\lambda a} \approx \hat{q}(S_t, A_t, \mathbf{w}_t) + \sum_{k=t}^{\infty} \delta_k^a \prod_{i=t+1}^k \gamma_i \lambda_i \rho_i, \quad (12.27)$$

в которой используется представление основанной на действиях TD-ошибки в виде математического ожидания:

$$\delta_t^a = R_{t+1} + \gamma_{t+1} \bar{V}_t(S_{t+1}) - \hat{q}(S_t, A_t, \mathbf{w}_t). \quad (12.28)$$

Как и раньше, аппроксимация становится точной, если приближенная функция ценности не изменяется.

Упражнение 12.10. Докажите, что приближенное равенство (12.27) становится точным, если функция ценности не изменяется. Для сокращения записи рассмотрите случай $t = 0$ и воспользуйтесь нотацией $Q_k \doteq \hat{q}(S_k, A_k, \mathbf{w})$. Указание: начните с выписывания δ_0^a и $G_0^{\lambda a}$, затем $G_0^{\lambda a} - Q_0$. $\square$

Упражнение 12.11. Усеченный вариант общего дохода с разделенной стратегией обозначается $G_{t:h}^{\lambda a}$. Попробуйте угадать его правильное выражение, взяв за основу (12.27). $\square$

По аналогии с тем, что мы проделали для случая состояний, можно записать обновление прямого представления, основанное на (12.27), преобразовать сумму обновлений, применив правило суммирования, и, наконец, прийти к следующей форме следа приемлемости для ценностей действий:

$$\mathbf{z}_t \doteq \gamma_t \lambda_t \rho_t \mathbf{z}_{t-1} + \nabla \hat{q}(S_t, A_t, \mathbf{w}_t). \quad (12.29)$$

Этот след приемлемости в сочетании с TD-ошибкой на основе математического ожидания (12.28) и обычным полугradientным правилом обновления параметров (12.7) образует элегантный и эффективный алгоритм Expected Sarsa(λ), применимый к данным как единой, так и разделенной стратегии. Вероятно, это лучший алгоритм такого типа в настоящее время (хотя, конечно, устойчивость не гарантируется, если он не комбинируется с каким-нибудь методом из следующих разделов). В случае единой стратегии с постоянными λ и γ и обычной TD-ошибкой пары состояние–действие (12.16) этот алгоритм совпадает с алгоритмом Sarsa(λ), описанным в разделе 12.7.

Упражнение 12.12. Детально проведите все намеченные выше шаги вывода (12.29) из (12.27). Начните с обновления (12.15), подставьте $G_t^{\lambda a}$ из (12.26) вместо G_t^{λ} , а затем выполните те же шаги, что привели к (12.25). $\square$

При $\lambda = 1$ эти алгоритмы оказываются тесно связаны с соответствующими алгоритмами Монте-Карло. Можно было бы ожидать, что для эпизодических задач и офлайн-обновления имеет место точная эквивалентность, но на самом деле связь тоньше и несколько слабее. При этих наиболее благоприятных условиях все равно поэпизодная эквивалентность обновлений отсутствует, а налицо только эквивалентность их математических ожиданий. Это не должно вызывать удивления, поскольку эти методы производят неотменяемые обновления по мере следования по траектории, тогда как настоящие методы Монте-Карло не стали бы производить никаких обновлений на траектории, если какое-либо действие на ней имеет нулевую вероятность при следовании целевой стратегии. В частности, все эти методы, даже при $\lambda = 1$, все равно выполняют бутстрэппинг в том смысле, что их цели зависят от текущих оценок ценностей, – просто эта зависимость исчезает в математическом ожидании ценности. Хорошо это или плохо на практике – другой вопрос. Недавно были предложены методы, достигающие точной эквивалентности (Sutton, Mahmood, Precup and van Hasselt, 2014). В них дополнительно требуется вектор «предварительных весов» (provisional weights), в нем запоминаются обновления, которые были произведены, но могут быть отменены (или усилены) в зависимости от предпринятых впоследствии действий. Варианты этих методов для состояний и пар состояний–действие называются PTD(λ) и PQ(λ) соответственно; буква «Р» является сокращением от «Provisional».

Практические последствия всех этих новых методов с разделенной стратегией пока не установлены. Несомненно, возникнут проблемы с высокой дисперсией, свойственные всем методам с разделенной стратегией, в которых используется выборка по значимости (раздел 11.9).

Если $\lambda < 1$, то все эти алгоритмы включают бутстрэппинг, и к ним относится смертельная триада (раздел 11.3), т. е. устойчивость можно гарантировать только для табличного случая, для агрегирования состояний и других ограниченных форм аппроксимации функций. Для линейной и более общей форм аппроксимации вектор параметров может расходиться, как в примерах из главы 11. Как было сказано там же, у проблемы обучения с разделенной стратегией есть две стороны. Следы приемлемости с разделенной стратегией эффективно решают лишь одну часть проблемы, корректируя ожидаемую ценность цели, но совсем никак не затрагивают вторую, связанную с распределением обновлений. Алгоритмические подходы к решению второй части проблемы обучения с разделенной стратегией с помощью следов приемлемости перечислены в разделе 12.11.

Упражнение 12.13. Как выглядят «голландский» и «замещающий» варианты следов приемлемости с разделенной стратегией для методов вычисления ценности состояний и действий? □

12.10. От $Q(\lambda)$ Уоткинса к $TREE-BACKUP(\lambda)$

С годами было предложено несколько методов, обобщающих Q -обучение на следы приемлемости. Первым из них является метод $Q(\lambda)$ Уоткинса, в котором следы приемлемости затухают, как обычно, при условии что выбирается жадное действие, а при выборе первого нежадного действия сразу сбрасываются в ноль. На рис. 12.12 показана диаграмма предшествующих состояний для этого метода. В главе 6 мы объединили Q -обучение и метод Expected Sarsa, получив вариант

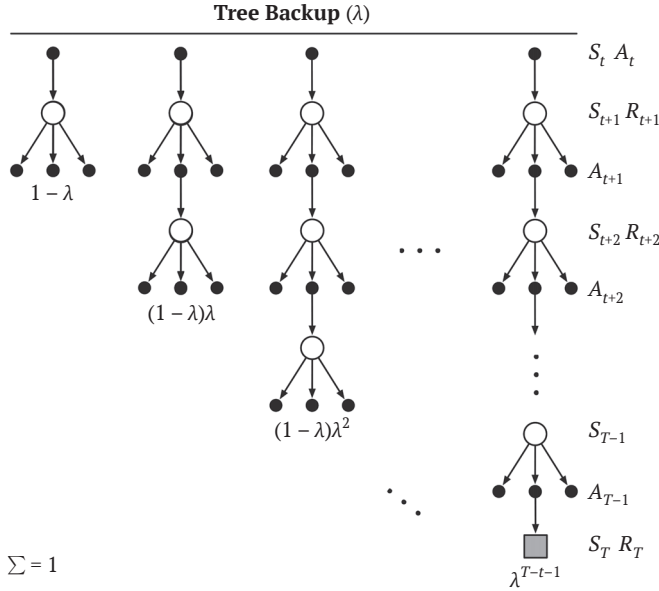


Рис. 12.13 ❖ Диаграмма предшествующих состояний для λ -варианта алгоритма обновления по дереву

Как всегда, эту формулу можно записать также в приближенной форме (игнорируя изменение приближенной функции ценности) в виде суммы TD-ошибок:

$$G_t^{\lambda a} \approx \hat{q}(S_t, A_t, \mathbf{w}_t) + \sum_{k=t}^{\infty} \delta_k^a \prod_{i=t+1}^k \gamma_i \lambda_i \pi(A_i | S_i),$$

воспользовавшись формой (12.28), основанной на действиях TD-ошибки в виде математического ожидания.

Выполнив те же шаги, что и в предыдущем разделе, мы придем к специальному обновлению следов приемлемости, включающему вероятности выбранных действий при целевой стратегии:

$$\mathbf{z}_t \doteq \gamma_i \lambda_i \pi(A_i | S_i) \mathbf{z}_{t-1} + \nabla \hat{q}(S_t, A_t, \mathbf{w}_t).$$

Присовокупив обычное правило обновления параметров (12.7), мы получаем определение алгоритма ТВ(λ). Как и для всех полугradientных алгоритмов, устойчивость ТВ(λ) не гарантируется при использовании данных разделенной стратегии и достаточно мощного аппроксиматора функций. Для получения таких гарантий ТВ(λ) следует объединить с одним из методов, представленных в следующем разделе.

**Упражнение 12.14.* Как можно обобщить алгоритм Double Expected Sarsa на следы приемлемости? □

12.11. УСТОЙЧИВЫЕ МЕТОДЫ С РАЗДЕЛЕННОЙ СТРАТЕГИЕЙ СО СЛЕДАМИ ПРИЕМЛЕМОСТИ

Было предложено несколько методов с использованием следов приемлемости, которые дают гарантии устойчивости при обучении с разделенной стратегией. Здесь мы представим четыре наиболее важных в стандартной нотации, принятой в этой книге, включая функции бутстрэппинга и обесценивания. Все они основаны на идеях градиентных или эмфатических TD-алгоритмов, изложенных в разделах 11.7 и 11.8 соответственно. Во всех алгоритмах предполагается линейная аппроксимация функций, хотя в литературе можно найти и обобщения на нелинейную аппроксимацию.

$GTD(\lambda)$ – алгоритм со следами приемлемости, аналогичный TDC – лучшему из двух градиентных TD-алгоритмов для предсказания ценности состояний, которые рассматривались в разделе 11.7. Его цель – обучить параметр $\mathbf{w}_t$, так чтобы $\hat{v}(s, \mathbf{w}_t) \doteq \mathbf{w}_t^\top \mathbf{x}(s) \approx v_\pi(s)$ даже на данных, полученных при следовании другой стратегии b . Уравнение обновления для него имеет вид:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \delta_t^s \mathbf{z}_t - \alpha \gamma_{t+1} (1 - \lambda_{t+1}) (\mathbf{z}_t^\top \mathbf{v}_t) \mathbf{x}_{t+1},$$

где δ_t^s , $\mathbf{z}_t$ и ρ_t определены обычным для ценностей состояний образом (см. (12.23) (12.25), (11.1)) и

$$\mathbf{v}_{t+1} \doteq \mathbf{v}_t + \beta \delta_t^s \mathbf{z}_t - \beta (\mathbf{v}_t^\top \mathbf{x}_t) \mathbf{x}_t, \quad (12.30)$$

где, как в разделе 11.7, $\mathbf{v} \in \mathbb{R}^d$ – вектор такой же размерности, как $\mathbf{w}$, инициализированный значением $\mathbf{v}_0 = \mathbf{0}$, а $\beta > 0$ – второй параметр размера шага.

$GQ(\lambda)$ – градиентный TD-алгоритм для оценки ценностей действий со следами приемлемости. Его цель – обучить параметр $\mathbf{w}_t$, так чтобы $\hat{q}(s, a, \mathbf{w}_t) \doteq \mathbf{w}_t^\top \mathbf{x}(s, a) \approx q_\pi(s, a)$ на данных разделенной стратегии. Если целевая стратегия ε -жадная или еще каким-то образом смещенная в сторону жадной стратегии относительно $\hat{q}$, то $GQ(\lambda)$ можно использовать в качестве алгоритма управления. Обновление для него имеет вид:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha \delta_t^a \mathbf{z}_t - \alpha \gamma_{t+1} (1 - \lambda_{t+1}) (\mathbf{z}_t^\top \mathbf{v}_t) \bar{\mathbf{x}}_{t+1},$$

где $\bar{\mathbf{x}}_t$ – средний вектор признаков для состояния S_t при целевой стратегии

$$\bar{\mathbf{x}}_t \doteq \sum_a \pi(a|S_t) \mathbf{x}(S_t, a);$$

δ_t^a – форма TD-ошибки в виде математического ожидания, которую можно записать в виде:

$$\delta_t^a \doteq R_{t+1} + \gamma_{t+1} \mathbf{w}_t^\top \bar{\mathbf{x}}_{t+1} - \mathbf{w}_t^\top \bar{\mathbf{x}}_t;$$

$\mathbf{z}_t$ определяется обычным для ценностей действий образом (12.29), а все остальное, включая обновление $\mathbf{v}_t$ (12.30), – как в $GTD(\lambda)$.

$HTD(\lambda)$ – гибридный алгоритм оценки ценности состояний, сочетающий в себе черты $GTD(\lambda)$ и $TD(\lambda)$. Самая привлекательная его особенность состоит в том, что он является строгим обобщением $TD(\lambda)$ на обучение с разделенной стратегией. Это

означает, что если поведенческая стратегия совпадает с целевой, то HTD(λ) превращается в TD(λ), чего нельзя сказать о GTD(λ). Привлекательно это потому, что TD(λ) часто работает быстрее GTD(λ), когда оба алгоритма сходятся, и TD(λ) требует задания только одного размера шага. HTD(λ) описывается следующими уравнениями:

$$\begin{aligned} \mathbf{w}_{t+1} &\doteq \mathbf{w}_t + \alpha \delta_t^s \mathbf{z}_t + \alpha ((\mathbf{z}_t - \mathbf{z}_t^b)^\top \mathbf{v}_t)(\mathbf{x}_t - \gamma_{t+1} \mathbf{x}_{t+1}), \\ \mathbf{v}_{t+1} &\doteq \mathbf{v}_t + \beta \delta_t^s \mathbf{z}_t - \beta (\mathbf{z}_t^{b\top} \mathbf{v}_t)(\mathbf{x}_t - \gamma_{t+1} \mathbf{x}_{t+1}) \quad \text{и} \quad \mathbf{v}_0 \doteq \mathbf{0}, \\ \mathbf{z}_t &\doteq \rho_t (\gamma_t \lambda_t \mathbf{z}_{t-1} + \mathbf{x}_t) \quad \text{и} \quad \mathbf{z}_{-1} \doteq \mathbf{0}, \\ \mathbf{z}_t^b &\doteq \gamma_t \lambda_t \mathbf{z}_{t-1}^b + \mathbf{x}_t \quad \text{и} \quad \mathbf{z}_{-1}^b \doteq \mathbf{0}, \end{aligned}$$

где $\beta > 0$ – второй размер шага. Помимо второго набора весов $\mathbf{v}_t$, в HTD(λ) имеет-ся также второй набор следов приемлемости, $\mathbf{z}_t^b$. Это традиционные накаплива-ющиеся следы приемлемости для поведенческой стратегии, которые становятся равны $\mathbf{z}_t$, если все ρ_t равны 1. В таком случае последний член в обновлении $\mathbf{w}_t$ об-ращается в 0, и все обновление сводится к обновлению для TD(λ).

Emphatic-TD(λ) – обобщение одношагового эмфатического TD-алгоритма (разде-лы 9.11 и 11.8) на следы приемлемости. Получающийся алгоритм сохраняет строгие гарантии сходимости для случая разделенной стратегии, допуская в то же время произвольный уровень бутстрэппинга, правда, ценой высокой дисперсии и потен-циально медленной сходимости. *Emphatic-TD(λ)* описывается уравнениями:

$$\begin{aligned} \mathbf{w}_{t+1} &\doteq \mathbf{w}_t + \alpha \delta_t \mathbf{z}_t, \\ \delta_t &\doteq R_{t+1} + \gamma_{t+1} \mathbf{w}_t^\top \mathbf{x}_{t+1} - \mathbf{w}_t^\top \mathbf{x}_t, \\ \mathbf{z}_t &\doteq \rho_t (\gamma_t \lambda_t \mathbf{z}_{t-1} + M_t \mathbf{x}_t) \quad \text{и} \quad \mathbf{z}_{-1} \doteq \mathbf{0}, \\ M_t &\doteq \lambda_t I_t + (1 - \lambda_t) F_t, \\ F_t &\doteq \rho_{t-1} \gamma_t F_{t-1} + I_t \quad \text{и} \quad F_0 \doteq i(S_0), \end{aligned}$$

где $M_t \geq 0$ – общая форма *значимости*, $F_t \geq 0$ называется *догоняющим следом* (followon trace), а $I_t \geq 0$ – *интересованность*, определенная в разделе 11.8. Заметим, что M_t , как и δ_t , на самом деле не занимает памяти. Эту переменную можно исклю-чить из алгоритма, подставив ее определение в уравнение следа приемлемости. Псевдокод и программа, реализующая истинно онлайн-вариант алгоритма *Emphatic-TD(λ)*, имеются в сети (Sutton, 2015b).

В случае единой стратегии ($\rho_t = 1$ для всех t) *Emphatic-TD(λ)* похож на традици-онный TD(λ), но все же заметно отличается от него. На самом деле гарантирует-ся, что *Emphatic-TD(λ)* сходится для всех зависящих от состояния функций λ , чего нельзя сказать о TD(λ). TD(λ) гарантированно сходится только для всех постоян-ных λ . См. контрпример Ю (Yu) в работе (Ghiassian, Rafiee, and Sutton, 2016).

12.12. ВОПРОСЫ РЕАЛИЗАЦИИ

Может показаться, что табличные методы со следами приемлемости гораздо сложнее одношаговых методов. При наивной реализации для каждого состоя-ния (или пары состояние–действие) на каждом временном шаге потребовалось бы обновлять как оценку ценности, так и след приемлемости. Это не проблема для параллельных компьютеров с SIMD-архитектурой или вероятных реализаций

искусственных нейронных сетей (ИНС), но становится таковой, если реализация осуществляется на традиционных последовательных компьютерах. По счастью, для типичных значений λ и γ следы приемлемости почти всех состояний почти всегда близки к нулю. Лишь для недавно посещенных состояний следы значительно больше нуля, и только эти немногочисленные состояния необходимо обновлять, чтобы получить хорошую аппроксимацию описанных алгоритмов.

Таким образом, при практической реализации на обычных компьютерах можно запоминать и обновлять лишь немногие следы, существенно большие нуля. Благодаря этому приему вычислительные затраты на использование следов в табличных методах, как правило, лишь в несколько раз превышают затраты в одно-разовом методе. Во сколько именно, конечно, зависит от λ и γ и от стоимости других вычислений. Отметим, что табличный случай в некотором смысле является наихудшим с точки зрения вычислительной сложности следов приемлемости. При использовании аппроксимации функций преимущества отказа от использования следов, вообще говоря, снижаются. Например, если используются ИНС и обратное распространение, то применение следов приемлемости лишь вдвое увеличивает объем потребляемой памяти и вычислений на каждом шаге. Усеченные λ -доходные методы (раздел 12.3) могут быть вычислительно эффективны на традиционных компьютерах, хотя всегда требуют дополнительной памяти.

12.13. Выводы

Следы приемлемости в сочетании с TD-ошибками дают эффективный и инкрементный способ выбора между методами Монте-Карло и TD-методами. Ту же задачу решают и n -шаговые методы из главы 7, но методы на основе следов приемлемости более общие, зачастую быстрее обучаются и предлагают различные компромиссы в части вычислительной сложности. Эта глава представляет собой введение в разрабатываемые в настоящее время теоретические основы применения следов приемлемости для обучения с единой и разделенной стратегией с переменными параметрами бутстрэппинга и обесценивания. Одним из аспектов этой элегантной теории являются истинно онлайн-методы, которые точно воспроизводят поведение дорогостоящих идеальных методов, сохраняя в то же время вычислительную эффективность традиционных TD-методов. Другой аспект – возможность автоматического преобразования интуитивно понятных методов прямого представления в более эффективные инкрементные алгоритмы обратного представления. Мы проиллюстрировали эту общую идею на примере логического вывода, который начался классическим дорогостоящим алгоритмом Монте-Карло, а закончился дешевой инкрементной реализацией без TD, в которой использовались те же инновационные следы приемлемости, что и в истинных онлайн-методах.

Как было отмечено в главе 5, методы Монте-Карло могут иметь преимущества в немарковских задачах, потому что не выполняют бутстрэппинга. Поскольку следы приемлемости делают TD-методы больше похожими на методы Монте-Карло, они также обладают преимуществами в этих случаях. Если имеет смысл предпочесть TD-методы за другие их достоинства, но задача хотя бы частично немарковская, то показано применение метода на основе следов приемлемости. Следы приемлемости – первая линия обороны как от надолго отложенных вознаграждений, так и от немарковских задач.

Варьируя λ , мы можем поместить методы на основе следов приемлемости в любое место континуума, простирающегося от методов Монте-Карло до TD-методов. Так куда же мы их поместим? На этот вопрос еще нет теоретического ответа, но начинает вырисовываться ясный эмпирический ответ. В задачах, где каждый эпизод состоит из большого числа шагов или период полужизни обесценивания насчитывает много шагов, применение следов приемлемости, похоже, дает значительно лучшие результаты (см. рис. 12.14). С другой стороны, если следы настолько длинные, что получается чистый метод Монте-Карло или близкий к нему, то качество резко снижается. По-видимому, наилучший выбор – что-то среднее. Следы приемлемости нужно использовать, чтобы приблизиться к методам Монте-Карло, но не слишком близко. Возможно, в будущем мы научимся более точно управлять компромиссом между TD-методами и методами Монте-Карло с помощью переменного параметра λ , но пока не ясно, как сделать это надежно и с пользой.

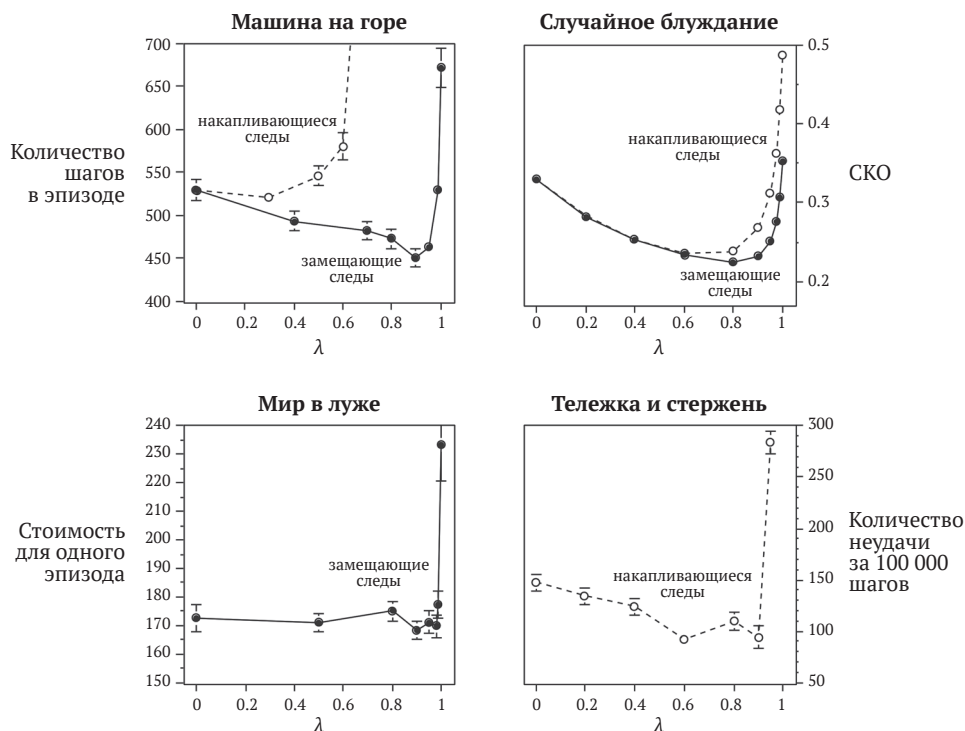


Рис. 12.14 ❖ Влияние на производительность обучения с подкреплением в четырех тестовых задачах. Во всех случаях наилучшая производительность (нижнее число на графике) наблюдалась при некотором промежуточном значении λ . Приложения на двух левых рисунках – простые непрерывные задачи управления, решаемые с помощью алгоритма $Sarsa(\lambda)$ и плиточного кодирования, с замещающими или накапливающимися следами (Sutton, 1996). На правом верхнем рисунке показано оценивание стратегии с помощью алгоритма $TD(\lambda)$ в задаче о случайном блуждании (Singh and Sutton, 1996). На правом нижнем рисунке показаны непубликовавшиеся данные для задачи о балансировании стержня (пример 3.4) из старой работы (Sutton, 1984)

В методах со следами приемлемости объем вычислений больше, чем в одношаговых, но за это мы получаем значительно более быстрое обучение, особенно в случаях, когда вознаграждение откладывается на много шагов. Поэтому следы приемлемости имеет смысл использовать, когда данных мало и их нельзя обработать повторно, как часто бывает в онлайн-овых приложениях. С другой стороны, в офлайн-овых приложениях, когда данные можно без труда сгенерировать, быть может, с помощью недорогого имитационного моделирования, использование следов приемлемости часто не окупается. В таких случаях наша задача – не получить как можно больше от ограниченного объема данных, а просто обработать как можно больше данных максимально быстро. Ускорение обработки одного элемента данных за счет следов обычно не окупает роста вычислительных затрат, так что лучше отдать предпочтение одношаговым методам.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

Следы приемлемости вошли в обучение с подкреплением благодаря плодотворным идеям Клопфа (Klopf, 1972). Наш подход к этой теме основан на работе Клопфа (Sutton, 1978a, 1978b, 1978c; Barto and Sutton, 1981a, 1981b; Sutton and Barto, 1981a; Barto, Sutton, and Anderson, 1983; Sutton, 1984). Возможно, мы были первыми, кто ввел в употребление термин «следы приемлемости» (Sutton and Barto, 1981a). Идея о том, что стимулы продуцируют эффекты последствия в нервной системе, важные для обучения, очень стара (см. главу 14). Из самых ранних применений следов приемлемости отметим методы типа исполнитель–критик, рассматриваемые в главе 13 (Barto, Sutton, and Anderson, 1983; Sutton, 1984).

12.1 Составные обновления (compound update) в первом издании этой книги назывались «complex backup».

λ -доход и его свойства уменьшения ошибки были введены в работе Watkins (1989) и затем получили развитие в работе Jaakkola, Jordan, and Singh (1994). Результаты о случайном блуждании в этом и следующем разделах публикуются впервые, как и термины «прямое представление» и «обратное представление». Понятие λ -доходного алгоритма было введено в первом издании книги. Представленный здесь уточненный подход разработан совместно с Хармом ван Сейеном (см., например, van Seijen and Sutton, 2014).

12.2 Метод TD(λ) с накапливающимися следами впервые описан в работе Sutton (1988, 1984). Сходимость в среднем доказана в работе Dayan (1992), а с вероятностью 1 – многими авторами, включая Peng (1993), Dayan and Sejnowski (1994), Tsitsiklis (1994) и Gurvits, Lin, and Hanson (1994). Верхняя граница ошибки асимптотического зависящего от λ решения линейного TD(λ) опубликована в статье Tsitsiklis and Van Roy (1997).

12.3 Усеченные TD-методы разработаны в Cichosz (1995) и van Seijen (2016).

12.4 Идея пересчета обновлений тщательно разработана ван Сейеном, первоначально под названием «обучение с наилучшим соответствием» («best-match learning») (van Seijen, 2011; van Seijen, Whiteson, van Hasselt, and Weiring, 2011).

12.5 Истинно онлайн-овый алгоритм TD(λ) принадлежит Харму ван Сейену (van Seijen and Sutton, 2014; van Seijen et al., 2016), хотя некоторые его ключевые

идеи были независимо открыты Хадо ван Хассельтом (частное сообщение). Название «голландские следы» – знак признания заслуг этих ученых. Замечающие следы описаны в работе Singh and Sutton (1996).

12.6 Материал этого раздела взят из работы van Hasselt and Sutton (2015).

12.7 Алгоритм Sarsa(λ) с накапливающимися следами впервые был исследован как метод управления в работах Rummery and Niranjan (1994), Rummery (1995). Истинный онлайн-версия Sarsa(λ) впервые описан в работе van Seijen and Sutton (2014). Алгоритм, приведенный на стр. 359, взят из работы van Seijen et al. (2016) и немного адаптирован. Результаты для задачи о машине на горе подготовлены специально для этой книги, за исключением рис. 12.11, заимствованного из статьи van Seijen and Sutton (2014).

12.8 Быть может, первым опубликованным обсуждением переменного параметра λ стала работа Уоткинса (Watkins, 1989), который отметил, что обрубание последовательности обновлений (рис. 12.12) при выборе нежадного действия в его алгоритме $Q(\lambda)$ можно было бы реализовать, временно положив λ равным 0.

Переменный параметр λ был введен в первом издании этой книги. Идея о переменном γ корнями уходит в работу об опциях (Sutton, Precup, and Singh, 1999) и предшествующую ей (Sutton, 1995a), а явно была оформлена в статье об алгоритме GQ(λ) (Maei and Sutton, 2010), где также приведены некоторые рекуррентные формы λ -дохода.

Другой подход к переменному λ развит в статье Yu (2012).

12.9 Следы приемлемости с разделенной стратегией введены в работе Precup et al. (2000, 2001), а затем эта идея получила развитие в работах Bertsekas and Yu (2009), Maei (2011), Maei and Sutton (2010), Yu (2012) и Sutton, Mahmood, Precup, and van Hasselt (2014). В частности, в последней работе описано мощное прямое представление для TD-методов с разделенной стратегией с зависящими от состояния λ и γ в общем случае. Приведенная здесь трактовка, вероятно, новая.

Этот раздел заканчивается элегантным алгоритмом Expected Sarsa(λ). Несмотря на свою естественность, этот алгоритм, насколько нам известно, ранее не был описан в литературе и не тестировался.

12.10 Алгоритм Уоткинса $Q(\lambda)$ описан в работе Watkins (1989). Сходимость его табличной эпизодической онлайн-версии доказана в работе Munos, Stepleton, Harutyunyan, and Bellemare (2016). Альтернативные алгоритмы $Q(\lambda)$ предложены в работах Peng and Williams (1994, 1996) и Sutton, Mahmood, Precup, and van Hasselt (2014). Алгоритм Tree-Backup(λ) приведен в работе Precup, Sutton, and Singh (2000).

12.11 Алгоритм GTD(λ) описан в работе Maei (2011), GQ(λ) – в работе Maei and Sutton (2010), а HTD(λ) – в работе White and White (2016) на основе одношагового алгоритма HTD, предложенного в статье Hackman (2012). Последние достижения в теории градиентных TD-методов принадлежат Ю (Yu, 2017). Алгоритм Emphatic-TD(λ) предложен в работе Sutton, Mahmood, а его устойчивость доказана в работе White (2016). В работах Yu (2015, 2016) доказана его сходимость, а дальнейшее развитие алгоритм получил в работах Hallak et al. (2015, 2016).

Глава 13

Методы градиента стратегии

В этой главе мы рассмотрим нечто новенькое. До сих пор почти все рассмотренные в этой книге методы были *методами ценности действий*; они обучались ценностям действий, а затем выбирали действия, исходя из оценок их ценности¹; принятые в этих методах стратегии вообще были бы невозможны без оценки ценности действий. В этой главе мы рассмотрим другие методы – они обучаются параметрической стратегии, которая может выбирать действия, вообще не прибегая к функции ценности. Функция ценности все же может использоваться для обучения параметров стратегии, но необязательно для выбора действия. Мы будем обозначать $\theta \in \mathbb{R}^d$ вектор параметров стратегии. Таким образом, мы записываем в виде $\pi(a|s, \theta) = \Pr\{A_t = a | S_t = s, \theta_t = \theta\}$ вероятность того, что действие a выбрано в момент t , при условии что в момент t окружающая среда находится в состоянии s , а параметр равен θ . Если в методе также используется обученная функция ценности, то ее вектор весов обозначается $\mathbf{w} \in \mathbb{R}^d$, как обычно в $\hat{v}(s, \mathbf{w})$.

В этой главе рассматриваются методы обучения параметра стратегии, основанные на градиенте некоторой скалярной меры качества $J(\theta)$ по параметру стратегии. Цель этих методов – максимизировать качество, поэтому производимые в них обновления аппроксимируют градиентный *подъем* по J :

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}, \quad (13.1)$$

где $\widehat{\nabla J(\theta_t)} \in \mathbb{R}^d$ – стохастическая оценка, математическое ожидание которой аппроксимирует градиент меры качества по ее аргументу θ_t . Все методы, которые следуют этой общей схеме, мы называем *методами градиента стратегии* – вне зависимости от того, обучается в них также приближенная функция ценности или нет. Методы, которые обучаются аппроксимировать одновременно стратегию и функции ценности, часто называют *методами исполнитель–критик*, причем под «исполнителем» понимается обученная стратегия, а под «критиком» – обученная функция ценности, обычно функция ценности состояний. Сначала будет рассмотрен эпизодический случай, в котором качество определено как ценность начального состояния при следовании параметрической стратегии, а затем мы

¹ Единственное исключение – градиентные алгоритмы бандита в разделе 2.8. На самом деле в том разделе, для случая бандита с одним состоянием, описаны многие из тех шагов, которые мы рассматриваем в этой главе для полных МППР. Для лучшего понимания этой главы мы рекомендуем еще раз перечитать раздел 2.8.

перейдем к непрерывному случаю, когда качество определено как среднее вознаграждение, как в разделе 10.3. В конце главы мы сможем выразить алгоритмы для обоих случаев в очень похожих терминах.

13.1. АППРОКСИМАЦИЯ СТРАТЕГИИ И ЕЕ ПРЕИМУЩЕСТВА

В методах градиента стратегии допустима параметризация стратегии любым способом, лишь бы функция $\pi(a|s, \theta)$ была дифференцируема по своим параметрам, т. е. при условии, что $\nabla \pi(a|s, \theta)$ (вектор-столбец, состоящий из частных производных $\pi(a|s, \theta)$ по элементам θ) существует и конечен для всех $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$, $\theta \in \mathbb{R}^d$. На практике, чтобы поощрить исследование, мы обычно требуем, чтобы стратегия никогда не становилась детерминированной (т. е. $\pi(a|s, \theta) \in (0, 1)$ для всех s, a, θ). В этом разделе мы познакомимся с самой распространенной параметризацией дискретного пространства действий и отметим преимущества, которыми она обладает по сравнению с методами на основе оценки ценности действий. Методы на основе стратегии также предлагают полезные способы работы с непрерывными пространствами действий, как будет описано в разделе 13.7.

Если пространство действий дискретно и не слишком велико, то естественная и популярная форма параметризации состоит в том, чтобы назначить параметрические числовые предпочтения $h(s, a, \theta) \in \mathbb{R}$ каждой паре состояние–действие. Действия с высокими предпочтениями в каждом состоянии выбираются с наибольшей вероятностью, например согласно экспоненциальному распределению softmax:

$$\pi(a|s, \theta) \doteq \frac{e^{h(s, a, \theta)}}{\sum_b e^{h(s, b, \theta)}}, \quad (13.2)$$

где $e \approx 2.71828$ – основание натуральных логарифмов. Заметим, что знаменатель выбран так, чтобы сумма всех вероятностей действий в каждом состоянии была равна 1. Такую параметризацию стратегии мы называем *softmax по предпочтениям действий*.

Сами предпочтения действий можно параметризовать произвольно. Например, предпочтения могут вычисляться искусственной нейронной сетью (ИНС), когда θ – вектор весов всех связей сети (так это делается в системе AlphaGo, описанной в разделе 16.6). Возможно также, что предпочтения просто линейно зависят от признаков

$$h(s, a, \theta) = \theta^\top \mathbf{x}(s, a), \quad (13.3)$$

где векторы признаков $\mathbf{x}(s, a) \in \mathbb{R}^d$ конструируются любым из методов, описанных в главе 9.

Одно из преимуществ параметризации стратегий в соответствии с softmax по предпочтениям действий состоит в том, что приближенная стратегия может приближаться к детерминированной, тогда как в случае ε -жадного выбора действий согласно их ценности всегда существует вероятность ε выбора случайного действия. Конечно, можно было бы производить выборку из распределения softmax, основанного на ценностях действий, но уже одно это не позволило бы стратегии приблизиться к детерминированной. Вместо этого оценки ценностей действий сходились бы к своим истинным значениям, которые отличались бы на конеч-

ную величину, что означало бы конкретные вероятности, отличные от 0 и 1. Если бы распределение softmax включало температурный параметр, то температуру можно было бы снижать со временем, чтобы приблизиться к детерминированности, но на практике было бы трудно выбрать кривую снижения и даже начальную температуру, не имея дополнительной априорной информации об истинных ценностях действий. Предпочтения действий отличаются, поскольку не стремятся к конкретным значениям, а управляются так, чтобы породить оптимальную стохастическую стратегию. Если оптимальная стратегия детерминирована, то предпочтения оптимальных действий будут бесконечно выше, чем у всех неоптимальных действий (если параметризация это допускает).

Второе преимущество параметризации стратегий в соответствии с softmax по предпочтениям действий состоит в том, что это позволяет выбирать действия с произвольными вероятностями. В задачах, где применяется аппроксимация функций, наилучшая приближенная стратегия может быть стохастической. Например, в карточных играх с неполной информацией оптимальная стратегия часто заключается в том, чтобы делать две разные вещи с определенными вероятностями, как, например, при блефе в покере. В методах оценки ценности действий нет естественного способа находить стохастические оптимальные стратегии, тогда как методы аппроксимации стратегий могут это делать, как показано в примере 13.1.

Пример 13.1. Короткий коридор с переставленными действиями

Рассмотрим короткий коридор в сеточном мире, показанный на вставке в рисунок ниже. Как обычно, за каждый шаг начисляется вознаграждение -1 . В каждом из трех незаключительных состояний имеется только два действия, right (направо) и left (налево). Последствия этих действий в первом и третьем состояниях определены как обычно (left в первом состоянии оставляет на месте), но во втором состоянии они переставлены местами, т. е. right приводит к перемещению влево, left – вправо. Задача трудная, потому что все состояния выглядят одинаково при аппроксимации функции. В частности, мы определяем $x(s, \text{right}) = [1, 0]^T$ и $x(s, \text{left}) = [0, 1]^T$ для всех s . Метод оценки ценности действий с ε -жадным выбором действий вынужден выбирать всего между двумя стратегиями: выбирать right с высокой вероятностью $1 - \varepsilon/2$ на всех шагах или выбирать left с такой же высокой вероятностью на всех шагах. Если $\varepsilon = 0.1$, то эти стратегии достигают ценности (начального состояния), меньшей 44 и 82 соответственно, как показано на графике. Но метод может добиться гораздо лучших результатов, если имеет возможность обучиться вероятности, с которой следует выбирать действие right. Наилучшая вероятность приблизительно равна 0.59, при этом достигается ценность, примерно равная -11.6 .



Быть может, самое простое преимущество параметризации стратегии над параметризацией функций ценности действий заключается в том, что стратегия может оказаться более простой для аппроксимации функцией. Задачи варьируются по сложности стратегий и функций ценности действий. Иногда функция ценности действий проще, так что аппроксимировать ее легче. А иногда проще стратегия – и тогда метод, основанный на стратегии, будет обучаться быстрее и найдет превосходящую асимптотическую стратегию (как в Тетрисе, см. Şimşek, Algorta, and Kothiyal, 2016).

Наконец, отметим, что выбор параметризации стратегии иногда является хорошим способом включить априорную информацию о желательной форме стратегии в систему обучения с подкреплением. Часто это главная причина для использования метода обучения, основанного на стратегии.

Упражнение 13.1. Воспользуйтесь своими знаниями о сеточном мире и его динамике, чтобы найти точное символьное выражение для оптимальной вероятности выбора действия `right` в примере 13.1. □

13.2. ТЕОРЕМА О ГРАДИЕНТЕ СТРАТЕГИИ

Помимо практических преимуществ параметризации стратегии над ε -жадным выбором действий, существует еще важное теоретическое преимущество. При непрерывной параметризации стратегии вероятности действий являются гладкой функцией от обученного параметра, тогда как при ε -жадном выборе вероятности могут резко изменяться при сколь угодно малом изменении оценок ценности действий, если это изменение приводит к выбору другого действия, имеющего максимальную ценность. Во многом из-за этого для методов на основе градиента стратегии имеются более сильные гарантии сходимости, чем для методов на основе ценности действий. В частности, именно непрерывность зависимости стратегии от параметров и позволяет использовать в методах градиента стратегии аппроксимацию градиентного подъема (13.1).

В эпизодическом и непрерывном случаях мера качества $J(\theta)$ определяется по-разному, поэтому и рассматривать их нужно по отдельности. Тем не менее мы попытаемся изложить оба случая единообразно и разработаем такую систему обозначений, чтобы главные теоретические результаты можно было описать одной системой уравнений.

В этом разделе рассматривается эпизодический случай, для которого мера качества определяется как ценность стартового состояния эпизода. Нотацию можно упростить без ограничения общности, предположив, что каждый эпизод начинается в некотором (неслучайном) состоянии s_0 . Тогда в эпизодическом случае мера качества определяется так:

$$J(\theta) \doteq v_{\pi_\theta}(s_0), \quad (13.4)$$

где v_{π_θ} – истинная функция ценности для π_θ – стратегии, определяемой параметром θ . Начиная с этого места, мы будем предполагать, что в эпизодическом случае обесценивание отсутствует ($\gamma = 1$), хотя для полноты картины предусматриваем возможность обесценивания в алгоритмах, приведенных во врезках.

Если применяется аппроксимация функций, то может показаться трудным изменить параметр стратегии, так чтобы гарантированно добиться улучшения. Проблема в том, что качество зависит как от выбора действий, так и от распределения состояний, в которых этот выбор производится, и что параметр зависит и от того, и от другого. Если известно состояние, то вычислить влияние параметра стратегии на действия, а значит, и на вознаграждение, довольно просто, зная, как устроена параметризация. Но влияние стратегии на распределение состояний – это функция окружающей среды, которая обычно неизвестна. Как можно оценить градиент качества по параметру стратегии, если градиент зависит от неизвестного влияния изменений стратегии на распределение состояний?

Доказательство теоремы о градиенте стратегии (эпизодический случай)

Применяя элементарные факты из математического анализа и переупорядочивая члены, мы можем без труда доказать теорему о градиенте стратегии. Чтобы не усложнять обозначения, будем неявно считать, что π всегда является функцией от θ и что все градиенты берутся по θ . Сначала отметим, что градиент функции ценности состояний можно записать в терминах функции ценности действий следующим образом:

$$\begin{aligned}
 \nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s, a) \right] \quad \text{для всех } s \in S & (\text{упражнение 3.18}) \\
 &= \sum_a \left[\nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \nabla q_\pi(s, a) \right] & (\text{правило произведения из математического анализа}) \\
 &= \sum_a \left[\nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \nabla \sum_{s', r} p(s', r|s, a) (r + v_\pi(s')) \right] & (\text{упражнение 3.19 и уравнение 3.2}) \\
 &= \sum_a \left[\nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \sum_{s'} p(s'|s, a) \nabla v_\pi(s') \right] & (\text{уравнение 3.4}) \\
 &= \sum_a \left[\nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \sum_{s'} p(s'|s, a) \right. & (\text{раскрытие}) \\
 &\quad \left. \sum_{a'} \left[\nabla \pi(a'|s') q_\pi(s', a') + \pi(a'|s') \sum_{s''} p(s''|s', a') \nabla v_\pi(s'') \right] \right] \\
 &= \sum_{x \in S} \sum_{k=0}^{\infty} \Pr(s \rightarrow x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x, a),
 \end{aligned}$$

где $\Pr(s \rightarrow x, k, \pi)$ – вероятность перехода из состояния s в состояние x за k шагов при следовании стратегии π . Отсюда сразу следует, что

$$\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_{\pi}(s_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \rightarrow s, k, \pi) \right) \sum_a \nabla \pi(a|s) q_{\pi}(s, a) \\
&= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_{\pi}(s, a) && \text{(врезка на стр. 240)} \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla \pi(a|s) q_{\pi}(s, a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_{\pi}(s, a) && \text{(уравнение 9.3)} \\
&\propto \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_{\pi}(s, a). && \text{ЧТД}
\end{aligned}$$

По счастью, для этой проблемы существует исчерпывающий теоретический ответ в форме *теоремы о градиенте стратегии*, которая дает аналитическое выражение градиента меры качества по параметру стратегии (это именно то, что мы должны аппроксимировать для применения градиентного спуска (13.1)), не содержащее производной распределения состояний. Теорема о градиенте стратегии для эпизодического случая утверждает, что

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_{\pi}(s, a) \nabla \pi(a|s, \boldsymbol{\theta}), \quad (13.5)$$

где градиенты представлены векторами-столбцами, состоящими из частных производных по элементам $\boldsymbol{\theta}$, а π обозначает стратегию, соответствующую вектору параметров $\boldsymbol{\theta}$. Символ $\propto$ здесь означает «пропорционально». В эпизодическом случае коэффициент пропорциональности равен средней длине эпизода, а в непрерывном – 1, т. е. это соотношение превращается в равенство. Распределение μ здесь (как и в главах 9 и 10) – распределение с единой стратегией π (см. стр. 240). Доказательство теоремы о градиенте стратегии для эпизодического случая приведено во врезке выше.

13.3. REINFORCE: метод Монте-Карло НА ОСНОВЕ ГРАДИЕНТА СТРАТЕГИИ

Теперь мы готовы к выводу первого алгоритма обучения на основе градиента стратегии. Напомним общий подход к стохастическому градиентному подъему (13.1) – необходим какой-то способ производить выборку так, чтобы математическое ожидание выборочного градиента было пропорционально фактическому градиенту меры качества, выраженной в виде функции от параметра. Выборочные градиенты должны быть всего лишь пропорциональны фактическим, потому что коэффициент пропорциональности можно включить в размер шага α , а он может быть произвольным. Теорема о градиенте стратегии дает точное выражение, пропорциональное градиенту; остается только найти способ формирования выборки, математическое ожидание которой точно или приближенно равно этому

выражению. Отметим, что в правой части выражения (13.5) находится взвешенная сумма по состояниям, в которой веса равны частотам состояний при целевой стратегии π ; если следовать π , то состояния будут встречаться в такой пропорции:

$$\begin{aligned}\nabla J(\boldsymbol{\theta}) &\propto \sum_s \mu(s) \sum_a q_\pi(s, a) \nabla \pi(a|s, \boldsymbol{\theta}) \\ &= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t, a) \nabla \pi(a|S_t, \boldsymbol{\theta}) \right].\end{aligned}\quad (13.6)$$

Здесь можно было бы остановиться и записать наш стохастический алгоритм градиентного подъема (13.1) в виде

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \sum_a \hat{q}(S_t, a, \mathbf{w}) \nabla \pi(a|S_t, \boldsymbol{\theta}), \quad (13.7)$$

где $\hat{q}$ – какая-то обученная аппроксимация q_π . Этот алгоритм, называемый методом *всех действий*, поскольку в уравнение обновления входят все действия, является многообещающим и заслуживает дальнейшего изучения, но сейчас нас интересует классический алгоритм REINFORCE (Williams, 1992), для которого уравнение обновления в момент t включает только A_t – действие, фактически предпринятое в момент t . Мы продолжим вывод REINFORCE, введя A_t точно так же, как мы ввели S_t в (13.6), – путем замены суммы по возможным значениям случайной величины математическим ожиданием при следовании π и последующей выборки из этого математического ожидания. Уравнение (13.6) включает подходящую сумму по действиям, но вес каждого слагаемого не равен $\pi(a|S_t, \boldsymbol{\theta})$, как требуется для математического ожидания при следовании π . Поэтому мы введем такие веса, умножив и поделив каждое слагаемое на $\pi(a|S_t, \boldsymbol{\theta})$, равенство при этом сохранится. Продолжая (13.6), имеем:

$$\begin{aligned}\nabla J(\boldsymbol{\theta}) &= \mathbb{E}_\pi \left[\sum_a \pi(a|S_t, \boldsymbol{\theta}) q_\pi(S_t, a) \frac{\nabla \pi(a|S_t, \boldsymbol{\theta})}{\pi(a|S_t, \boldsymbol{\theta})} \right] \\ &= \mathbb{E}_\pi \left[q_\pi(S_t, A_t) \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta})}{\pi(A_t|S_t, \boldsymbol{\theta})} \right] \quad (\text{замена } a \text{ выборкой } A_t \sim \pi) \\ &= \mathbb{E}_\pi \left[G_t \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta})}{\pi(A_t|S_t, \boldsymbol{\theta})} \right], \quad (\text{поскольку } \mathbb{E}_\pi[G_t|S_t, A_t] = q_\pi(S_t, A_t))\end{aligned}$$

где G_t , как обычно, обозначает доход. Последнее выражение в квадратных скобках – именно то, что нам нужно, – величина, из которой можно производить выборку на каждом временном шаге, так что математическое ожидание будет равно градиенту. Применение этой выборки к общему стохастическому алгоритму градиентного подъема (13.1) дает уравнение обновления в REINFORCE:

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta})}{\pi(A_t|S_t, \boldsymbol{\theta})}. \quad (13.8)$$

Это обновление интуитивно понятно. Каждый инкремент пропорционален произведению дохода G_t и вектора, равного градиенту вероятности предпринять фактически предпринятое действие, поделенному на вероятность предпринять это действие. Данный вектор определяет направление в пространстве парамет-

ров, на котором больше всего увеличивается вероятность повторить действие A_t при будущих посещениях состояния S_t . Обновление увеличивает вектор параметров в этом направлении пропорционально доходу и обратно пропорционально вероятности действия. Первое имеет смысл, потому что заставляет параметр сильнее всего сдвигаться в направлениях, где предпочитаются действия, дающие максимальный доход. А второе имеет смысл, потому что в противном случае действия, которые часто выбираются, получают преимущество (обновления будут чаще производиться в их направлении) и могут одержать победу, даже если не приносят максимального дохода.

Отметим, что в REINFORCE используется полный доход с момента t , который включает все будущие вознаграждения до конца эпизода. В этом смысле REINFORCE является алгоритмом Монте-Карло и корректно определен только в эпизодическом случае, когда все обновления выполняются ретроспективно, после того как эпизод завершен (как в алгоритмах Монте-Карло из главы 5). Это явно отражено в псевдокоде во врезке ниже.

Обратите внимание, что обновление в последней строке псевдокода отличается от правила обновления REINFORCE (13.8). Одно отличие состоит в том, что в псевдокоде используется компактное выражение $\nabla \ln \pi(A_t | S_t, \theta_t)$ масштабированного вектора $\frac{\nabla \pi(A_t | S_t, \theta_t)}{\pi(A_t | S_t, \theta_t)}$ в (13.8). То, что оба выражения эквивалентны, следует из тождества $\nabla \ln x = \nabla x / x$. В литературе этот вектор называется и обозначается по-разному; мы будем называть его просто *вектором приемлемости*. Заметим, что это единственное место в алгоритме, где встречается параметризация стратегии.

REINFORCE: метод управления Монте-Карло на основе градиента стратегии (эпизодический) для π .

Вход: дифференцируемая параметризация стратегии $\pi(a | s, \theta)$

Параметр алгоритма: размер шага $\alpha > 0$

Инициализировать параметр стратегии $\theta \in \mathbb{R}^{d'}$ (например, значением 0)

Повторять бесконечно (для каждого эпизода):

Сгенерировать эпизод $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$, следуя $\pi(\cdot | \cdot, \theta)$

Повторять для каждого шага $t = 0, 1, \dots, T-1$:

$$G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k \quad (G_t)$$

$$\theta \leftarrow \theta + \alpha \gamma^t G \nabla \ln \pi(A_t | S_t, \theta)$$

Второе отличие между обновлением в псевдокоде и в уравнении (13.8) состоит в том, что в первом имеется коэффициент γ^t . Это связано с тем, что, как было отмечено выше, в тексте рассматривается случай без обесценивания ($\gamma = 1$), а во врезках мы приводим алгоритмы для общего случая, с обесцениванием. Все рассуждения проходят и в случае с обесцениванием при надлежащих поправках (это относится и к врезке на стр. 240), но дополнительные детали отвлекают от главных идей.

*Упражнение 13.2. Обобщите рассуждения во врезке на стр. 240, теорему о градиенте стратегии (13.5), ее доказательство и шаги, приведшие к уравнению обнов-

ления в REINFORCE (13.8), так чтобы в (13.8) появился коэффициент γ^t и, следовательно, было устранено расхождение с псевдокодом. $\square$

На рис. 13.1 показаны результаты работы алгоритма REINFORCE для задачи о коротком коридоре в сеточном мире из примера 13.1.

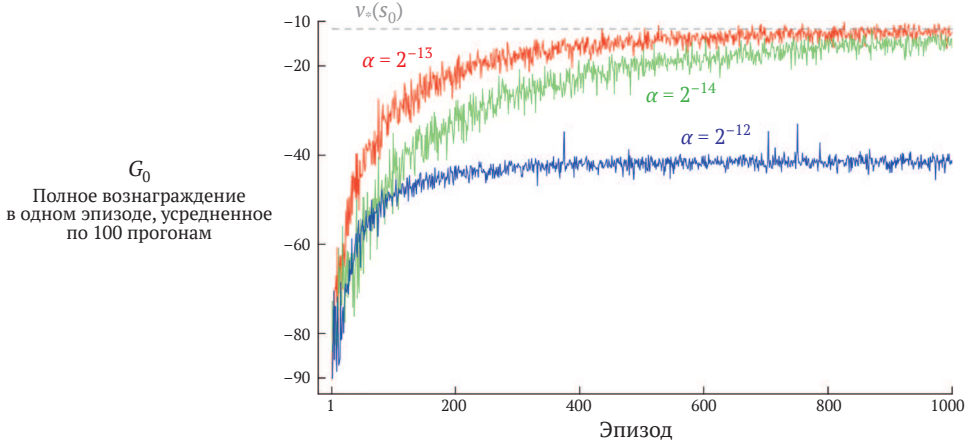


Рис. 13.1 ❖ Алгоритм REINFORCE для задачи о коротком коридоре в сеточном мире (пример 13.1). При удачно выбранном размере шага полное вознаграждение в одном эпизоде стремится к оптимальной ценности начального состояния

Будучи стохастическим градиентным методом, REINFORCE обладает хорошими теоретическими свойствами сходимости. По построению, ожидаемое обновление после эпизода направлено туда же, куда градиент меры качества. Тем самым гарантируется улучшение ожидаемого качества для достаточно малых α и сходимость к локальному оптимуму при стандартных условиях стохастической аппроксимации для убывающих α . Однако, будучи методом Монте-Карло, REINFORCE может иметь высокую дисперсию и обучаться медленно.

Упражнение 13.3. В разделе 13.1 мы рассмотрели параметризацию стратегии с помощью softmax по предпочтениям действий (13.2) с линейными предпочтениями действий (13.3). Для этой параметризации, применяя определения и факты из элементарного математического анализа, докажите, что вектор приемлемости имеет вид:

$$\nabla \ln \pi(a|s, \theta) = \mathbf{x}(s, a) - \sum_b \pi(b|s, \theta) \mathbf{x}(s, b). \quad (13.9)$$

$\square$

13.4. REINFORCE с базой

Теорему о градиенте стратегии (13.5) можно обобщить, включив сравнение ценностей действий с произвольной базой $b(s)$

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a (q_\pi(s, a) - b(s)) \nabla \pi(a|s, \theta). \quad (13.10)$$

База может быть любой функцией, даже случайной величиной, при условии что она не зависит от a ; равенство остается справедливым, потому что вычитаемая величина равна нулю:

$$\sum_a b(s) \nabla \pi(a|s, \theta) = b(s) \nabla = \sum_a \pi(a|s, \theta) = b(s) \nabla 1 = 0.$$

Теорему о градиенте стратегии с базой (13.10) можно использовать для вывода правила обновления, проделав примерно такие же шаги, как в предыдущем разделе. Получится новый вариант алгоритма REINFORCE, включающий базу общего вида:

$$\theta_{t+1} \doteq \theta_t + \alpha (G_t - b(S_t)) \frac{\nabla \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)}. \quad (13.11)$$

Поскольку база может быть всюду равна нулю, это обновление является строгим обобщением алгоритма REINFORCE. В общем случае база оставляет математическое ожидание обновления неизменным, но может оказать заметное влияние на его дисперсию. Например, в разделе 2.8 мы видели, что аналогичная база может значительно уменьшить дисперсию (а значит, ускорить обучение) градиентных алгоритмов бандита. В алгоритмах бандита база была просто числом (средним наблюдававшихся до сих пор вознаграждений), но в случае МППР база должна зависеть от состояния. В одних состояниях ценности всех действий высоки, и нам нужна высокая база, чтобы разделить действия большей и меньшей ценности; в других состояниях ценности всех действий малы, так что необходима низкая база.

Одним из способов выбора базы является оценка ценности состояния $\hat{v}(S_t, \mathbf{w})$, где $\mathbf{w} \in \mathbb{R}^m$ – вектор весов, обученный одним из методов, представленных в предыдущих главах. Поскольку REINFORCE – метод Монте-Карло для обучения параметра стратегии θ , кажется естественным использовать метод Монте-Карло и для обучения весов ценностей состояний $\mathbf{w}$. Полный псевдокод алгоритма REINFORCE с базой, в котором в качестве базы используется так обученная функция ценности состояний, приведен во врезке ниже.

REINFORCE с базой (эпизодический) для оценивания $\pi_\theta \approx \pi_*$

Вход: дифференцируемая параметризация стратегии $\pi(a|s, \theta)$

Вход: дифференцируемая функция ценности состояний $\hat{v}(s, \mathbf{w})$

Параметры алгоритма: размеры шагов $\alpha^\theta > 0$, $\alpha^w > 0$

Инициализировать параметр стратегии $\theta \in \mathbb{R}^d$ и веса ценностей состояний $\mathbf{w} \in \mathbb{R}^d$ (например, значением $\mathbf{0}$)

Повторять бесконечно (для каждого эпизода):

Сгенерировать эпизод $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$, следуя $\pi(\cdot|\cdot, \theta)$

Повторять для каждого шага эпизода $t = 0, 1, \dots, T - 1$:

$$G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k \quad (G_t)$$

$$\delta \leftarrow G - \hat{v}(S_t, \mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha^w \delta \nabla \hat{v}(S_t, \mathbf{w})$$

$$\theta \leftarrow \theta + \alpha^\theta \gamma^t \delta \nabla \ln \pi(A_t|S_t, \theta)$$

В этом алгоритме два размера шагов, обозначенных α^θ и α^w (где α^θ – то же, что α в (13.11)). Выбрать размер шага для ценностей (здесь α^w) сравнительно просто; в линейном случае у нас имеются эвристические правила для этого, например $\alpha^w = 0.1 \mathbb{E}[\|\nabla \hat{v}(S_t, \mathbf{w})\|_\mu^2]$ (см. раздел 9.6). Куда менее понятно, как задавать размер шага для параметра стратегии α^θ , наилучшее значение которого зависит от диапазона изменения вознаграждений и от параметризации стратегии.

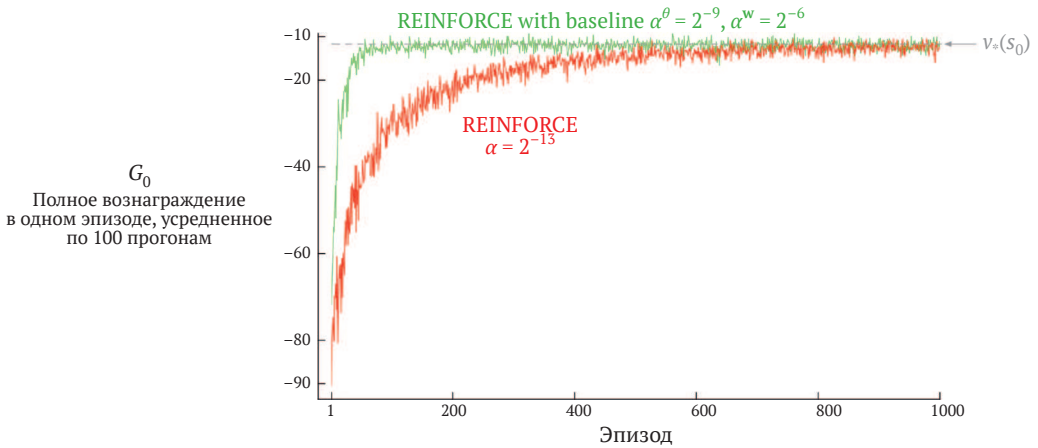


Рис. 13.2 ❖ В результате добавления базы алгоритм REINFORCE, возможно, будет обучаться быстрее, как показано здесь для задачи о коротком коридоре в сеточном мире (рис. 13.1). В данном случае для простого варианта REINFORCE взят размер шага, при котором он показывает наилучшие результаты (округленный до ближайшей степени 2, см. рис. 13.1)

На рис. 13.2 сравнивается поведение REINFORCE с базой и без базы для задачи о коротком коридоре в сеточном мире (пример 13.1). Здесь в качестве приближенной функции ценности в базе использовалась $\hat{v}(s, \mathbf{w}) = w$, т. е. вектор $\mathbf{w}$ содержит всего один элемент w .

13.5. Методы исполнитель–критик

Хотя метод REINFORCE с базой обучает как стратегию, так и функцию ценности состояний, мы не считаем его методом исполнитель–критик, потому что функция ценности состояний в нем используется только как база, но не как критик. То есть она не используется для бутстрэппинга (обновления оценки ценности состояния по оценкам ценности последующих состояний), а лишь как база для состояния, чья оценка обновляется. Это полезное различие, поскольку лишь посредством бутстрэппинга мы вводим смещение и асимптотическую зависимость от качества аппроксимации функции. Как мы видели, смещение вследствие бутстрэппинга и зависимость от представления состояний часто приносят пользу, потому что уменьшают дисперсию и ускоряют обучение. REINFORCE с базой – несмещенный алгоритм, который асимптотически сходится к локальному минимуму, но, как и все методы Монте-Карло, обычно обучается медленно (порождает оценки с высокой дисперсией) и неудобен для реализации в онлайн-режиме или для ре-

шения непрерывных задач. Ранее в этой книге мы видели, что методы на основе временных различий могут устранить эти неудобства, а благодаря многошаговым методам удастся гибко выбрать степень бутстрэппинга. Чтобы получить все эти преимущества в случае методов градиента стратегии, мы обращаемся к методам исполнитель–критик, в которых критик осуществляет бутстрэппинг.

Сначала рассмотрим одношаговые методы исполнитель–критик, аналог TD-методов, введенных в главе 6: TD(0), Sarsa(0) и Q-обучение. Главное, чем хороши одношаговые методы, – тот факт, что они полностью онлайнные и инкрементные, но при этом избегают сложностей, присущих следам приемлемости. Это частный случай методов на основе следов приемлемости, который, однако, проще для понимания. Одношаговые методы исполнитель–критик следующим образом заменяют полный доход в алгоритме REINFORCE (13.11) одношаговым доходом (и используют в качестве базы обученную функцию ценности состояний):

$$\theta_{t+1} \doteq \theta_t + \alpha(G_{t:t+1} - \hat{v}(S_t, \mathbf{w})) \frac{\nabla \pi(A_t | S_t, \theta_t)}{\pi(A_t | S_t, \theta_t)} \quad (13.12)$$

$$= \theta_t + \alpha(R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})) \frac{\nabla \pi(A_t | S_t, \theta_t)}{\pi(A_t | S_t, \theta_t)} \quad (13.13)$$

$$= \theta_t + \alpha \delta_t \frac{\nabla \pi(A_t | S_t, \theta_t)}{\pi(A_t | S_t, \theta_t)}. \quad (13.14)$$

В качестве сопутствующего метода обучения функции ценности состояний естественно взять полуградиентный TD(0). Полный псевдокод алгоритма приведен во врезке ниже. Отметим, что теперь это полностью онлайнный инкрементный алгоритм, в котором состояния, действия и вознаграждения обрабатываются по мере появления, и больше алгоритм к ним не возвращается.

Одношаговый метод исполнитель–критик (эпизодический) для оценивания $\pi_\theta \approx \pi_*$

Вход: дифференцируемая параметризация стратегии $\pi(a|s, \theta)$

Вход: дифференцируемая функция ценности состояний $\hat{v}(s, \mathbf{w})$

Параметры алгоритма: размеры шагов $\alpha^\theta > 0, \alpha^w > 0$

Инициализировать параметр стратегии $\theta \in \mathbb{R}^{d'}$ и веса ценностей состояний $\mathbf{w} \in \mathbb{R}^d$ (например, значением $\mathbf{0}$)

Повторять бесконечно (для каждого эпизода):

Инициализировать S (первое состояние эпизода)

$I \leftarrow 1$

Повторять, пока S не заключительное (для каждого временного шага)

$A \sim \pi(\cdot | S, \theta)$

Предпринять действие A , наблюдать S', R

$\delta \leftarrow R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$ (если S' заключительное, то $\hat{v}(S', \mathbf{w}) \doteq 0$)

$\mathbf{w} \leftarrow \mathbf{w} + \alpha^w \delta \nabla \hat{v}(S, \mathbf{w})$

$\theta \leftarrow \theta + \alpha^\theta I \delta \nabla \ln \pi(A | S, \theta)$

$I \leftarrow \gamma I$

$S \leftarrow S'$

Обобщение на n -шаговые методы прямого представления, а затем на λ -доходный алгоритм не вызывает трудностей. Одношаговый доход в (13.12) просто заменяется на $G_{t:t+n}$ или G_t^λ соответственно. Обратное представление λ -доходного алгоритма тоже строится просто с использованием отдельных следов приемлемости для исполнителя и критика – по образцу, приведенному в главе 12. Полный псевдокод алгоритма приведен во врезке ниже.

Метод исполнитель–критик со следами приемлемости (эпизодический) для оценивания $\pi_\theta \approx \pi_*$

Вход: дифференцируемая параметризация стратегии $\pi(a|s, \theta)$
 Вход: дифференцируемая функция ценности состояний $\hat{v}(s, \mathbf{w})$
 Параметры алгоритма: скорости затухания следов $\lambda^\theta \in [0, 1]$, $\lambda^w \in [0, 1]$;
 размеры шагов $\alpha^\theta > 0$, $\alpha^w > 0$
 Инициализировать параметр стратегии $\theta \in \mathbb{R}^{d'}$ и веса ценностей состояний $\mathbf{w} \in \mathbb{R}^d$ (например, значением $\mathbf{0}$)

Повторять бесконечно (для каждого эпизода):

Инициализировать S (первое состояние эпизода)

$\mathbf{z}^\theta \leftarrow \mathbf{0}$ (d' -мерный вектор следа приемлемости)

$\mathbf{z}^w \leftarrow \mathbf{0}$ (d -мерный вектор следа приемлемости)

$I \leftarrow 1$

Повторять, пока S не заключительное (для каждого временного шага)

$A \sim \pi(\cdot | S, \theta)$

Предпринять действие A , наблюдать S', R

$\delta \leftarrow R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$ (если S' заключительное, то $\hat{v}(S', \mathbf{w}) \doteq 0$)

$\mathbf{z}^w \leftarrow \gamma \lambda^w \mathbf{z}^w + \nabla \hat{v}(S, \mathbf{w})$

$\mathbf{z}^\theta \leftarrow \gamma \lambda^\theta \mathbf{z}^\theta + I \nabla \ln \pi(A | S, \theta)$

$\mathbf{w} \leftarrow \mathbf{w} + \alpha^w \delta \mathbf{z}^w$

$\theta \leftarrow \theta + \alpha^\theta \delta \mathbf{z}^\theta$

$I \leftarrow \gamma I$

$S \leftarrow S'$

13.6. МЕТОД ГРАДИЕНТА СТРАТЕГИИ ДЛЯ НЕПРЕРЫВНЫХ ЗАДАЧ

В разделе 10.3 мы говорили, что для непрерывных задач с неразграниченными эпизодами качество следует определять в терминах среднего темпа вознаграждения на временном шаге:

$$\begin{aligned}
 J(\theta) &\doteq r(\pi) \doteq \lim_{h \rightarrow \infty} \frac{1}{h} \sum_{t=1}^h \mathbb{E}[R_t | S_0, A_{0:t-1} \sim \pi] \\
 &= \lim_{h \rightarrow \infty} \mathbb{E}[R_t | S_0, A_{0:t-1} \sim \pi] \\
 &= \sum_s \mu(s) \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) r,
 \end{aligned} \tag{13.15}$$

где μ – установившееся распределение при стратегии π , $\mu(s) = \lim_{t \rightarrow \infty} \Pr\{S_t = s \mid A_{0:t} \sim \pi\}$, которое, по предположению, существует и независимо от S_0 (свойство эргодичности). Напомним, что это специальное распределение такое, что если действия выбираются согласно π , то мы остаемся в том же самом распределении:

$$\sum_s \mu(s) \sum_a \pi(a|s, \theta) p(s'|s, a) = \mu(s) \quad \text{для всех } s' \in S. \quad (13.16)$$

Полный псевдокод алгоритма исполнитель–критик в непрерывном случае (обратное представление) приведен во врезке ниже.

**Метод исполнитель–критик со следами приемлемости (непрерывный)
для оценивания $\pi_\theta \approx \pi_*$**

Вход: дифференцируемая параметризация стратегии $\pi(a|s, \theta)$

Вход: дифференцируемая функция ценности состояний $\hat{v}(s, \mathbf{w})$

Параметры алгоритма: скорости затухания следов $\lambda^\theta \in [0, 1]$, $\lambda^w \in [0, 1]$, $\alpha^\theta > 0$, $\alpha^w > 0$, $\alpha^R > 0$

Инициализировать $\bar{R} \in \mathbb{R}$ (например, значением 0)

Инициализировать параметр стратегии $\theta \in \mathbb{R}^{d'}$ и веса ценностей состояний $\mathbf{w} \in \mathbb{R}^d$ (например, значением 0)

$\mathbf{z}^w \leftarrow \mathbf{0}$ (d -мерный вектор следа приемлемости)

$\mathbf{z}^\theta \leftarrow \mathbf{0}$ (d' -мерный вектор следа приемлемости)

Повторять бесконечно (для каждого временного шага):

$A \sim \pi(\cdot | S, \theta)$

Предпринять действие A , наблюдать S', R

$\delta \leftarrow R - \bar{R} + \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$

$\bar{R} \leftarrow \bar{R} + \alpha^R \delta$

$\mathbf{z}^w \leftarrow \lambda^w \mathbf{z}^w + \nabla \hat{v}(S, \mathbf{w})$

$\mathbf{z}^\theta \leftarrow \lambda^\theta \mathbf{z}^\theta + \nabla \ln \pi(A | S, \theta)$

$\mathbf{w} \leftarrow \mathbf{w} + \alpha^w \delta \mathbf{z}^w$

$\theta \leftarrow \theta + \alpha^\theta \delta \mathbf{z}^\theta$

$S \leftarrow S'$

Естественно, в непрерывном случае мы определяем ценности $v_\pi(s) \doteq \mathbb{E}_\pi[G_t | S_{t=s}]$ и $q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t | S_{t=s}, A_{t=a}]$ относительно дифференциального дохода:

$$G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \dots \quad (13.17)$$

При таких определениях теорема о градиенте стратегии в формулировке для эпизодического случая (13.5) остается справедливой и в непрерывном случае. Доказательство приведено во врезке ниже. Уравнения прямого и обратного представлений также не изменяются.

Доказательство теоремы о градиенте стратегии (непрерывный случай)

Доказательство теоремы о градиенте стратегии для непрерывного случая начинается так же, как в эпизодическом случае. Снова мы неявно считаем, что π всегда является функцией от θ и что все градиенты берутся по θ . Напомним, что в непрерывном случае $J(\theta) = r(\pi)$ (13.15) и что v_π и q_π обозначают ценности относительно дифференциального дохода (13.17). Градиент функции ценности состояний можно для любого $s \in \mathcal{S}$ записать в виде:

$$\begin{aligned} \nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s, a) \right] \quad \text{для всех } s' \in \mathcal{S} && \text{(упражнение 3.18)} \\ &= \sum_a \left[\nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \nabla q_\pi(s, a) \right] && \text{(правило произведения из математического анализа)} \\ &= \sum_a \left[\nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \nabla \sum_{s', r} p(s', r|s, a) (r - r(\theta) + v_\pi(s')) \right] \\ &= \sum_a \left[\nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \left[-\nabla r(\theta) + \sum_{s'} p(s'|s, a) \nabla v_\pi(s') \right] \right]. \end{aligned}$$

После переупорядочения членов получаем:

$$\nabla r(\theta) = \sum_a \left[\nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \sum_{s'} p(s'|s, a) \nabla v_\pi(s') \right] - \nabla v_\pi(s).$$

Заметим, что левую часть можно записать в виде $\nabla J(\theta)$, не зависящем от s . Поэтому правая часть тоже не зависит от s , и мы можем смело просуммировать ее по всем $s \in \mathcal{S}$ с весами $\mu(s)$, ничего не изменив (поскольку $\sum_s \mu(s) = 1$):

$$\begin{aligned} \nabla J(\theta) &= \sum_s \mu(s) \left(\sum_a \left[\nabla \pi(a|s) q_\pi(s, a) + \pi(a|s) \sum_{s'} p(s'|s, a) \nabla v_\pi(s') \right] - \nabla v_\pi(s) \right) \\ &= \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s, a) \\ &\quad + \sum_s \mu(s) \sum_a \pi(a|s) \sum_{s'} p(s'|s, a) \nabla v_\pi(s') - \sum_s \mu(s) \nabla v_\pi(s) \\ &= \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s, a) \\ &\quad + \underbrace{\sum_{s'} \sum_s \mu(s) \sum_a \pi(a|s) p(s'|s, a) \nabla v_\pi(s')}_{\mu(s') \text{ (13.16)}} - \sum_s \mu(s) \nabla v_\pi(s) \\ &= \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s, a) + \sum_{s'} \mu(s') \nabla v_\pi(s') - \sum_s \mu(s) \nabla v_\pi(s) \\ &= \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s, a). \quad \text{ЧТД} \end{aligned}$$

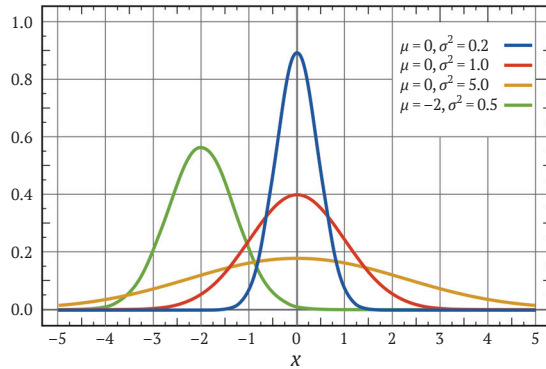
13.7. ПАРАМЕТРИЗАЦИЯ СТРАТЕГИИ ДЛЯ НЕПРЕРЫВНЫХ ДЕЙСТВИЙ

Основанные на стратегии методы предлагают практические способы работы с большими пространствами действий, даже непрерывными, в которых количество действий бесконечно. Вместо вычисления обученных вероятностей для каждого из многих действий мы обучаемся статистике распределения вероятностей. Например, множество действий может состоять из вещественных чисел, а сами действия выбираются из нормального (гауссова) распределения.

Функцию плотности вероятности нормального распределения традиционно принято записывать в виде:

$$p(x) \doteq \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \quad (13.18)$$

где μ и σ – среднее и стандартное отклонение нормального распределения, а $\pi \approx 3.14159$. На рисунке ниже изображены функции плотности вероятности для нескольких значений среднего и нормального отклонения. Значением $p(x)$ является плотность вероятности в точке x , а не вероятность. Она может быть больше 1; единице должна быть равна общая площадь под кривой $p(x)$. В общем случае, чтобы узнать, чему равна вероятность попадания x в некоторый диапазон, следует проинтегрировать $p(x)$ по этому диапазону.



Для параметризации стратегию можно определить как плотность нормального распределения вещественных скалярных действий, тогда среднее и стандартное отклонение станут параметрическими аппроксиматорами, зависящими от состояния. Это означает, что

$$\pi(a|s, \theta) \doteq \frac{1}{\sigma(s, \theta)\sqrt{2\pi}} \exp\left(-\frac{(a - \mu(s, \theta))^2}{2\sigma(s, \theta)^2}\right), \quad (13.19)$$

где $\mu: \mathcal{S} \times \mathbb{R}^{d'} \rightarrow \mathbb{R}$ и $\sigma: \mathcal{S} \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^+$ – параметрические аппроксиматоры функций.

Чтобы завершить пример, нам нужна только форма этих аппроксиматоров. Для этого разобьем вектор параметров стратегии на две части: $\theta = [\theta_\mu, \theta_\sigma]^T$, одна из

которых используется для аппроксимации среднего, другая – для аппроксимации стандартного отклонения. Среднее можно аппроксимировать линейной функцией. Стандартное отклонение всегда должно быть положительным, аппроксимировать его лучше экспонентой линейной функции. Таким образом:

$$\mu(s, \theta) \doteq \theta_\mu^\top \mathbf{x}_\mu(s) \quad \text{и} \quad \sigma(s, \theta) \doteq \exp(\theta_\sigma^\top \mathbf{x}_\sigma(s)), \quad (13.20)$$

где $\mathbf{x}_\mu(s)$ и $\mathbf{x}_\sigma(s)$ – векторы признаков состояний, построенные, быть может, одним из методов, описанных в главе 9. При таких определениях все алгоритмы, описанные далее в этой главе, становятся применимы для обучения выбору вещественных действий.

Упражнение 13.4. Покажите, что для гауссовой параметризации стратегии (13.19) вектор приемлемости состоит из таких двух частей:

$$\begin{aligned} \nabla \ln \pi(a|s, \theta_\mu) &= \frac{\nabla \pi(a|s, \theta_\mu)}{\pi(a|s, \theta)} = \frac{1}{\sigma(s, \theta)^2} (a - \mu(s, \theta)) \mathbf{x}_\mu(s) \quad \text{и} \\ \nabla \ln \pi(a|s, \theta_\sigma) &= \frac{\nabla \pi(a|s, \theta_\sigma)}{\pi(a|s, \theta)} = \left(\frac{(a - \mu(s, \theta))^2}{\sigma(s, \theta)^2} - 1 \right) \mathbf{x}_\sigma(s). \end{aligned} \quad \square$$

Упражнение 13.5. *Логистический блок Бернулли* – это стохастический нейроноподобный блок, применяемый в некоторых ИНС (раздел 9.7). Его входом в момент t является вектор признаков $\mathbf{x}(S_t)$, а выходом – случайная величина A_t , принимающая два значения, 0 и 1, с вероятностями $\Pr\{A_t = 1\} = P_t$ и $\Pr\{A_t = 0\} = 1 - P_t$ (распределение Бернулли). Обозначим $h(s, 0, \theta)$ и $h(s, 1, \theta)$ предпочтения, отдаваемые двум действиям блока в состоянии s при заданном параметре стратегии θ . Предположим, что разность между предпочтениями действий описывается взвешенной суммой элементов вектора входов в блок, т. е. $h(s, 1, \theta) - h(s, 0, \theta) = \theta^\top \mathbf{x}(s)$, где θ – вектор весов блока.

- Покажите, что если для преобразования предпочтений действий в стратегии используется экспоненциальное распределение softmax (13.2), то $P_t = \pi(1|S_t, \theta_t) = 1/(1 + \exp(\theta_t^\top \mathbf{x}(S_t)))$ (логистическая функция).
- Как выглядит обновление θ_t до θ_{t+1} в методе REINFORCE Монте-Карло при получении дохода G_t ?
- Выразите приемлемость $\nabla \ln \pi(a|s, \theta)$ для логистического блока Бернулли в терминах a , $\mathbf{x}(s)$ и $\pi(a|s, \theta)$, вычислив градиент.

Указание: отдельно для каждого действия вычислите производную логарифма по $P_t = \pi(a|s, \theta)$, объедините оба результата в одно выражение, зависящее от a и P_t , а затем воспользуйтесь правилом дифференцирования сложной функции, заметив, что производная логистической функции $f(x)$ равна $f(x)(1 - f(x))$.

13.8. РЕЗЮМЕ

До этой главы мы занимались *методами ценности действий*, т. е. методами, которые обучаются ценностям действий, а затем используют их для решения о том, какое действие выбрать. Но здесь мы рассматривали методы, которые обучаются параметрической стратегии, позволяющей выбирать действия, не обращаясь

к оценкам их ценностей. В частности, мы обсудили *методы градиента стратегии*, т. е. методы, которые на каждом шаге обновляют параметр стратегии в направлении оценки градиента показателя качества по параметру стратегии.

У методов, которые обучают и хранят параметр стратегии, много преимуществ. Они могут обучиться конкретным вероятностям выбора действий. Они могут обучиться подходящим уровням исследования и асимптотически приблизиться к детерминированным стратегиям. Они естественно применяются к непрерывным пространствам действий. Все это делается легко для методов, основанных на стратегии, но неуклюже или вообще невозможно для ϵ -жадных методов и вообще для методов ценности действий. Кроме того, в некоторых задачах проще представить параметрически стратегию, а не функцию ценности; такие задачи больше подходят для применения методов, параметризующих стратегию.

У методов, параметризующих стратегию, имеется также важное теоретическое преимущество над методами ценности действий в виде *теоремы о градиенте стратегии*, которая дает точную формулу, описывающую влияние параметра стратегии на качество и не содержащую производных распределения состояния. Эта теорема закладывает теоретический фундамент под все методы градиента стратегии.

Метод REINFORCE непосредственно вытекает из теоремы о градиенте стратегии. Добавление функции ценности состояний в качестве *базы* уменьшает дисперсию REINFORCE, не внося при этом смещения. Использование функции ценности состояний для бутстрэппинга вносит смещение, но зачастую это даже желательно по той же причине, по какой бутстрэппинговые TD-методы превосходят методы Монте-Карло (значительное уменьшение дисперсии). Функция ценности состояний назначает уровень доверия различным вариантам выбора действий, предлагаемым стратегией, – играет роль критика, поэтому функция ценности называется критиком, а стратегия – исполнителем. В целом же такого рода методы называются методами исполнитель–критик.

В общем и целом сильные и слабые стороны методов градиента стратегии существенно другие, чем у методов ценности действий. В настоящее время они не так хорошо изучены в некоторых отношениях, но интерес к ним велик и исследования продолжаются.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

Методы, которые мы теперь называем методами градиента стратегии, на самом деле были одними из первых применявшихся в обучении с подкреплением (Witten, 1977; Barto, Sutton, and Anderson, 1983; Sutton, 1984; Williams, 1987, 1992) и предшествующих ему областях (Phansalkar and Thathachar, 1995). В 1990-х годах они были в значительной мере вытеснены методами ценности действий, которые находились в фокусе нашего внимания в остальных главах. Но в последние годы интерес к методам исполнитель–критик и методам градиента стратегии вообще вернулся. Из тем, которые остались за рамками нашего рассмотрения, упомянем методы натурального градиента (Amari, 1998; Kakade, 2002; Peters, Vijayakumar and Schaal, 2005; Peters and Schall, 2008; Park, Kim and Kang, 2005; Bhatnagar, Sutton, Ghavamzadeh and Lee, 2009; see Grondman, Busoniu, Lopes and Babuska, 2012)

и детерминированные методы градиента стратегии (Silver et al., 2014), методы градиента стратегии с разделенной стратегией (Degris, White, and Sutton, 2012; Maei, 2018) и регуляризацию энтропии (Schulman, Chen, and Abbeel, 2017). К числу важных приложений относятся автопилоты для акробатических вертолетов и программа AlphaGo (раздел 16.6).

Изложение в этой главе основано главным образом на работе Sutton, McAllester, Singh, and Mansour (2000), где был введен термин «методы градиента стратегии». Полезный обзор имеется в работе Bhatnagar et al. (2009). Одна из ранних работ по теме – Aleksandrov, Sysoyev, and Shemenева (1968). В статье Thomas (2014) впервые осознано, что коэффициент γ^t , встречающийся в алгоритмах, представленных во врезках, необходим в случае эпизодических задач с обесцениванием.

- 13.1 В разработке примера 13.1 и его результатов, представленных в этой главе, принимал участие Эрик Грейвс (Eric Graves).
- 13.2 Теорема о градиенте стратегии, изложенная здесь и на стр. 387, впервые была доказана в работах Marbach and Tsitsiklis (1998, 2001), а затем независимо в работе Sutton et al. (2000). Похожее выражение было получено в работе Cao and Chen (1997). Из других ранних результатов отметим статьи Konda and Tsitsiklis (2000, 2003), Baxter and Bartlett (2001) и Baxter, Bartlett, and Weaver (2001). Некоторые дополнительные результаты были получены в работе Sutton, Singh, and McAllester (2000).
- 13.3 Алгоритм REINFORCE сформулирован в работах Williams (1987, 1992). В работе Phansalkar and Thathachar (1995) доказаны теоремы о локальной и глобальной сходимости для модифицированных вариантов REINFORCE. Алгоритм всех действий впервые описан в неопубликованной, но получившей широкое распространение незаконченной статье Sutton, Singh, and McAllester, 2000, а затем аналитически и эмпирически изучался в работе Asadi, Allen, Roderick, Mohamed, Konidaris, and Littman (2017), которые называли его алгоритмом «среднего исполнителя–критика». Обобщение на непрерывные действия описано в статье Ciosek and Whiteson (2018), где употреблялся термин «ожидаемые градиенты стратегии».
- 13.4 База была введена в оригинальных работах Williams (1987, 1992). В работе Greensmith, Bartlett, and Baxter (2004) проанализирована потенциально улучшенная база (см. Dick, 2015). В работе Thomas and Brunskill (2017) аргументируется, что зависящую от действия базу можно использовать, не внося смещения.
- 13.5–6 Методы исполнитель–критик относятся к числу первых исследованных в обучении с подкреплением (Witten, 1977; Barto, Sutton, and Anderson, 1983; Sutton, 1984). Представленные здесь алгоритмы основаны на работе Degris, White, and Sutton (2012), где также начато изучение методов градиента стратегии с разделенной стратегией. Иногда методы исполнитель–критик в литературе называют *авантажными* методами исполнитель–критик.
- 13.7 Первым, кто показал, как можно таким образом работать с непрерывными действиями, похоже, был Уильямс (Williams, 1987, 1992). Рисунок на стр. 388 заимствован из Википедии.

Часть III

ЗАГЛЯНЕМ ПОГЛУБЖЕ

В последней части книги мы выйдем за пределы стандартных идей обучения с подкреплением, представленных в первых двух частях, и совершим краткий экскурс в их связи с психологией и нейронауками, опишем избранные приложения обучения с подкреплением и некоторые передовые рубежи, где куется будущее этой дисциплины.

Глава 14

Психология

В предыдущих главах мы развивали идеи алгоритмов, основываясь только на вычислительных соображениях. В этой главе мы взглянем на некоторые алгоритмы под другим углом: с точки зрения психологии и того, как в ней изучается обучение животных. Мы хотим, во-первых, обсудить, как идеи и алгоритмы обучения с подкреплением соотносятся с открытиями психологов в области обучения животных, а во-вторых, объяснить, какое влияние обучение с подкреплением оказывает на исследования в этой области. Обучение с подкреплением предлагает четкий формализм, в котором систематизированы задачи, доходы и алгоритмы. Он чрезвычайно полезен для интерпретации экспериментальных данных, для планирования новых видов экспериментов и для выявления факторов, которые могут оказаться критическими для манипулирования и измерения. Идея оптимизации дохода в долгосрочной перспективе, лежащая в основе обучения с подкреплением, вносит вклад в наше понимание тех особенностей обучения и поведения животных, которые в противном случае представляли бы загадку.

Некоторые параллели между обучением с подкреплением и психологическими теориями неудивительны, потому что развитие обучения с подкреплением черпало идеи из психологических теорий обучения. Но, как постоянно подчеркивалось в этой книге, в обучении с подкреплением изучаются идеализированные ситуации с точки зрения исследователя или конструктора искусственного интеллекта, а цель – решение вычислительных задач с помощью эффективных алгоритмов, а не детальное воспроизведение или объяснение механизмов обучения, характерных для животных. В результате некоторые описанные нами параллели связывают идеи, независимо возникшие в своих дисциплинах. Мы полагаем, что такие точки соприкосновения особенно важны, потому что позволяют выявить вычислительные принципы, важные для обучения как искусственных, так и естественных систем.

По большей части мы описываем параллели между обучением с подкреплением и теориями обучения, разработанными для объяснения того, как животные – крысы, голуби и кролики – обучаются в контролируемых лабораторных условиях. В XX веке были проведены тысячи таких экспериментов, и еще многие ставятся до сих пор. Иногда эти эксперименты игнорировались как не имеющие отношения к более широким проблемам психологии, но вообще в них исследуются трудноуловимые свойства обучения животных, зачастую в ответ на точно сформулированные теоретические вопросы. По мере того как интересы психологов смещались в сторону более когнитивных аспектов обучения, т. е. таких ментальных процессов, как мышление и способность к рассуждениям, эксперименты по

обучению животных начали играть меньшую роль, чем раньше. Но эти эксперименты привели к открытию принципов обучения, изначально присущих и широко распространенных в животном мире, принципов, которыми не следует пренебрегать при проектировании искусственных систем обучения. Кроме того, как мы увидим, некоторые аспекты познавательного процесса естественным образом связаны с вычислительным подходом к проблеме, который предлагает обучение с подкреплением.

В последний раздел этой главы включены ссылки, относящиеся как к обсуждаемым нами связям, так и к тем, которые мы опустили. Мы надеемся, что эта глава побудит читателей к более глубокому изучению этих связей. Также в последний раздел вошло обсуждение того, какое отношение употребляемая в обучении с подкреплением терминология имеет к психологии. Многие термины и выражения, встречающиеся в обучении с подкреплением, заимствованы из теорий обучения животных, но их смысл в вычислительно-конструкторском контексте не всегда совпадает с тем, который они имеют в психологии.

14.1. ПРЕДСКАЗАНИЕ И УПРАВЛЕНИЕ

Описанные в этой книге алгоритмы можно отнести к двум большим категориям: алгоритмы *предсказания* и алгоритмы *управления*. Они естественно возникают в методах решения задачи обучения с подкреплением, поставленной в разделе 3. Во многих отношениях эти категории соответствуют категориям обучения, которые давно и плодотворно изучают психологи: *классическое*, или *павловское*, *обусловливание* и *инструментальное*, или *оперантное*, *обусловливание*. Эти параллели отнюдь не случайны, поскольку психология оказала влияние на обучение с подкреплением, но тем не менее они поражают, т. к. связывают идеи, возникшие в связи с совершенно разными целями.

Алгоритмы предсказания, описанные в этой книге, служат для оценки количественных величин, зависящих от ожидаемого разворачивания признаков окружающей среды в будущем. Конкретно нас интересует оценивание вознаграждения, на которое агент может рассчитывать в будущем в процессе взаимодействия со средой. В этом качестве алгоритмы предсказания – это *алгоритмы оценивания стратегии*, являющиеся неотъемлемой составной частью алгоритмов улучшения стратегии. Но алгоритмы предсказания не ограничиваются предсказанием будущего вознаграждения; они способны предсказывать любой признак окружающей среды (см., например, Modayil, White, and Sutton, 2014). Параллель между алгоритмами предсказания и классическим обусловливанием покоится на общем для них свойстве предсказания предстоящих стимулов, независимо от того, являются эти стимулы вознаграждением (или наказанием) или нет.

Ситуация с экспериментами по инструментальному, или оперантному, обусловливанию иная. Здесь экспериментальная установка устроена так, что животное получает нечто такое, что ему нравится (вознаграждение) или не нравится (наказание), в зависимости от своих действий. Животное обучается увеличивать тягу к поведению, порождающему вознаграждение, и уменьшать тягу к поведению, влекущему наказание. Говорят, что подкрепляющий стимул *контингентен* поведению животного, тогда как в классической теории условных рефлексов это не

так (хотя трудно устранить всякую контингентность поведения в эксперименте по классическому обусловливанию). Эксперименты по инструментальному обусловливанию похожи на тот, что привел к закону эффекта Торндайка, обсуждавшемуся в главе 1. *Управление*¹ лежит в основе этой формы обучения, которая соответствует работе алгоритмов улучшения стратегии в обучении с подкреплением.

Размышления о классическом обусловливании в терминах предсказания и об инструментальном обусловливании в терминах управления – отправная точка для связывания нашего вычислительного взгляда на обучение с подкреплением с обучением животных, но в реальности ситуация сложнее. Классическое обусловливание не сводится к предсказанию, оно включает также и действие, как и режим управления, который иногда называют *павловским управлением*. Кроме того, классическое и инструментальное обусловливания взаимодействуют различными интересными способами, так что в большинстве экспериментов оба вида обучения оказываются переплетены. Но, несмотря на эти осложнения, проведение параллелей между классическим/инструментальным обусловливанием и предсказанием/управлением удобно в качестве первого приближения к установлению связей между обучением с подкреплением и обучением животных.

В психологии термин «подкрепление» служит для описания обучения как при классическом, так и при инструментальном обусловливании. Первоначально он относился только к усилению характера поведения, но часто употребляется и тогда, когда речь идет об ослаблении характера поведения. Стимул, который считается причиной изменения поведения, называют подкрепителем вне зависимости от того, контингентен он предыдущему поведению животного или нет. В конце этой главы мы более подробно обсудим эту терминологию и ее связь с терминологией, употребляемой в машинном обучении.

14.2. КЛАССИЧЕСКОЕ ОБУСЛОВЛИВАНИЕ

Изучая работу пищеварительной системы, знаменитый русский физиолог Иван Павлов обнаружил, что врожденные реакции на некоторые раздражители можно вызвать с помощью других раздражителей, никак не связанных с врожденными спусковыми механизмами. Павлов ставил эксперименты на собаках, подвергнутых незначительной хирургической операции, чтобы можно было точно измерить интенсивность слюнного рефлекса. В одном описанном им случае у собаки при большинстве условий не происходило выделение слюны, но спустя 5 секунд после предложения пищи выделялось примерно шесть капель слюны в течение следующих нескольких секунд. После нескольких повторных подач другого стимула, не связанного с едой, в данном случае звука метронома, раздававшегося незадолго до принятия пищи, собака начала выделять слюну в ответ на звук метронома точно так же, как на пищу. «Таким образом, активность слюнной железы была вызвана звуковыми колебаниями – раздражителем, никак не связанным с пищей» (Pavlov, 1927, p. 22). Подводя итоги значению этого открытия, Павлов писал:

¹ Для нас управление означает нечто отличное от того, что обычно понимается под этим словом в теориях обучения животных; там окружающая среда управляет агентом, а не наоборот. См. наши замечания по терминологии в конце этой главы.

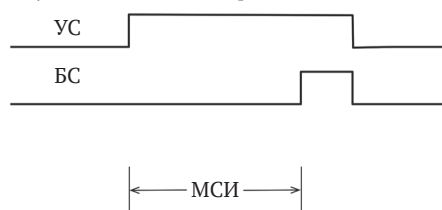
Довольно очевидно, что в естественных условиях нормальное животное должно реагировать не только на раздражители, которые приносят непосредственное удовлетворение или вред, но и на другие физические или химические агенты – звуковые и световые волны и т. п., – которые сами по себе только сигнализируют о приближении этих раздражителей; не вид и не звуки, издаваемые хищником, опасны для более мелких животных, а его клыки и когти (Pavlov, 1927, p. 14).

Такое соединение новых стимулов с врожденными рефlekсами теперь называют классическим, или павловским, обусловливанием. Павлов называл врожденные рефlekсы (например, слюноотделение в описанном выше эксперименте) «безусловными рефlekсами» (БР), естественные стимулы, запускаящие их срабатывание (например, пищу), – «безусловными раздражителями» (БС – безусловные стимулы), а новые рефlekсы, вызываемые предвещающими стимулами (например, то же слюноотделение), – «условными рефlekсами» (УР). Стимул, который первоначально нейтрален, т. е. не вызывает сильной реакции (например, звук метронома), становится «условным раздражителем» (УС – условный стимул), если животное обучается тому, что он предвещает БС, и начинает проявлять УР в ответ на УС. Эти термины до сих пор используются при описании экспериментов по классическому обусловливанию. БС называется подкрепителем, потому что подкрепляет проявление УР в ответ на УС.

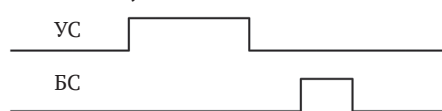
На рисунке справа показана конфигурация стимулов в двух стандартных типах экспериментов по классическому обусловливанию. При *обусловливании с задержкой* УС распространяется на весь межстимульный интервал (МСИ), т. е. время между началом УС и началом БС (в самом распространенном варианте УС заканчивается тогда же, когда БС). При *следовом обусловливании* БС начинается после завершения УС, а временной интервал между началом БС и началом УС называется следовым интервалом.

Слюноотделение собаки Павлова в ответ на звук метронома – лишь один пример классического обусловливания, которое интенсивно изучалось на разных системах реагирования у разных видов животных. БР часто являются подготовительными, как слюноотделение собаки Павлова, или защитными, как моргание в ответ на что-то, раздражающее глаз, или оцепенение в ответ на вид хищника. Опыт, состоящий в предсказуемой связи между УС и БС на протяжении ряда испытаний, побуждает животное обучиться тому, что УС предвещает БС, поэтому животное может отреагировать на УС условным рефlekсом, который подготавливает животное или защищает его от предсказанного БС. Некоторые УР похожи на БР, но начинаются раньше и имеют отличия, повышающие их эффективность. Например, в одном типе экспериментов, подвергшемся всестороннему изучению, звуковой УС точно предсказывает воздушный удар (БС) в глаз кролика и вызывает БР – закрытие защитного внутреннего века, которое

Обусловливание с задержкой



Следовое обусловливание



называется мигательной перепонкой. После одного или нескольких испытаний звуковой сигнал вызывает срабатывание УР, заключающегося в закрытии перепонки еще до воздушного удара, и в конечном итоге максимальное закрытие происходит как раз в тот момент, когда ожидается удар. Этот УР, начинающийся в предвидении воздушного удара и соответствующим образом синхронизированный, дает лучшую защиту, чем просто начало закрытия в ответ на раздражающий БС. Способность действовать в предвосхищении важных событий путем обучения предсказуемым связям между стимулами настолько полезна, что получила широкое распространение в животном царстве.

14.2.1. Блокирующее обусловливание и обусловливание высшего порядка

В экспериментах наблюдалось много интересных свойств классического обусловливания. Помимо предсказательной природы БР, в разработке моделей классического обусловливания видную роль сыграли два часто наблюдаемых свойства: блокирующее обусловливание и обусловливание высшего порядка. Блокировка возникает, когда животное не может обучиться УР, если потенциальный УС подается вместе с другим УС, который ранее использовался для выработки у животного этого УР. Например, на первом этапе эксперимента по блокировке с обусловливанием закрытия мигательной перепонки у кролика сначала вырабатывался условный рефлекс на звуковой УС в сочетании с БР на воздушный удар. На втором этапе включались дополнительные испытания, в которых к звуковому сигналу добавлялся второй стимул, скажем световой сигнал, с целью сформировать составной свето-звуковой УС, сопровождаемый тем же БС – воздушным ударом. На третьем этапе кролику подавался только второй стимул – световой сигнал, чтобы узнать, научился ли кролик реагировать на него условным рефлексом. Оказывается, что кролик очень редко отвечает УС на свет или вообще не отвечает: обучение световому сигналу заблокировано предыдущим обучением звуковому сигналу¹. Подобные результаты, касающиеся блокировки, поставили под сомнение идею о том, что обусловливание зависит только от простого соседства во времени, т. е. что необходимое и достаточное условие выработки условного рефлекса – частое следование БС за УС с небольшим промежутком времени. В следующем разделе мы опишем *модель Рескорлы–Вагнера* (Rescorla and Wagner, 1972), в которой предложено авторитетное объяснение блокировки.

Обусловливание высшего порядка имеет место, когда ранее обусловленный УС выступает в роли БС при обусловливании другого, первоначально нейтрального стимула. Павлов описал эксперимент, в котором его помощник сначала выработал у собаки условный рефлекс слюноотделения на звук метронома, который предвещал БС кормления, как описано выше. За этим этапом последовало несколько

¹ Необходимо сравнение с контрольной группой, доказывающее, что именно предыдущее обусловливание звуковым сигналом несет ответственность за блокировку обучения световому сигналу. Для этого производятся испытания со свето-звуковым УС, но без предварительного обусловливания звуком. В этом случае обучение реакции на свет происходит беспрепятственно. Полный отчет об этой процедуре приведен в работе Moore and Schmajuk (2008).

испытаний, когда в поле зрения собаки помещался черный квадрат, к которому собака первоначально была равнодушна. Это сопровождалось звуком метронома, за которым *не* следовало кормление. После всего десяти испытаний собака начала выделять слюну просто при виде черного квадрата, хотя за этим никогда не следовало предложение пищи. Сам звук метронома играл роль БС при выработке условного рефлекса слюноотделения на УС черного квадрата. Это было обусловливание второго порядка. Если бы черный квадрат использовался как БС для выработки УР слюноотделения на еще один нейтральный УС, то мы имели бы обусловливание третьего порядка и т. д. Обусловливание высшего порядка, особенно выше второго, трудно продемонстрировать, отчасти потому, что подкрепитель высшего порядка теряет свою подкрепительную ценность из-за того, что во время испытаний не сопровождается оригинальным БС. Но при правильных условиях, например чередовании испытаний первого порядка с испытаниями высшего порядка или при подаче общего активирующего стимула, обусловливание выше второго порядка все-таки можно продемонстрировать. Как будет описано ниже, в *TD-модели классического обусловливания* используется важнейшая для нашего подхода идея бутстрэппинга с целью обобщить модель блокировки Рескорлы–Вагнера, включив в нее как предвосхищающую природу УР, так и обусловливание высшего порядка.

Инструментальное обусловливание высшего порядка также имеет место. В данном случае стимул, который устойчиво предсказывает первичное подкрепление, сам становится подкрепителем, а подкрепление считается первичным, если его свойство приносить вознаграждение или наказание встроено в животное в процессе эволюции. Предсказательный стимул становится вторичным подкрепителем или, более общо, *подкрепителем высшего порядка*, или *обусловленным*, – последний термин лучше отражает суть дела, когда предсказанный подкрепляющий стимул сам является вторичным или еще более высокого порядка. Обусловленный подкрепитель приносит *обусловленное подкрепление*: вознаграждение или наказание. Обусловленное подкрепление действует как первичное подкрепление, повышая стремление животного к поведению, которое приводит к обусловленному вознаграждению, и уменьшая стремление к поведению, которое приводит к обусловленному наказанию. (См. наши замечания в конце главы, в которых объясняются отличия нашей терминологии от принятой в психологии.)

Обусловленное подкрепление – ключевое явление, которое объясняет, например, почему мы работаем за деньги, выступающие в роли обусловленного подкрепителя, ценность которых определяется исключительно тем, что предположительно можно приобрести за них. В методах исполнитель–критик, описанных в разделе 13.5 (и обсуждаемых в контексте нейронаук в разделах 15.7 и 15.8), критик применяет TD-метод для оценивания стратегии исполнителя, а его оценки ценности служат обусловленным подкреплением для исполнителя, давая ему возможность улучшать свою стратегию. Эта аналогия с инструментальным обусловливанием высшего порядка помогает решить проблему распределения поощрения, упомянутую в разделе 1.7, потому что критик в каждый момент предлагает исполнителю вознаграждение, когда первичный сигнал вознаграждения задерживается. Мы обсудим это подробнее в разделе 14.4.

14.2.2. Модель Рескорлы–Вагнера

Рескорла и Вагнер создали свою модель преимущественно для объяснения блокировки. Ее основная идея заключается в том, что животное обучается, только когда события идут вразрез с его ожиданиями, иными словами, когда оно удивлено (хотя это необязательно выражается в каком-то *осознанном* ожидании или эмоции). Мы сначала представим модель Рескорлы–Вагнера с применением терминологии и обозначений авторов, а затем перейдем к терминологии и обозначениям, которые использовали при описании TD-модели.

Вот как описывали свою модель сами Рескорла и Вагнер. Модель корректирует «ассоциативную силу» каждого стимула, являющегося частью составного УС, а именно число, описывающее, насколько сильно или надежно этот компонент предсказывает БС. Когда составной УС, включающий несколько компонентов, подается в испытании классического обусловливания, ассоциативная сила каждого компонента изменяется в зависимости от ассоциативной силы, связанной со всей смесью компонентов, – так называемой «агрегированной ассоциативной силы», – а не только от ассоциативной силы, связанной с каждым компонентом в отдельности.

Рескорла и Вагнер рассматривали составной УС АХ, включающий компоненты А и Х, считая, что животное уже могло испытывать стимул А, а стимул Х для него, возможно, внове. Обозначим V_A , V_X и V_{AX} ассоциативные силы стимулов А, Х и составного стимула АХ соответственно. Предположим, что в некотором испытании за составным УС АХ следует БС, который мы обозначим Y. Тогда ассоциативные силы компонентных стимулов изменяются по следующим законам:

$$\Delta V_A = \alpha_A \beta_Y (R_Y - V_{AX});$$

$$\Delta V_X = \alpha_X \beta_Y (R_Y - V_{AX});$$

где $\alpha_A \beta_Y$ и $\alpha_X \beta_Y$ – параметры размера шага, зависящие от идентификаторов компонент УС и БС, а R_Y – асимптотический уровень ассоциативной силы, которую может поддерживать БС Y. (Рескорла и Вагнер использовали букву λ вместо R , но мы остановились на R , чтобы избежать путаницы с нашим употреблением λ и поскольку обычно интерпретируем эту величину как абсолютную величину сигнала вознаграждения, учитывая, что в классическом обусловливании БС необязательно является вознаграждением или наказанием.) Главное предположение модели заключается в том, что агрегированная ассоциативная сила V_{AX} равна $V_A + V_X$. Ассоциативные силы, измененные на показанные выше дельты, становятся ассоциативными силами в начале следующего испытания.

Для полноты картины модель должна включать механизм генерирования отклика, т. е. способ отображения значений V в УР. Поскольку это отображение должно зависеть от особенностей эксперимента, Рескорла и Вагнер не стали его определять, а просто предположили, что большие значения V должны порождать более сильные или более вероятные УР, а отрицательное V означает, что УР не вырабатывается.

Модель Рескорлы–Вагнера описывает выработку УР, объясняя блокировку. До тех пор пока агрегированная ассоциативная сила V_{AX} составного стимула меньше асимптотического уровня ассоциативной силы R_Y , который способен поддерживать

БС Y , ошибка предсказания $R_Y - V_{AX}$ положительна. Это означает, что в последующих испытаниях ассоциативные силы V_A и V_X компонентных стимулов увеличиваются, пока агрегированная ассоциативная сила V_{AX} не станет равна R_Y , после чего ассоциативные силы перестают изменяться (если только не изменяется БС). Когда в составной УС, которым животное уже обусловлено, добавляется новый компонент, последующее обусловливание пополненным составным стимулом приводит к меньшему увеличению или вообще не приводит к увеличению ассоциативной силы добавленного компонента УС, поскольку ошибка уже обратилась в ноль или стала очень мала. Появление БС уже предсказывается почти идеально, поэтому новый компонент УС не изменяет или мало изменяет ошибку, т. е. не вызывает удивления. Предшествующее обучение блокирует обучение новому компоненту.

Чтобы перейти от модели Рескорлы–Вагнера к TD-модели классического обусловливания (которую мы называем просто TD-моделью), мы сначала переформулируем их модель в терминах, используемых в этой книге. Точнее, мы соотнесем ее с нашей нотацией для обучения линейной аппроксимации функций (раздел 9.4), а процесс обусловливания будем интерпретировать как обучение предсказанию «абсолютной величины БС» после некоторого испытания на основе составного УС, предъявленного в этом испытании, где абсолютная величина БС Y совпадает с R_Y в модели Рескорлы–Вагнера. Мы введем также состояния. Модель Рескорлы–Вагнера – это модель *уровня испытаний*, т. е. речь в ней идет о том, как изменяется ассоциативная сила при переходе от одного испытания к другому без учета деталей происходящего в процессе испытаний и между ними. Поэтому нам необязательно рассматривать, как изменяются состояния во время испытаний, до тех пор, пока мы не представим полную TD-модель в следующем разделе. Вместо этого мы просто будем рассматривать состояние как способ пометить испытание в терминах набора предъявленных в нем компонентных УС.

Итак, предположим, что тип испытания, или состояние s , описывается вещественным вектором признаков $\mathbf{x}(s) = (x_1(s), x_2(s), \dots, x_d(s))^T$, где $x_i(s) = 1$, если УС _{i} , i -я компонента составного УС, предъявлена в испытании, и 0 в противном случае. Тогда если обозначить $\mathbf{w}$ d -мерный вектор ассоциативных сил, то агрегированная ассоциативная сила для типа испытания s равна

$$\hat{v}(s, \mathbf{w}) = \mathbf{w}^T \mathbf{x}(s). \quad (14.1)$$

Эта величина соответствует *оценке ценности* в обучении с подкреплением, и мы интерпретируем ее как *предсказание БС*.

Временно будем обозначать t номер завершеного испытания, забыв про его обычный смысл – временной шаг (мы вернемся к обычному пониманию t , когда обобщим это на TD-модель ниже), и предположим, что S_t – состояние, соответствующее испытанию t . Обусловливающее испытание t приводит к следующему обновлению вектора ассоциативных сил $\mathbf{w}_t$:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \delta_t \mathbf{x}(S_t), \quad (14.2)$$

где α – размер шага, а (поскольку здесь мы описываем модель Рескорлы–Вагнера) δ_t – *ошибка предсказания*:

$$\delta_t = R_t - \hat{v}(S_t, \mathbf{w}_t). \quad (14.3)$$

R_t – цель предсказания в испытании t , т. е. абсолютная величина БС, или, в терминах Рескорлы и Вагнера, ассоциативная сила, которую может поддержать БС в этом испытании. Заметим, что из-за множителя $x(S_t)$ в уравнении (14.2) в результате испытания корректируются только ассоциативные силы компонентов УС, предъявленных в испытании. Можно считать ошибку предсказания мерой удивления, а агрегированную ассоциативную силу – ожиданиями животного, которые не оправдываются, если не совпадают с целевой абсолютной величиной БС.

С точки зрения машинного обучения, модель Рескорлы–Вагнера представляет собой правило обучения с учителем для исправления ошибок. По существу, это то же, что правило наименьших средних квадратов (LMS), или правило Видроу–Хоффа (Widrow and Hoff, 1960) для нахождения весов – в данном случае ассоциативных сил – таких, что среднее квадратов всех ошибок как можно ближе к нулю. Это алгоритм «аппроксимации кривой», или регрессии, который широко применяется в научно-технических приложениях (см. раздел 9.4)¹.

Модель Рескорлы–Вагнера оказала очень большое влияние на теорию обучения животных, поскольку продемонстрировала, что «механистическая» теория способна объяснить основные факты, относящиеся к блокировке, не прибегая к более сложным когнитивным теориям, в т. ч. идее о том, что животное явно распознает добавление нового стимула, а затем с помощью поиска в своей краткосрочной памяти переоценивает прогностические связи, в которых участвует УС. Модель Рескорлы–Вагнера показала, как традиционные теории обучения, основанные на принципе смежности, которые утверждают, что смежность стимулов во времени является необходимым и достаточным условием обучения, можно без особых усилий дополнить с целью учета блокировки (Moore and Schmajuk, 2008).

Модель Рескорлы–Вагнера дает простое объяснение блокировки и некоторых других особенностей классического обусловливания, но не является ни полной, ни совершенной моделью такого обусловливания. Для объяснения ряда иных наблюдаемых эффектов нужны другие идеи, поэтому прогресс на пути понимания многих тонкостей классического обусловливания все еще продолжается. TD-модель классического обусловливания, которую мы опишем ниже, тоже неполна и несовершенна, но она обобщает модель Рескорлы–Вагнера, помогая понять, как временные связи между стимулами в процессе испытания и между испытаниями могут повлиять на обучение и как возникает обусловливание высшего порядка.

14.2.3. TD-модель

TD-модель – это модель *реального времени*, а не уровня испытаний, как модель Рескорлы–Вагнера. Один шаг t в модели Рескорлы–Вагнера представляет полное обусловливающее испытание. Модель неприменима к деталям происходящего в процессе испытания или между испытаниями. В каждом испытании животное может подвергаться различным стимулам, которые начинаются в определенные

¹ Единственными различиями между правилом LMS и моделью Рескорлы–Вагнера является то, что для LMS элементами входных x_t векторов могут быть произвольные вещественные числа и – по крайней мере, в простейшей форме правила LMS – размер шага α не зависит от входного вектора или идентификатора стимула, отвечающего за цель предсказания.

моменты и продолжаются определенное время. Эти временные соотношения сильно влияют на обучение. В модели Рескорлы–Вагнера также нет механизма обусловливания высшего порядка, тогда как в TD-модели обусловливание высшего порядка – естественное следствие идеи бутстрэппинга, лежащей в основе TD-алгоритмов.

Чтобы описать TD-модель, начнем с описанной выше формулировки модели Рескорлы–Вагнера, но теперь пусть t обозначает временные шаги внутри или между испытаниями, а не номер полного испытания. Будем считать, что интервал времени между t и $t + 1$ мал, скажем 0.01 с, а испытание будем представлять себе как последовательность состояний, ассоциированных с временными шагами; состояние в момент t теперь описывает детали представления стимулов в этот момент, а не просто является меткой для компонентов УС, предъявленных в испытании. Фактически мы можем вообще отказаться от идеи испытаний. С точки зрения животного, испытание – просто эпизод в его непрерывном взаимодействии с миром. Следуя нашему обычному представлению об агенте, взаимодействующем с окружающей средой, мы можем считать, что животное опытным путем перебирает бесконечную последовательность состояний s , каждое из которых представлено вектором признаков $x(s)$. Впрочем, все равно часто удобнее рассматривать испытания как интервалы времени, в течение которых комбинации стимулов повторяются в эксперименте.

Признаки состояний не ограничиваются описанием внешних стимулов, испытываемых животным; они могут описывать паттерны нервной активности, возникающие в мозге животного в ответ на стимулы, и эти паттерны могут зависеть от истории, т. е. быть устойчивыми откликами на последовательности стимулов. Разумеется, мы точно не знаем, как выглядят эти паттерны нервной активности, но модель реального времени типа TD-модели позволяет исследовать последствия, обучаясь различным гипотезам о внутреннем представлении внешних стимулов. По указанным причинам TD-модель не связывает себя конкретным представлением состояний. Кроме того, поскольку TD-модель включает обесценивание и следы приемлемости, охватывающие временные интервалы между стимулами, она открывает возможность исследовать, как обесценивание и следы приемлемости взаимодействуют с представлениями стимулов для предсказания результатов экспериментов по классическому обусловливанию.

Далее мы опишем некоторые представления состояний, используемые в TD-модели, и ряд вытекающих из них следствий. Но пока забудем о деталях представления, а просто предположим, что каждое состояние s представлено вектором признаков $x(s) = (x_1(s), x_2(s), \dots, x_n(s))^T$. Тогда агрегированная ассоциативная сила, соответствующая состоянию s , определяется выражением (14.1), таким же, как в модели Рескорлы–Вагнера, но обновление вектора ассоциативных сил w в TD-модели производится по-другому. Поскольку t теперь обозначает момент времени, а не полное испытание, TD-модель управляет обучением в соответствии с таким правилом обновления:

$$w_{t+1} = w_t + \alpha \delta_t z_t, \quad (14.4)$$

Здесь $x_t(S_t)$, встречающееся в обновлении Рескорлы–Вагнера (14.2), заменено вектором следов приемлемости z_t , а δ_t теперь определено не как в (14.3), а является TD-ошибкой:

$$\delta_t = R_{t+1} + \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t), \quad (14.5)$$

где γ – коэффициент обесценивания (от 0 до 1), R_t – цель предсказания в момент t , а $\hat{v}(S_{t+1}, \mathbf{w}_t)$ и $\hat{v}(S_t, \mathbf{w}_t)$ – агрегированные ассоциативные силы в моменты $t + 1$ и t , определенные выражением (14.1).

Каждый элемент i вектора следов приемлемости $\mathbf{z}_t$ увеличивается или уменьшается в соответствии с элементом $x_i(S_t)$ вектора признаков $\mathbf{x}(S_t)$, а в противном случае затухает со скоростью, определяемой произведением $\gamma\lambda$:

$$\mathbf{z}_{t+1} = \gamma\lambda\mathbf{z}_t + \mathbf{x}(S_t). \quad (14.6)$$

Здесь λ – обычный параметр затухания следа приемлемости.

Заметим, что при $\gamma = 0$ TD-модель сводится к модели Рескорлы–Вагнера со следующими отличиями: смысл t различен (номер испытания в модели Рескорлы–Вагнера и временной шаг в TD-модели), и в TD-модели имеет место опережение на один шаг в цели предсказания R . TD-модель эквивалентна обратному представлению полугradientного алгоритма TD(λ) с линейной аппроксимацией функций (глава 12) с тем отличием, что R_t в этой модели необязательно является сигналом вознаграждения, как в случае, когда TD-алгоритм применяется для обучения функции ценности с целью улучшения стратегии.

14.2.4. Имитирование TD-модели

Модели обусловливания в реальном времени, в частности TD-модель, интересны тем, что дают предсказания для широкого круга ситуаций, которые нельзя представить моделями уровня испытаний. К их числу относятся время подачи и продолжительности допускающих обусловливание стимулов, время подачи этих стимулов относительно времени подачи БС, а также время и формы УР. Например, БС, вообще говоря, должен начинаться после начала нейтральных стимулов, если мы хотим добиться обусловливания, причем скорость и эффективность обучения зависят от межстимульного интервала (МСИ) – интервала между началами УС и БС. Условные стимулы, вообще говоря, начинаются до БС, а их временные характеристики изменяются в процессе обучения. При обусловливании составным УС все его компоненты необязательно должны начинаться и заканчиваться в одно и то же время; иногда подается так называемый *серийный составной стимул*, компоненты которого следуют друг за другом. Из-за подобных временных соотношений важно учитывать, как представлены стимулы, как эти представления разворачиваются во времени в процессе испытаний и между ними и как они взаимодействуют с обесцениванием и следами приемлемости.

На рис. 14.1 показаны три представления стимулов, которые использовались при исследовании поведения TD-модели: *полное серийное составное* представление (ПСС), *микростимульное* представление (МС) и представление *присутствия* (Ludvig, Sutton, and Kehoe, 2012). Они различаются по степени обобщения между близкими моментами времени, когда стимул присутствует.

Самое простое из представлений на рис. 14.1 – представление присутствия, показанное в правом столбце. В этом представлении имеется по одному признаку для каждого компонентного УС, предъявленного во время испытания, и этот при-

знак равен 1, если компонент присутствует, и 0 в противном случае¹. Представление присутствия – не реалистичная гипотеза о представлении стимулов в мозге животного, но, как будет описано ниже, TD-модель с таким представлением может порождать многие феномены синхронизации, наблюдаемые в классическом обусловливании.

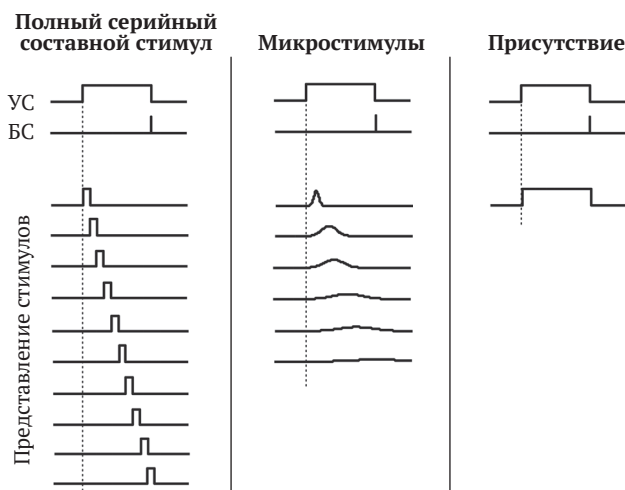


Рис. 14.1 ❖ Три представления стимулов (столбцы), которые иногда используются совместно с TD-моделью. В каждой строке показан один элемент представления стимула. Представления отличаются временным градиентом обобщения: в полном серийном составном представлении (левый столбец) обобщения между близкими во времени точками нет вовсе, а в представлении присутствия (правый столбец) обобщение полное. Микростимульное представление занимает промежуточное положение. Степень временного обобщения определяет, с какой временной гранулярностью животное обучается предсказывать БС. Заимствовано из работы E. A. Ludvig, R. S. Sutton, E. J. Kehoe «Learning & Behavior, Evaluating the TD Model of Classical Conditioning» с разрешения издательства Springer

Для ПСС-представления (левый столбец на рис. 14.1) начало каждого внешнего стимула инициирует последовательность точно позиционированных во времени коротких внутренних сигналов, которая продолжается до завершения внешнего стимула². Это все равно, что предположить, что в нервной системе животного

¹ В нашем формализме имеются разные состояния S_t для каждого временного шага t во время испытания. А для испытания, в котором составной УС включает компонентные УС различной продолжительности, подаваемые в разные моменты на протяжении испытания, существует признак x_i для каждого компонента УС, $i = 1, \dots, n$, где $x_i(S_t) = 1$ для тех моментов t , когда УС _{i} присутствует, и 0 в противном случае.

² В нашем формализме для каждой компоненты УС, предъявленной во время испытания, и для каждого временного шага t в процессе испытания существует отдельный признак x_i^t такой, что $x_i^t(S_t) = 1$, если $t = t'$, для любого момента t' , в котором УС _{i} присутствует, и 0 в противном случае. Это отличается от ПСС-представления в работе Sutton and Barto (1990), в котором на каждом временном шаге имеются одни и те признаки, но нет никакой ссылки на внешний стимул; отсюда и название *полное серийное составное*.

имеются часы, которые точно отмеряют время предъявления стимула; инженеры называют это «линией задержки с отводами». Как и представление присутствия, ПСС-представление нереалистично в качестве гипотезы о том, как сам мозг представляет стимулы, но в работе Ludvig et al. (2012) оно названо «полезной фикцией», поскольку может раскрыть детали работы TD-модели, не слишком ограниченной представлением стимулов. ПСС-представление используется также в большинстве TD-моделей нейронов головного мозга, продуцирующих дофамин; эту тему мы рассмотрим в главе 15. Кроме того, ПСС-представление нередко рассматривают как существенную часть TD-модели, хотя эта точка зрения ошибочна.

МС-представление (средний столбец на рис. 14.1) похоже на ПСС-представление тем, что внешний стимул инициирует каскад внутренних, но в этом случае внутренние стимулы – микростимулы – необязательно имеют такую прямоугольную форму без перекрытия; они растянуты во времени и могут перекрываться. По мере отдаления от начала стимула различные множества микростимулов могут становиться более или менее активными, и каждый последующий микростимул становится все шире, а его максимальный уровень – все меньше. Конечно, существует много МС-представлений, зависящих от природы микростимулов, и в литературе изучен ряд примеров МС-представлений, иногда вместе с предположениями о том, как их мог бы порождать мозг животного (см. «Библиографические и исторические замечания» в конце этой главы). МС-представления выглядят более реалистично, чем представления присутствия или ПСС-представления, в роли гипотез о представлении стимулов в нервной системе и позволяют соотнести поведение TD-модели с более широким кругом явлений, наблюдаемых в экспериментах с животными. В частности, благодаря предположению о том, что каскады микростимулов инициируются в равной мере БС и УС, а также изучению важного влияния на обучение взаимодействиям между микростимулами, следами приемлемости и обесцениванием TD-модель помогает высказывать гипотезы, которые объясняют многие тонкие феномены классического обусловливания и то, как мозг животного мог бы продуцировать их. Мы еще вернемся к этой теме, особенно в главе 15, где будем обсуждать связь между обучением с подкреплением и нейронауками.

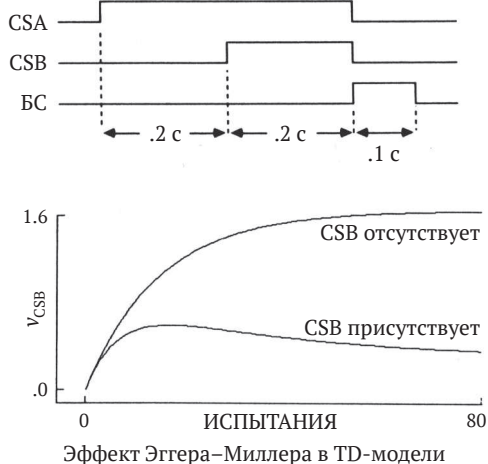
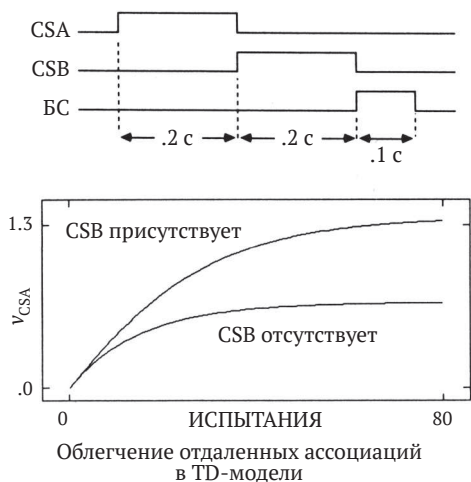
Однако даже для простейшего представления присутствия TD-модель порождает все основные свойства классического обусловливания, объясняемые моделью Рескорлы–Вагнера, а также особенности обусловливания, выходящие за рамки моделей уровня испытаний. Например, мы уже отмечали одну заметную особенность классического обусловливания – чтобы обусловливание произошло, БС, вообще говоря, должен начинаться после подачи нейтральных стимулов, а после обусловливания УР начинает проявляться до подачи БС. Иными словами, для обусловливания обычно необходим положительный МСИ, и УР обычно предшествует БС. Степень зависимости силы обусловливания (например, процента УР, вызванных УС) от МСИ существенно меняется для разных видов и систем реагирования, но, как правило, она обладает следующими свойствами: пренебрежимо мала для нулевого или отрицательного МСИ, т. е. когда начало БС совпадает или предшествует началу УС (хотя в некоторых исследованиях обнаружено, что иногда ассоциативная сила немного увеличивается или становится отрицательной при отрицательном МСИ); возрастает до максимума при положительном МСИ, когда обусловливание наиболее эффективно, а затем убывает до нуля спустя вре-

мя, которое сильно зависит от системы реагирования. Точная форма этой зависимости для TD-модели определяется значениями параметров и деталями представления стимулов, но сама по себе зависимость от МСИ относится к числу фундаментальных свойств TD-модели.

В связи с серийным составным обусловливанием, т. е. составным УС, компоненты которого подаются последовательно, возникает, в частности, один теоретический вопрос, касающийся облегчения отдаленных ассоциаций. Установлено, что если пустой следовый интервал между первым УС (CSA) и БС заполнен вторым УС (CSB), так что образуется серийный составной стимул, то выработка условного рефлекса на CSA облегчается. На рисунке справа изображено поведение TD-модели с представлением присутствия в имитации такого эксперимента с показанными на диаграмме деталями синхронизации. В полном соответствии с экспериментальными результатами (Kehoe, 1982) модель демонстрирует улучшение скорости и асимптотического уровня обусловливания первым УС благодаря присутствию второго УС.

Хорошо известной демонстрацией влияния временных соотношений между стимулами в испытании на обусловливание является эксперимент Эггера и Миллера (Egger and Miller, 1962), в котором подавались два перекрывающихся УС с конфигурацией задержки, показанной на рисунке справа вверху. Хотя CSB находился в более выгодном временном соотношении с БС, присутствие CSA значительно ослабило выработку условного рефлекса на CSB по сравнению с контрольными экспериментами, в которых CSA отсутствовал. В нижней части рисунка показан тот же результат, сгенерированный TD-моделью в ходе имитации этого эксперимента с представлением присутствия.

TD-модель объясняет блокировку, потому что это правило обучения, исправляющее ошибку, как и модель Рескорлы–Вагнера. Но она не только объясняет основные факты, относящиеся к блокировке, но и предсказывает (с представлением присутствия и более сложными), что блокировка меняет направление, если блокированный стимул перемещается на более раннее время, так что его начало предшествует началу блокирующего стимула (как CSA на рис. 14.2). Эта особенность поведения TD-модели заслуживает внимания, потому что она не была замечена в момент, когда эта модель появилась. Напомним, в чем состоит суть блокировки:



если животное уже обучилось тому, что один УС предвещает БС, то эффективность обучения тому, что другой УС также предвещает БС, сильно ослаблена, т. е. заблокирована. Но если второй УС начинается раньше, чем уже обученный, то – согласно TD-модели – обучение новому УС не блокируется. На самом деле по мере продолжения обучения и приобретения ассоциативной силы вновь добавленным УС ранее обученный УС теряет ассоциативную силу. Поведение TD-модели при таких условиях показано в нижней части рис. 14.2. Этот имитационный эксперимент отличался от эксперимента Эггера–Миллера (показан в нижней части предыдущего рисунка) тем, что более короткий УС, начинавшийся позже, подавался до обучения, пока не сформировалась полная ассоциация с БС. Это неожиданное предсказание навело Кехоу, Шройрса и Грэма (Kehoe, Schreurs, and Graham, 1987) на мысль поставить эксперимент с использованием хорошо изученной подготовки мигательной перепонки у кролика. Их результаты подтвердили предсказание модели, и было отмечено, что модели, отличные от основанной на TD, испытывают серьезные трудности при объяснении полученных данных.

В TD-модели более ранний предвещающий стимул имеет приоритет над более поздним, потому что, как и все методы предсказания, описанные в этой книге, TD-модель основана на идее обратного обновления, или бутстрэппинга: обновления ассоциативных сил сдвигают силы в конкретном состоянии в направлении силы в более поздних состояниях. Еще одно следствие бутстрэппинга заключается в том, что TD-модель объясняет обусловливание высшего порядка – особенность классического обусловливания, которая выходит за рамки модели Рескорлы–Вагнера и ей подобных. Как было описано выше, обусловливание высшего порядка состоит в том, что ранее обусловленный УС может выступать в роли БС при обусловливании другим, первоначально нейтральным стимулом. На рис. 14.3 показано поведение TD-модели (снова в представлении присутствия) в эксперименте по обусловливанию высшего порядка – в данном случае второго. На первом этапе (на рисунке не показан) CSB обучается предсказывать БС, так что его ассоциативная сила увеличивается, в данном случае до 1.65. На втором этапе CSA подается в паре с CSB в отсутствие БС – в последовательной конфигурации, показанной в верхней части рисунка. CSA

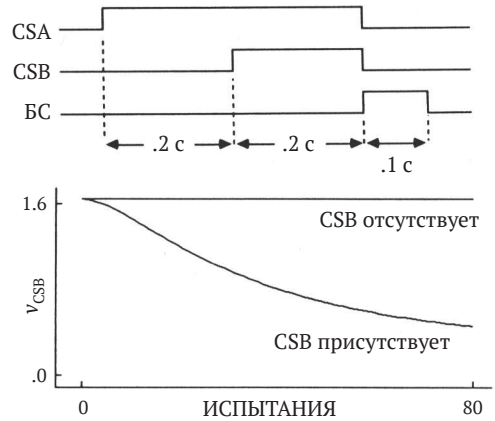


Рис. 14.2 ♦ Первичность во времени подавляет блокировку в TD-модели

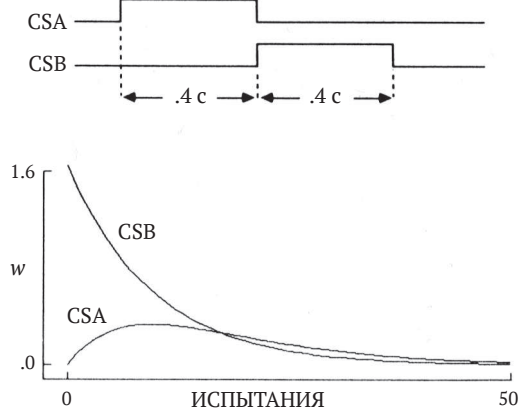


Рис. 14.3 ♦ Обусловливание второго порядка в TD-модели

было описано выше, обусловливание высшего порядка состоит в том, что ранее обусловленный УС может выступать в роли БС при обусловливании другим, первоначально нейтральным стимулом. На рис. 14.3 показано поведение TD-модели (снова в представлении присутствия) в эксперименте по обусловливанию высшего порядка – в данном случае второго. На первом этапе (на рисунке не показан) CSB обучается предсказывать БС, так что его ассоциативная сила увеличивается, в данном случае до 1.65. На втором этапе CSA подается в паре с CSB в отсутствие БС – в последовательной конфигурации, показанной в верхней части рисунка. CSA

приобретает ассоциативную силу, хотя он никогда не подавался в паре с БС. При продолжении обучения ассоциативная сила CSA достигает максимума, а затем спадает, потому что спадает ассоциативная сила CSB, вторичного подкрепителя, и он теряет возможность обеспечивать вторичное подкрепление. Ассоциативная сила CSB уменьшается, потому что БС в этих испытаниях обусловливания высшего порядка вообще не появлялся. Это *испытания затухания* для CSB, потому что его прогностическая связь с БС нарушена, и, стало быть, способность выступать в роли подкрепителя снижена. Та же закономерность наблюдается в экспериментах с животными. Это затухание обусловленного подкрепления в испытаниях обусловливания высшего порядка затрудняет демонстрацию обусловливания высшего порядка, если только не возобновлять уже установленные прогностические связи, периодически вставляя испытания первого порядка.

TD-модель порождает аналог обусловливания второго и более высокого порядка, потому что $\hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_t)$ входит в TD-ошибку δ_t (14.5). Это означает, что в результате предыдущего обучения $\hat{v}(S_{t+1}, w_t)$ может отличаться от $\hat{v}(S_t, w_t)$, вследствие чего δ_t (временное различие) будет ненулевым. Эта разность имеет такой же статус, как R_{t+1} в (14.5), откуда следует, что в том, что касается обучения, между временным различием и присутствием БС нет никакой разницы. На самом деле эта особенность TD-алгоритма – одна из основных причин его разработки, которую мы теперь воспринимаем в свете его связей с динамическим программированием, описанных в главе 6. Бутстрэппинг ценностей тесно связан с обусловливанием второго и более высокого порядка.

В вышеописанных примерах поведения TD-модели мы интересовались только изменениями ассоциативной силы компонентов УС; мы не изучали, как модель предсказывает свойства выработанных у животного условных рефлексов (УР): их временные характеристики, форму и развитие на протяжении испытаний. Эти свойства зависят от вида животного, наблюдаемой системы реагирования и параметров испытаний обусловливания, но во многих экспериментах с разными животными и разными системами реагирования абсолютная величина УР или вероятность УР возрастает по мере приближения времени УС. Например, в классическом обусловливании реакции мигательной перепонки у кролика время от начала УС до начала движения мигательной перепонки уменьшается в каждом последующем испытании, а амплитуда этого упреждающего закрытия постепенно увеличивается на протяжении интервала между УС и БС, достигая максимума в момент ожидаемого наступления БС. Время и форма этого УС критичны с точки зрения способности к адаптации: если закрыть глаз слишком рано, то пострадает зрение (хотя мигательная перепонка полупрозрачна), а если слишком поздно, то будет утрачена защитная ценность этого действия. Улавливание подобных особенностей УС представляет сложность для моделей классического обусловливания.

В определение TD-модели не входит никакой механизм преобразования динамики предсказания БС, $\hat{v}(S_t, w_t)$, в профиль, который можно было бы сравнить со свойствами УС животного. Простейший способ – уравнивать динамику имитированного УС с динамикой предсказания БС. В таком случае признаки имитированных УС и характер их изменения в испытаниях зависят только от выбранного представления стимула и значений параметров модели α , γ и λ .

На рис. 14.4 показана динамика предсказаний БС в различные моменты обучения при каждом из трех представлений на рис. 14.1. В этих имитациях БС подавал-

ся спустя 25 временных шагов после начала УС и $\alpha = 0.05$, $\gamma = 0.95$, $\lambda = 0.97$. В случае ПСС-представления (рис. 14.4 слева) кривая предсказания БС, построенная TD-моделью, экспоненциально возрастает на всем интервале от УС до БС и достигает максимума точно в момент БС (на временном шаге 25). Этот экспоненциальный рост является результатом обесценивания в правиле обучения TD-модели. В случае представления присутствия (рис. 14.4 в центре) кривая предсказания БС почти постоянна на протяжении всего времени присутствия стимула, поскольку для каждого стимула нужно обучиться всего одному весу, или ассоциативной силе. Следовательно, TD-модель с представлением присутствия не может воспроизвести многие особенности синхронизации УС. В случае МС-представления (рис. 14.4 справа) динамика предсказания БС TD-моделью сложнее. После 200 испытаний профиль предсказания является разумной аппроксимацией кривой предсказания БС, полученной для ПСС-представления.

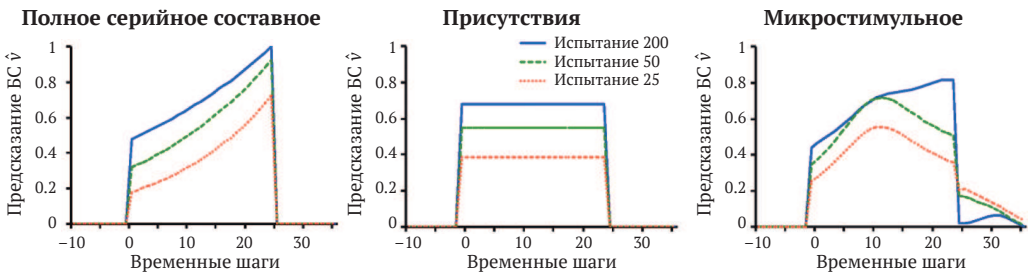


Рис. 14.4 ❖ Динамика предсказания БС по мере поступления данных в TD-модель с тремя разными представлениями стимулов. **Слева:** в случае полного серийного составного представления (ПСС) предсказание БС экспоненциально возрастает на всем интервале, достигая максимума в момент БС. В пределе (испытание 200) предсказание БС совпадает с интенсивностью БС (1 в имитациях). **В центре:** в случае представления присутствия предсказание БС сходится почти к постоянному уровню, который определяется интенсивностью БС и длиной интервала УС-БС. **Справа:** в случае микростимульного представления в пределе TD-модель аппроксимирует экспоненциально растущую динамику, характерную для ПСС, с помощью линейной комбинации различных микростимулов. Заимствовано с небольшими изменениями из работы E. A. Ludvig, R. S. Sutton, E. J. Kehoe «Learning & Behavior, Evaluating the TD Model of Classical Conditioning» с разрешения издательства Springer

Не предполагалось, что кривые предсказания БС, изображенные на рис. 14.4, должны точно совпадать с профилями УР, получающимися в процессе обусловливания в каком-то конкретном эксперименте с животными. Однако они иллюстрируют, какое сильное влияние оказывает представление стимулов на предсказания TD-модели. Кроме того, хотя здесь мы об этом упоминаем только мимоходом, характер взаимодействия между представлением стимулов, с одной стороны, и обесцениванием и следами приемлемости – с другой, важен для определения свойств, которыми обладают профили предсказания БС, порождаемые TD-моделью. Еще один аспект, оставшийся за рамками нашего обсуждения, – влияние различных механизмов генерации отклика, которые преобразуют предсказания БС в профили УР; на рис. 14.4 показаны «исходные» профили предсказания БС. Но даже без специальных предположений о том, как мозг животного может продуцировать

наблюдаемый отклик по предсказаниям БС, профили на рис. 14.4 в случае представлений ПСС и МС возрастают по мере приближения к моменту БС и достигают максимума в этот момент – точно так, как наблюдалось во многих экспериментах по обусловливанию животных.

TD-модель в сочетании с конкретными представлениями стимулов и механизмами генерации откликов способна объяснить на удивление широкий круг явлений, наблюдаемых в экспериментах по классическому обусловливанию, но и она далека от совершенства. Для объяснения других деталей классического обусловливания эту модель необходимо расширить, быть может, добавив элементы и механизмы для адаптивного изменения некоторых параметров. Другие подходы к моделированию классического обусловливания существенно расходятся с процессом исправления ошибки Рескорлы–Вагнера. Например, в байесовских вероятностных моделях опыт используется для пересмотра оценок вероятностей. Все эти модели улучшают наше понимание классического обусловливания.

Быть может, самой заметной особенностью TD-модели является то, что она основана на теории – той, что описана в этой книге, – которая предлагает объяснение того, что *пытается сделать* нервная система животного в процессе обусловливания: она пытается сформировать точные *долгосрочные предсказания*, совместимые с ограничениями, которые налагают способ представления стимулов и механизмы работы нервной системы. Иными словами, теория предлагает нормативное объяснение классического обусловливания, в котором основное место занимает долгосрочное, а не сиюминутное предсказание.

Разработка TD-модели классического обусловливания – пример, в котором явной целью было моделирование некоторых деталей поведения животных в процессе обучения. Стало быть, TD-обучение является фундаментом не только *алгоритма*, но и этой *модели* аспектов биологического обучения. В главе 15 мы увидим, что TD-обучение, ко всему прочему, легло в основу влиятельной модели функционирования нейронов, продуцирующих дофамин – химическое вещество в мозге млекопитающих, играющее важную роль в обработке вознаграждения. Это примеры того, как теория обучения с подкреплением тесно соприкасается с данными о нервной деятельности и поведении животных.

Теперь мы перейдем к рассмотрению параллелей между обучением с подкреплением и поведением животных в экспериментах по инструментальному обусловливанию, основному типу лабораторных экспериментов, с помощью которых психологи изучают обучение животных.

14.3. ИНСТРУМЕНТАЛЬНОЕ ОБУСЛОВЛИВАНИЕ

В экспериментах по *инструментальному обусловливанию* обучение зависит от последствий поведения: доставка подкрепляющего стимула зависит от того, что делает животное. Напротив, в экспериментах по классическому обусловливанию подкрепляющий стимул – БС – доставляется независимо от поведения животного. Обычно считается, что инструментальное обусловливание – то же самое, что *оперантное обусловливание* – этот термин Скиннер (B. F. Skinner, 1938, 1963) ввел в экспериментах с подкреплением, зависящим от поведения, – хотя эксперименты и теории ученых, употребляющих эти два термина, различаются во многих

отношениях, на некоторых из которых мы остановимся ниже. Своими корнями инструментальное обусловливание уходит в эксперименты американского психолога Эдварда Торндайка, поставленные за сто лет до публикации первого издания этой книги.

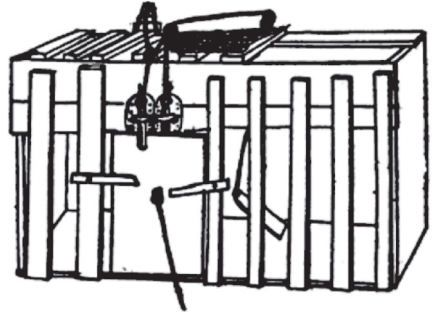
Торндайк наблюдал поведение кошек, помещенных в «проблемный ящик» типа показанного на рисунке справа, из которого они могли выйти, совершив правильные действия. Например, кошка могла открыть дверь одного ящика, выполнив последовательность трех действий: нажать на педаль в задней части ящика, вытянуть цепочку, подцепив ее когтями, и нажать на рычаг, подняв или опустив его. Будучи впервые помещены в ящик, из которого была видна еда, почти все кошки Торндайка демонстрировали «очевидные признаки дискомфорта» и хаотичное поведение, пытаясь «инстинктивно выбраться из заключения» (Thorndike, 1898).

В экспериментах с участием разных кошек и ящиков с разными замками Торндайк фиксировал время, которое понадобилось каждой кошке, чтобы выбраться из ящика в каждой из нескольких попыток. Он заметил, что почти во всех случаях время в каждой последующей попытке уменьшалось, например, с 300 до 6–7 секунд. Поведение кошек в проблемном ящике он описывал следующим образом:

Пытаясь выбраться, кошка импульсивно царапает всю поверхность ящика и, вероятно, случайно задевает цепочку, петлю или рычаг, открывающий дверь. Постепенно все прочие не приведшие к успеху действия отбрасываются, а то, которое оказалось успешным, фиксируется в сознании и связывается с полученным вслед за ним удовольствием. В конце концов, после многих попыток кошка, помещенная в ящик, сразу же нажимает рычаг или петлю, действуя четко определенным образом (Thorndike 1898, p. 13).

Эти и другие эксперименты (с собаками, цыплятами, обезьянами и даже рыбами) позволили Торндайку сформулировать ряд «законов» обучения, из которых наибольшее влияние оказал *закон эффекта*, процитированный в главе 1. Этот закон описывает то, что теперь известно как обучение методом проб и ошибок. Как отмечалось в главе 1, многие аспекты закона эффекта породили споры, а с годами его детали подверглись изменениям. Тем не менее этот закон – в той или иной форме – выражает сохраняющий актуальность принцип научения.

Существенные черты алгоритмов обучения с подкреплением соответствуют особенностям обучения животных, которые описываются законом эффекта. Во-первых, алгоритмы обучения с подкреплением *селекционные*, т. е. они пытаются искать альтернативы и выбирать подходящие путем сравнения последствий. Во-вторых, алгоритмы обучения с подкреплением *ассоциативные*, т. е. альтернативы, найденные в результате селекции, ассоциируются с конкретными ситуациями или состояниями для формирования стратегии агента. Как и обучение, описываемое



Один из проблемных ящиков Торндайка
Печатается по изданию: Thorndike. Animal Intelligence: An Experimental Study of the Associative Processes in Animals // The Psychological Review, Series of Monograph Supplements II (4). Macmillan, New York, 1898

законом эффекта, обучение с подкреплением – процесс, который не просто *находит* действия, порождающие наибольшее вознаграждение, а еще и *связывает* эти действия с ситуациями или состояниями. Торндайк говорил об обучении путем «отбора и связывания» (Hilgard, 1956). Естественный отбор в эволюции – главный пример селекционного процесса, но он не является ассоциативным (по крайней мере, в общепринятом понимании); обучение с учителем – ассоциативный, но не селекционный процесс, поскольку опирается на инструкции, которые прямо говорят агенту, как он должен изменить поведение.

В терминах вычислений закон эффекта описывает элементарный способ комбинирования *поиска* и *запоминания*: поиск – в форме опробования и отбора из многих действий в каждой ситуации, а запоминание – в форме ассоциаций, связывающих ситуации с найденными действиями, которые – пока – проявили себя наилучшим образом в этих ситуациях. Поиск и запоминание – существенные компоненты всех алгоритмов обучения с подкреплением вне зависимости от того, какую форму принимает запоминание: стратегия агента, функция ценности или модель окружающей среды.

Необходимость встраивать в алгоритм обучения с подкреплением поиск означает, что он должен тем или иным способом производить исследование. Очевидно, что и животные исследуют, но между учеными, занимавшимися первыми экспериментами по обучению животных, не было согласия в том, в какой мере животное использует ориентиры при выборе действий в ситуациях наподобие проблемного ящика Торндайка. Являются ли действия результатом «абсолютно случайного, слепого нащупывания» (Woodworth, 1938, p. 777) или имеются какие-то ориентиры, вытекающие из предшествующего обучения, рассуждения или еще чего-то? Некоторые ученые, включая Торндайка, похоже, склоняются к первой точке зрения, тогда как другие выступают за более осмысленное исследование. Алгоритмы обучения с подкреплением оставляют широкий простор для выбора степени подсказки, доступной агенту при выборе действий. Формы исследования, встречающиеся в алгоритмах, описанных в этой книге, например ϵ -жадный и дающий верхнюю доверительную границу выбор действий, относятся к числу простейших. Возможны и более сложные методы с единственным условием, что для эффективной работы алгоритма должна присутствовать *какая-то* форма исследования.

Особенность нашей трактовки обучения с подкреплением, состоящая в том, что мы допускаем зависимость множества действий, доступных в любой момент времени, от текущего состояния окружающей среды, отражает наблюдения Торндайка за поведением кошек в проблемном ящике. Кошки выбирали действия из числа тех, которые инстинктивно выполняли в текущей ситуации. Торндайк называл их «инстинктивными импульсами». Будучи впервые помещена в проблемный ящик, кошка энергично царапается, рвет когтями и кусается: это инстинктивная реакция кошки на пребывание в ограниченном пространстве. Успешные действия выбираются из числа этих, а не из множества всех возможных действий. Это похоже на свойство нашего формализма, согласно которому действие, предпринятое в состоянии s , выбирается из множества допустимых действий $\mathcal{A}(s)$. Задание таких множеств – важный аспект обучения с подкреплением, потому что это может сильно упростить обучение. Они подобны инстинктивным действиям животного. С другой стороны, кошки Торндайка, возможно, исследовали окружение в соответствии с инстинктивным, зависящим от контекста *упорядочением* действий,

а не просто выбирали действия из множества инстинктивных импульсов. Это еще один способ облегчить обучение с подкреплением.

Среди наиболее выдающихся ученых, на которых оказал влияние закон эффекта, были Кларк Халл (см., например, Hull, 1943) и Б. Ф. Скиннер (см. например, Skinner, 1938). Центральное место в их работах занимала идея выбора поведения на основе его последствий. У обучения с подкреплением есть черты, роднящие его с теорией Халла, включавшей механизмы типа приемлемости и вторичного подкрепления для объяснения способности к обучению в ситуации, когда между действием и последующим подкрепляющим стимулом проходит значительное время (см. раздел 14.4). Случайность также играла роль в теории Халла – для этого в ней предусмотрен механизм «осцилляции поведения», позволяющий ввести в поведение элементы исследования.

Скиннер не в полной мере поддерживал идею запоминания в законе эффекта. Будучи противником идеи ассоциативных связей, он вместо этого придавал большое значение выбору из вариантов спонтанного поведения. Он ввел термин «оперантный», чтобы подчеркнуть ключевую роль эффектов действия на среду, окружающую животное. В отличие от экспериментов Торндайка и других, которые состояли из последовательностей отдельных испытаний, в экспериментах Скиннера по оперантному обусловливанию животному разрешалось длительное время демонстрировать естественное поведение без прерывания со стороны экспериментатора. Скиннер придумал камеру для оперантного обусловливания, которая теперь называется «ящиком Скиннера». В простейшем варианте она содержит рычаг или кнопку, которую животное может нажать для получения вознаграждения, например пищи или воды, доставляемых в согласии с точно определенным правилом, которое называется графиком вознаграждения. Регистрируя общее число нажатий рычага в виде функции от времени, Скиннер и его последователи могли исследовать эффект различных графиков вознаграждения на частоту нажатия рычага животным. Моделирование результатов подобных экспериментов с помощью принципов обучения с подкреплением, изложенных в этой книге, – плохо изученная тема, но некоторые работы мы все же упоминаем в разделе «Библиографические и исторические замечания».

Еще один вклад Скиннера связан с тем, что он первым осознал, что эффективность обучения животного можно повысить, если подкреплять последовательные приближения к желаемому поведению. Этот процесс он называл *формированием* поведения. Хотя данную технику применяли и другие ученые, включая самого Скиннера, в полной мере он оценил ее важность, когда вместе с сотрудниками пытался обучить голубя играть в боулинг, толкая деревянный шар клювом. Долгое время они не видели никакого действия толкания, которое можно было бы подкрепить, а затем

...решили подкреплять любой отклик, имевший малейшее сходство с толканием, – поначалу хоть простой взгляд на шар, – а потом отбирать отклики, больше напоминающие конечную форму. Результат нас поразил. Через несколько минут шар уже отскакивал от стенок ящика, как будто голубь был чемпионом по сквошу (Skinner, 1958, p. 94).

Голубь не только обучился поведению, не свойственному голубям, но и быстро прошел все стадии интерактивного процесса, в котором его поведение и подкреп-

ление изменялись и влияли друг на друга. Скиннер сравнил процесс изменения подкрепления с работой скульптора, который создает желаемую форму из глины. Формирование является действенной техникой также в компьютерных системах обучения с подкреплением. Если агенту трудно получить хоть какое-то ненулевое вознаграждение, потому ли что ситуаций, приносящих вознаграждение, вообще мало, или потому что они недоступны при заданном начальном поведении, то нужно начать с более легкой задачи и постепенно увеличивать ее трудность по мере того, как агент обучается эффективной, а иногда и непременной стратегии.

Особенно релевантной в контексте инструментального обусловливания является заимствованная из психологии концепция *мотивации*, которая относится к процессам, влияющим на направление и силу, или энергичность, поведения. Например, кошки Торндайка были мотивированы искать выход из проблемного ящика, потому что хотели получить пищу, находящуюся снаружи. Достижение этой цели было для них вознаграждением и подкрепляло действия, ведущие к обретению свободы. Трудно точно связать многогранную концепцию мотивации с вычислительным подходом, принятым в обучении с подкреплением, но очевидные связи с некоторыми гранями все же существуют.

В некотором смысле сигнал вознаграждения, получаемый агентом в обучении с подкреплением, лежит в основе его мотивации: агент мотивирован максимизировать полное вознаграждение, полученное в течение длительного времени. Тогда ключевой гранью мотивации является то, что именно приносит вознаграждение за опыт агента. В обучении с подкреплением сигналы вознаграждения зависят от состояния окружающей агента среды и действий агента. Далее, как отмечено в главе 1 (стр. 42), состояние окружающей среды включает не только информацию, внешнюю относительно машины, вмещающей агента, например организма или робота, но и о том, что находится внутри этой машины. Некоторые компоненты внутреннего состояния соответствуют тому, что психологи называют *мотивационным состоянием* животного, влияющим на то, что является для животного вознаграждением. Например, вознаграждение в виде пищи может оказаться более ценным для голодного животного, чем для только что поевшего. Концепция зависимости от состояния достаточно широка и допускает много типов модулирующих воздействий на генерирование сигналов вознаграждения.

Функции ценности – еще одна связь с психологической концепцией мотивации. Если главным мотивом выбора действий является стремление получить как можно большее вознаграждение, то для агента обучения с подкреплением, который выбирает действия с помощью функции ценности, самый непосредственный мотив – *увеличить градиент функции ценности*, т. е. выбирать действия, которые с большой вероятностью ведут в следующие состояния с наибольшей ценностью (или, что по существу то же самое, действия с наибольшей ценностью действий). Для таких агентов функции ценности – главная движущая сила, определяющая направление поведения.

Еще одна грань мотивации состоит в том, что мотивационное состояние животного влияет не только на обучение, но также на силу, или энергичность, поведения животного после обучения. Например, обучившись находить пищу в лабиринте, голодная крыса будет бежать к цели быстрее, чем сытая. Этот аспект мотивации не так очевидно связан с представленным здесь подходом к обучению с подкреплением, но в разделе «Библиографические и исторические замечания» мы при-

ведем ссылки на несколько публикаций, в которых предложены теории энергичности поведения, основанные на обучении с подкреплением.

Теперь обратимся к вопросу об обучении, при котором подкрепляющие стимулы появляются спустя длительное время после подкрепляемого события. Алгоритмические механизмы, которые делают возможным обучение с отложенным подкреплением – следы приемлемости и TD-обучение, – имеют близкие параллели с гипотезами психологов о том, как обучаются животные при таких условиях.

14.4. ОТЛОЖЕННОЕ ПОДКРЕПЛЕНИЕ

Закон эффекта требует обратного влияния на связи, и некоторые ранние критики этого закона не могли понять, как настоящее может влиять на то, что уже произошло в прошлом. Это сомнение еще усиливал тот факт, что обучение может происходить даже при наличии значительной задержки между действием и соответствующим вознаграждением или наказанием. Аналогично в классическом обусловливании обучение может иметь место, когда между началом БС и началом УС проходит заметное время. Эта «проблема отложенного вознаграждения» связана с тем, что Мински (Minsky, 1961) называл «проблемой распределения поощрения в самообучающихся системах»: как распределить поощрение за успех между многими решениями, которые могли ему способствовать? В алгоритмах обучения с подкреплением, представленных в этой книге, имеется два основных механизма решения этой проблемы. Первый – следы приемлемости, второй – использование TD-методов для обучения функций ценности, которые дают почти мгновенные оценки действий (в задачах типа экспериментов по инструментальному обусловливанию) или непосредственные цели предсказания (в задачах типа экспериментов по классическому обусловливанию). Обоим методам соответствующие механизмы в теориях обучения животных.

В работе Pavlov (1927) подчеркнуто, что каждый стимул должен оставлять в нервной системе след, существующий в течение некоторого времени после завершения стимула, и высказано предположение, что следы стимула делают возможным обучение, когда имеется разрыв во времени между началом УС и началом БС. По сей день обусловливание при таких условиях называется следовым обусловливанием (стр. 396). В предположении, что след УС еще остается в момент начала БС, обучение происходит вследствие одновременного присутствия следа и БС. В главе 15 мы обсудим некоторые гипотезы о следовых механизмах в нервной системе.

Стимульные следы предлагались также как средство заполнить временной интервал между действиями и последующими вознаграждениями или наказаниями в инструментальном обусловливании. Например, в авторитетной теории обучения Халла «молярные стимульные следы» объясняли то, что он называл *градиентом цели* животного, – описание того, почему максимальная сила инструментально обусловленного отклика уменьшается с ростом задержки подкрепления (Hull, 1932, 1943). Халл высказал гипотезу, что действия животного оставляют внутренние стимулы, следы которых экспоненциально убывают по мере увеличения времени, прошедшего с момента действия. Анализируя доступные в то время данные об обучении животных, он предположил, что следы обращаются в ноль по истечении 30–40 секунд.

Следы приемлемости, используемые в описанных в этой книге алгоритмах, напоминают следы Халла: это затухающие следы прошлых посещений состояния или прошлых пар состояние–действие. Следы приемлемости были введены Клопфом (Klopf, 1972) в его нейронной теории, где они являлись продолженными во времени следами прошлой активности синапсов, соединяющих нейроны. Следы Клопфа сложнее, чем экспоненциально затухающие следы в наших алгоритмах, и мы вернемся к их обсуждению, когда займемся его теорией в разделе 15.9.

Для объяснения градиентов цели, простирающихся на более долгие периоды, чем стимульные следы, Халл (Hull 1943) предположил, что более длительные градиенты являются результатом того, что обусловленное подкрепление распространяется назад от цели – этот процесс действует совместно с его молярными стимульными следами. Эксперименты с животными показывали, что если условия благоприятствуют развитию обусловленного подкрепления в период задержки, то эффект обучения не спадает с увеличением задержки так же сильно, как в условиях, препятствующих вторичному подкреплению. Благоприятные для обусловленного подкрепления условия складываются, если какие-то стимулы регулярно повторяются на протяжении времени задержки. Тогда ситуация выглядит так, будто вознаграждение не задержано вовсе, потому что имеется более близкое обусловленное подкрепление. Поэтому Халл представлял себе, что существует первичный градиент, основанный на задержке первичного подкрепления, опосредованный стимульными следами, и что он последовательно модифицируется и продлевается обусловленным подкреплением.

Представленные в книге алгоритмы, в которых одновременно используются следы приемлемости и функции ценности с целью реализовать обучение с отложенным подкреплением, соответствуют гипотезе Халла о том, как животные ухитряются обучаться при таких условиях. Архитектура исполнитель–критик, рассмотренная в разделах 13.5, 15.7 и 15.8, ярко иллюстрирует эту параллель. Критик применяет TD-алгоритм, чтобы обучить функцию ценности, ассоциированную с текущим поведением системы, т. е. чтобы предсказать доход текущей стратегии. Исполнитель обновляет текущую стратегию, основываясь на предсказаниях критика, а точнее на изменениях его предсказаний. TD-ошибка, порождаемая критиком, выступает в роли сигнала обусловленного подкрепления для исполнителя и дает мгновенную оценку качества, даже когда сам первичный сигнал вознаграждения поступает со значительной задержкой. Алгоритмы, оценивающие функции ценности действий, например Q-обучение и Sarsa, точно так же применяют принципы TD-обучения, чтобы сделать возможным обучение с отложенным подкреплением посредством обусловленного подкрепления. Близкая параллель между TD-обучением и деятельностью продуцирующих дофамин нейронов, которая обсуждается в главе 15, дает дополнительный аргумент в поддержку связей между алгоритмами обучения с подкреплением и этим аспектом теории обучения Халла.

14.5. КОГНИТИВНЫЕ КАРТЫ

Основанные на модели алгоритмы обучения с подкреплением пользуются моделями окружающей среды, в которых имеются элементы, похожие на то, что в психологии называется *когнитивными картами*. Напомним (см. обсуждение плани-

рования и обучения в главе 8), что под моделью окружающей среды мы понимаем все, что агент может использовать для предсказания реакции среды на свои действия в терминах переходов состояний и вознаграждений, а под планированием имеется в виду любой процесс, который вычисляет стратегию по такой модели. Модель окружающей среды состоит из двух частей: переходы состояний кодируют знания о влиянии действий на изменение состояний, а модель вознаграждения – знания о сигналах вознаграждения, ожидаемых для каждого состояния или каждой пары состояние–действие. Основанный на модели алгоритм выбирает действия, используя модель для предсказания последствий возможного образа действий в терминах будущих состояний и сигналов вознаграждения, ожидаемых в этих состояниях. Простейший вид планирования – сравнить предсказанные последствия групп различных «воображаемых» последовательностей решений.

Вопросы о том, используют ли животные модели окружающей среды, и если да, то как выглядят эти модели и как животное обучается им, сыграли важную роль в истории исследований по обучению животных. Некоторые ученые демонстрировали *латентное обучение* и тем ставили под сомнение преобладавший тогда взгляд на обучение и поведение как на связь «стимул–реакция» (С–Р), соответствующий простейшей безмодельной стратегии обучения. В самом раннем эксперименте по латентному обучению в лабиринт запускали две группы крыс. Экспериментальная группа не получала вознаграждения на первой стадии эксперимента, но в начале второй стадии в кормушке внезапно появлялась пища. Для контрольной группы пища присутствовала в кормушке на обеих стадиях. Вопрос заключался в том, обучатся ли чему-нибудь крысы из экспериментальной группы на первой стадии в отсутствие вознаграждения. Не похоже, чтобы экспериментальные крысы чему-то обучились на первой стадии, но, обнаружив пищу на второй стадии, они быстро догнали крыс из контрольной группы. Отсюда был сделан вывод, что «в течение периода без вознаграждения крысы (из экспериментальной группы) прошли латентное обучение лабиринту и смогли воспользоваться его плодами, как только появилось вознаграждение» (Blodgett, 1929).

Латентное обучение чаще всего ассоциируют с психологом Эдвардом Толменом (Edward Tolman), который интерпретировал этот и подобные результаты как знак того, что животные могли обучиться «когнитивной карте окружающей среды» в отсутствие вознаграждений или наказаний и что они могли воспользоваться этой картой впоследствии, когда получили мотивацию к достижению цели (Tolman, 1948). Когнитивная карта могла также помочь крысе спланировать маршрут к цели, отличающийся от того, которым она прошла при первоначальном исследовании. Объяснения такого рода результатов привели к длительной дискуссии, лежащей в основе двух течений в психологии: бихевиоризма и когнитивизма. В современной терминологии когнитивные карты не ограничиваются моделями пространственного размещения, а являются более общими моделями окружающей среды, или моделями «пространства задач» животного (см., например, Wilson, Takahashi, Schoenbaum, and Niv, 2014). Объяснение экспериментов по латентному обучению когнитивными картами аналогично утверждению о том, что животные применяют алгоритмы на основе модели и что модели окружающей среды можно обучиться даже без явного вознаграждения или наказания. Затем, когда у животного появляется мотивация в виде вознаграждения или наказания, эти модели используются для планирования.

Объясняя, как животные обучаются когнитивным картам, Толмен говорил, что они обучаются ассоциациям типа стимул–стимул, или С–С, ощущая на опыте последовательности стимулов в процессе исследования среды. В психологии это называется *теорией ожидания*: при заданных ассоциациях С–С появление стимула порождает ожидание следующего стимула. Это очень похоже на то, что специалисты по теории управления называют *идентификацией системы*, когда модель системы с неизвестной динамикой обучается на основе помеченных обучающих примеров. В простейших вариантах с дискретным временем обучающие примеры имеют вид пар $S-S'$, где S – состояние, а S' – следующее состояние, одновременно являющееся меткой. Если наблюдается S , то модель создает «ожидание» того, что следующим наблюдаемым состоянием будет S' . Более полезные для планирования модели включают также действия, т. е. примеры имеют вид $SA-S'$, где S' является ожидаемым, когда действие A предпринято в состоянии S . Полезно также обучиться тому, как окружающая среда генерирует вознаграждения. В этом случае примеры имеют вид $S-R$ или $SA-R$, где R – сигнал вознаграждения, ассоциированный с S или парой SA . Все это формы обучения с учителем, в результате которого агент может построить некое подобие когнитивных карт вне зависимости от того, получает ли он ненулевые сигналы вознаграждения в процессе исследования окружающей среды.

14.6. ПРИВЫЧНОЕ И ЦЕЛЕУСТРЕМЛЕННОЕ ПОВЕДЕНИЕ

Различие между алгоритмами обучения с подкреплением, основанными на модели и работающими без таковой, соответствует различию, которое психологи проводят между *привычным* и *целеустремленным* управлениями, обученными паттернами поведения. Привычки – это паттерны поведения, запускаемые в ответ на подходящие стимулы и продолжающиеся более-менее автоматически. Целеустремленное поведение в терминологии психологов преднамеренно в том смысле, что управляется знаниями о ценности целей и связи между действиями и их последствиями. Иногда говорят, что привычками управляют предшествующие стимулы, а целеустремленным поведением – его последствия (Dickinson, 1980, 1985). Целеустремленное управление обладает тем преимуществом, что может быстро изменить поведение животного, когда изменяется реакция окружающей среды на его действия. Хотя привычное поведение быстро реагирует на информацию, поступающую от окружающей среды, оно не способно быстро адаптироваться к изменениям среды. Выработка целеустремленного управления поведением, вероятно, стала главным достижением в эволюции интеллекта животных.

На рис. 14.5 иллюстрируется различие между основанными на модели и безмодельными стратегиями принятия решений в гипотетической задаче, где крыса должна пройти лабиринт с двумя разными кормушками, в которых находится вознаграждение указанной величины (рис. 14.5 сверху). Начав в состоянии S_1 , крыса должна сначала выбрать действие L (налево) или R (направо), а затем снова выбрать L или R в состоянии S_2 или S_3 , чтобы добраться до одной из кормушек. Кормушки – это заключительные состояния каждого эпизода в эпизодической задаче крысы. Безмодельная стратегия (рис. 14.5 слева внизу) опирается на сохраненные ценности пары состояние–действие. Эти ценности действий являют-

ся оценками наибольшего дохода, который крыса может ожидать для каждого действия, предпринятого в каждом (незаключительном) состоянии. Для их получения необходимо много испытаний – проходов лабиринта от начала до конца. Когда ценности действий станут достаточно хорошими оценками оптимального дохода, крыса просто должна будет выбрать в каждом состоянии действие с наибольшей ценностью – и это решение будет оптимальным. В данном случае после достижения достаточной точности оценок ценности действий крыса выбирает L в состоянии S_1 и R в состоянии S_2 , чтобы получить максимальный доход 4. Другая безмодельная стратегия могла бы просто полагаться на кешированную стратегию, а не на ценности действий, напрямую связывая S_1 с L и S_2 с R. Ни в одной из этих стратегий решения не нуждаются в модели окружающей среды. Нет необходимости обращаться к модели переходов состояний, и не нужна никакая связь между признаками кормушек и доставляемыми ими вознаграждениями.

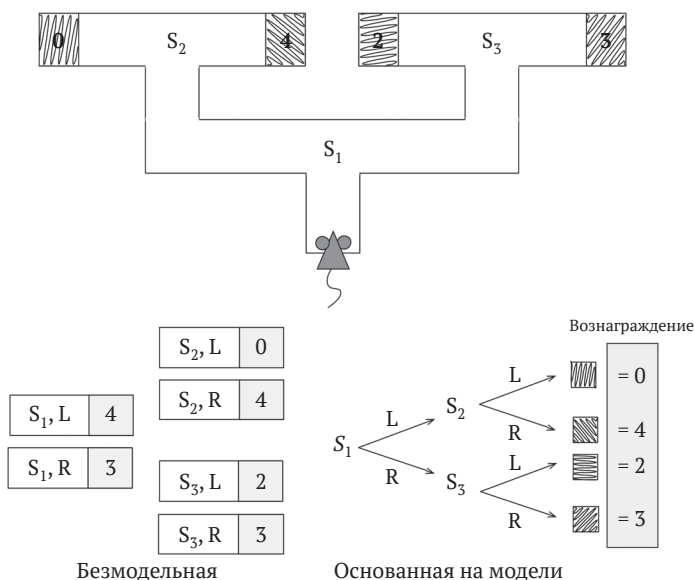


Рис. 14.5 ❖ Основанная на модели и безмодельная стратегии решения гипотетической задачи о последовательном выборе действий. **Сверху:** крыса проходит лабиринт с двумя кормушками, с которыми ассоциировано вознаграждение указанной величины. **Слева внизу:** безмодельная стратегия опирается на сохраненные ценности действий для всех пар состояние–действие, полученные по результатам многих испытаний. Для принятия решения крыса просто выбирает в каждом состоянии действие с наибольшей ценностью. **Справа внизу:** в основанной на модели стратегии крыса обучается модели окружающей среды, состоящей из информации о переходах состояние–действие–следующее состояние, и модели вознаграждения, состоящей из информации о вознаграждении, ассоциированном с каждой кормушкой. Крыса принимает решения, куда поворачивать в каждом состоянии, пользуясь моделью для имитации последовательного выбора действий с целью найти путь, приносящий максимальный доход. Взято из статьи Y. Niv, D. Joel, and P. Dayan «A Normative Perspective on Motivation» p. 376, 2006 в сборнике «Trends in Cognitive Science» volume 10, number 8. Печатается с разрешения Elsevier

На рис. 14.5 (справа внизу) иллюстрируется подход, основанный на модели. Используется модель окружающей среды, включающая модель переходов состояний и модель вознаграждения. Модель переходов состояний изображена в виде решающего дерева, а модель вознаграждения ассоциирует отличительные признаки кормушек с содержащимся в них вознаграждением. (Вознаграждения, ассоциированные с состояниями S_1 , S_2 и S_3 , также являются частью модели вознаграждения, но здесь они равны 0 и не показаны.) Основанный на модели агент решает, в какую сторону повернуть в каждом состоянии, используя модель для имитации последовательностей выбора действий с целью найти путь, приносящий максимальный доход. В данном случае доходом является вознаграждение, полученное при выходе из лабиринта в конце пути. При наличии достаточно точной модели крыса сначала выбрала бы действие L, а затем R и получила бы вознаграждение 4. Сравнение предсказанного дохода на имитированных путях – простая форма планирования, различные виды которого обсуждались в главе 8.

Когда среда, окружающая безмодельного агента, изменяет способ реагирования на действия агента, агенту необходимо приобрести новый опыт в изменившейся среде, в процессе чего он может обновить свою стратегию и (или) функцию ценности. Например, при безмодельном подходе, показанном на рис. 14.5 (слева внизу), если в одной из кормушек появится другое вознаграждение, то крысе придется пройти лабиринт, возможно много раз, чтобы, добравшись до кормушки, опытным путем получить знания о новом вознаграждении, и каждый раз обновлять свою стратегию или функцию ценности действий (или то и другое сразу), опираясь на этот опыт. Ключевой момент заключается в том, что, для того чтобы безмодельный агент изменил действие, которое его стратегия предписывает в некотором состоянии, или изменил ценность действия, ассоциированного с состоянием, он должен перейти в это состояние, совершать действия из него, быть может много раз, и на опыте испытывать последствия своих действий.

Основанный на модели агент может адаптироваться к изменениям в окружающей среде, не прибегая к такого рода «личному опыту» исследования состояний и действий, на которые повлияло изменение. Изменение в его модели автоматически (посредством планирования) изменяет его стратегию. В ходе планирования можно определить такие последствия изменений в окружающей среде, которые никогда не связались бы вместе в собственном опыте агента. Например, снова обращаясь к задаче о лабиринте на рис. 14.5, представим себе, что крысу, которая ранее обучилась модели переходов и вознаграждений, поместили прямо в кормушку справа от S_2 , и она обнаружила, что вознаграждение в ней теперь равно 1, а не 4. Модель вознаграждения крысы изменится, хотя на выбор действий, требуемых для достижения этой кормушки, это никак не повлияло. Процесс планирования включит знания о новом вознаграждении после прохода лабиринта, не нуждаясь в дополнительном опыте: в данном случае изменение стратегии состоит в том, чтобы поворачивать направо в обоих состояниях S_1 и S_3 ради получения дохода 3.

Именно такая логика лежит в основе *экспериментов по снижению ценности желаемого исхода*. Результаты этих экспериментов проливают свет на то, обучилось ли животное новой привычке или его поведением управляет целеустремленность. Эксперименты по снижению ценности желаемого исхода похожи на эксперименты по латентному обучению тем, что вознаграждение изменяется на

разных стадиях. После начальной стадии обучения с вознаграждением ценность вознаграждения на выходе изменяется, возможно, принимая нулевое или даже отрицательное значение.

Один из первых экспериментов такого типа был поставлен в работе Adams and Dickinson (1981). Они обучали крыс методами инструментального обусловливания, пока крысы не научились энергично нажимать рычаг, подающий сахарные гранулы в камеру. Затем крыс помещали в такую же камеру, но без рычага и с независимой подачей пищи, т. е. гранулы доставлялись независимо от действий крыс. После 15 минут такого свободного доступа к гранулам крысам из одной группы впрыскивался хлорид лития, вызывающий тошноту. Это повторялось на протяжении трех сеансов, в последнем из которых ни одна из крыс, получивших инъекцию, не притронулась к независимо доставляемым гранулам. Это значит, что ценность гранул как вознаграждения уменьшилась – гранулы обесценились. На следующей стадии, которая происходила днем позже, крыс снова поместили в камеру и подвергли сеансу тормозящего обучения, т. е. рычаг был возвращен на место, но отсоединен от раздатчика гранул, так что его нажатие не приводило к доставке гранул. Вопрос заключался в том, станут ли крысы, для которых ценность гранул снизилась, нажимать рычаг реже, чем остальные, даже не испытав на опыте отсутствие вознаграждения в результате нажатия. Оказалось, что у крыс, получивших инъекцию, частота реакции была существенно ниже, чем у неинъецированных крыс, *с самого начала испытаний на торможение*.

Адамс и Дикинсон пришли к выводу, что инъецированные крысы ассоциировали нажатие рычага с последующей тошнотой с помощью когнитивной карты, связывающей нажатие рычага с гранулами, а гранулы – с тошнотой. Поэтому в испытаниях на торможение крысы «знали», что за нажатием рычага последует нечто неприятное, и уменьшали количество нажатий с самого начала. Важно, что количество нажатий уменьшалось, хотя крысы не испытывали тошноту непосредственно после нажатия: когда им было плохо, никакого рычага не было и в помине. Похоже, они объединили знание об исходе выбора поведения (нажатие рычага сопровождается получением капсулы) с ценностью исхода (капсул следует избегать) и в результате смогли перестроить свое поведение. Не все психологи согласны с этим «когнитивным» объяснением результатов такого эксперимента, существуют и другие объяснения, но основанное на модели объяснение планирования широко признано.

Ничто не мешает агенту одновременно использовать безмодельные и основанные на модели алгоритмы, для этого даже есть веские причины. Из собственного опыта мы знаем, что после достаточного количества повторений целеустремленное поведение превращается в привычное. Эксперименты показывают, что у крыс дело обстоит так же. В работе Adams (1982) описан эксперимент, цель которого – проверить, может ли продленное обучение превратить целеустремленное поведение в привычное. Для этого автор сравнил, как снижение ценности желаемого исхода влияет на крыс, получивших разные объемы обучения. Если бы продленное обучение сделало крыс менее чувствительными к снижению ценности по сравнению с крысами, обучавшимися не так долго, то это свидетельствовало бы о том, что продленное обучение способствует превращению поведения в привычку. Эксперимент Адамса был поставлен примерно так же, как описанный выше эксперимент Адамса и Дикинсон. В несколько упрощенном изложении он состо-

ял в том, что крысы в одной группе обучались, пока не произвели 100 нажатий на рычаг, а крысы в другой группе – пока количество нажатий не достигло 500. После обучения ценность гранул была понижена (с помощью инъекции хлорида лития) для крыс из обеих групп. Затем обе группы были подвергнуты сеансу тормозящего обучения. Адамс задался вопросом, повлияет ли снижение ценности на частоту нажатий рычага переобученными крысами сильнее, чем получившими обычный объем обучения. Положительный ответ свидетельствовал бы, что продленное обучение снижает чувствительность к снижению ценности желаемого исхода. Оказалось, что снижение ценности заметно уменьшило частоту нажатий непереобученными крысами. Напротив, для переобученных крыс снижение ценности мало повлияло на частоту нажатий; на самом деле продленное обучение даже привело к более энергичному поведению. (В полном эксперименте участвовали еще контрольные группы, доказавшие, что сам по себе объем обучения не существенно влиял на частоту нажатий после обучения.) Этот результат позволяет предположить, что непереобученные крысы действовали целеустремленно, с учетом знаний об исходе действий, тогда как у переобученных развилась привычка к нажатию рычага.

Взгляд на этот и другие результаты с вычислительной точки зрения проливает свет на то, почему от животных можно ожидать привычного поведения в одних обстоятельствах, целеустремленного – в других и почему они переходят от одного режима к другому по мере продолжения обучения. Хотя несомненно, что алгоритмы, используемые животными, не точно совпадают с описанными в книге, поведение животных можно в какой-то мере понять, рассмотрев компромиссы, вытекающие из различных алгоритмов обучения с подкреплением. В работе специалистов по вычислительным нейронаукам Daw, Niv, and Dayan (2005) высказана идея, что животные применяют как безмодельные, так и основанные на модели процессы. Каждый процесс предлагает некоторое действие, и для выполнения выбирается действие, предложенное тем процессом, который заслуживает больше доверия в соответствии с мерами доверия, выработанными в ходе обучения. На ранних этапах обучения более надежным считается процесс планирования, являющийся частью основанной на модели системы, поскольку он связывает воедино краткосрочные предсказания, которые могут оказаться точными при наличии меньшего опыта, чем в случае долгосрочных предсказаний безмодельного процесса. Но по мере продолжения обучения более надежным становится безмодельный процесс, потому что планирование может приводить к ошибкам из-за неточности модели и срезания углов, без которого планирование было бы неосуществимо, в частности из-за различных форм «обрезки ветвей»: удаления кажущихся бесперспективными ветвей дерева поиска. В соответствии с этой идеей следовало бы ожидать перехода от целеустремленного поведения к привычному по мере накопления опыта. Предлагались и другие объяснения того, как животные выбирают между целеустремленным и привычным управлением, исследования этого и смежных вопросов в теории поведения и нейронауках продолжаются.

Различия между безмодельными и основанными на модели алгоритмами доказали свою полезность в этих исследованиях. Можно изучить вычислительные свойства этих типов алгоритмов в абстрактной постановке, которая обнажает основные преимущества и ограничения каждого типа. Это позволило бы поставить

новые и заострить существующие вопросы, а значит, спроектировать эксперименты, благодаря которым психологи смогут лучше понять привычное и целенаправленное управление поведением.

14.7. РЕЗЮМЕ

В этой главе мы ставили целью обсудить параллели между обучением с подкреплением и экспериментальным изучением обучения животных в психологии. С самого начала мы подчеркивали, что обучение с подкреплением в том виде, в каком оно описано в этой книге, не предназначено для моделирования деталей поведения животных. Это абстрактный вычислительный каркас для исследования идеализированных ситуаций с точки зрения искусственного интеллекта и конструирования технических систем. Но многие из базовых алгоритмов обучения с подкреплением родились из психологических теорий, а в некоторых случаях эти алгоритмы внесли вклад в разработку новых моделей обучения животных. В этой главе описаны параллели, которые особенно бросаются в глаза.

Различия между алгоритмами предсказания и управления в обучении с подкреплением можно сравнить с различием между классическим (павловским) и инструментальным обусловливанием в теории обучения животных. Ключевое отличие экспериментов по инструментальному и классическому обусловливанию заключается в том, что в первом случае подкрепляющий стимул контингентен поведению животного, а во втором – нет. Обучение с целью предсказания с применением TD-алгоритма соответствует классическому обусловливанию, и мы описали *TD-модель классического обусловливания* как один из примеров, когда принципы обучения с подкреплением объясняют некоторые детали поведения животного во время обучения. Эта модель обобщает авторитетную модель Рескорлы–Вагнера, включая в нее временной аспект, благодаря чему события в отдельных испытаниях оказывают влияние на обучение, и объясняет обусловливание второго порядка, при котором предшественники подкрепляющих стимулов сами становятся подкрепителями. Она также лежит в основе авторитетного взгляда на активность продуцирующих дофамин нейронов мозга; этим вопросом мы еще займемся в главе 15.

Обучение методом проб и ошибок лежит в основе той части обучения с подкреплением, которая связана с управлением. Мы рассказали о некоторых деталях экспериментов Торндайка с кошками и другими животными, которые привели его к формулировке *закона эффекта*, обсуждавшегося в главе 1. Мы отметили, что в обучении с подкреплением исследование не обязательно должно быть ограничено «слепым нащупыванием»; испытания можно генерировать хитроумными способами с использованием врожденных и приобретенных в процессе обучения знаний при условии, что какое-то исследование присутствует. Мы обсудили метод обучения Б. Ф. Скиннера, называемый *формированием*, в котором контингентность вознаграждения постепенно изменяется, чтобы обучить животное последовательному приближению к желаемому поведению. Формирование – неоценимый инструмент не только для обучения животных; его можно эффективно применять и для подкрепления обучаемых агентов. Также существует связь с идеей мотивационного состояния животного, которое влияет на то, к чему животное

будет стремиться, а чего избегать, и на то, какие события оно будет расценивать как вознаграждение, а какие – как наказание.

Представленные в этой книге алгоритмы обучения с подкреплением включают два базовых механизма решения проблемы отложенного подкрепления: следы приемлемости и функции ценности, обучаемые TD-алгоритмами. У обоих механизмов есть прототипы в теориях обучения животных. Следы приемлемости похожи на стимульные следы в ранних теориях, а функции ценности соответствуют роли вторичного подкрепления, поскольку дают почти мгновенную обратную связь в виде оценок.

Следующая параллель, рассмотренная в этой главе, – между моделями окружающей среды в обучении с подкреплением и тем, что в психологии называется *когнитивными картами*. Эксперименты, поставленные в середине XX века, имели целью продемонстрировать, что животные способны обучаться когнитивным картам в качестве альтернативы или дополнения к ассоциациям между состояниями и действиями, и впоследствии использовать их для выбора поведения, особенно когда окружающая среда неожиданно изменяется. Модели окружающей среды в обучении с подкреплением похожи на когнитивные карты тем, что могут быть обучены методами обучения с учителем, не полагаясь на сигналы вознаграждения, а затем использованы для планирования поведением.

Различие между *безмодельными* и *основанными на модели* алгоритмами обучения с подкреплением соответствует различию между *привычным* и *целеустремленным* поведением в психологии. Безмодельные алгоритмы принимают решения с помощью доступа к информации, сохраненной в стратегии или функции ценности действий, тогда как основанные на модели методы выбирают действия в результате заблаговременного планирования с применением модели окружающей среды. Эксперименты по снижению ценности желаемого исхода дают информацию о том, является ли поведение животного привычным или целеустремленным. Теория обучения с подкреплением помогла прояснить представления об этих вопросах.

Обучение животных, конечно, служит источником идей для обучения с подкреплением, но, будучи видом машинного обучения, обучение с подкреплением направлено на проектирование и осмысление эффективных алгоритмов обучения, а не на воспроизведение или объяснение нюансов поведения животных. Мы акцентировали внимание на тех аспектах обучения животных, которые очевидным образом связаны с методами решения задач предсказания и управления, и подчеркивали плодотворный двусторонний обмен идеями между обучением с подкреплением и психологией, однако не погружались слишком глубоко во многие детали и споры, которые занимают внимание исследователей поведения животных. Вполне вероятно, что в ходе будущего развития теории и алгоритмов обучения с подкреплением найдут применение связи со многими другими особенностями обучения животных, по мере более глубокого понимания их вычислительной полезности. Мы ожидаем, что обмен идеями между обучением с подкреплением и психологией продолжится и принесет плоды обоим дисциплинам.

Многие связи между обучением с подкреплением и различными областями психологии и других наук о поведении остались за рамками этой главы. Мы почти полностью опустили обсуждение связей с психологией принятия решений – дисциплиной, в которой изучается, как выбираются действия и принимаются

решения *после* завершения обучения. Мы также не обсуждали связи с экологическими и эволюционными аспектами поведения, изучаемые в этологии и поведенческой экологии: как животные относятся друг к другу и к своему физическому окружению и каков вклад их поведения в эволюционную приспособляемость. Оптимизация, МППР и динамическое программирование играют большую роль в этих областях, а наш акцент на взаимодействие агента с динамической окружающей средой связан с изучением поведения агента в комплексных «экологических условиях». Многоагентное обучение с подкреплением, которое вообще не рассматривается в этой книге, имеет связи с социальными аспектами поведения. И хотя мы об этом не говорили, ни в коем случае не следует считать, что обучение с подкреплением вовсе не учитывает эволюционные аспекты. Ничто в обучении с подкреплением не должно наводить на мысль об интерпретации обучения и поведения как *tabula rasa*¹. На самом деле опыт создания технических приложений подчеркнул важность встраивания в системы обучения с подкреплением знаний, аналогичных роли эволюции в изучении животных.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

Работы Ludvig, Bellemare, and Pearson (2011) и Shah (2012) представляют собой обзоры обучения с подкреплением в контексте психологии и нейронаук. Эти публикации могут служить полезным дополнением к этой и следующей главе.

14.1 В работе Dayan, Niv, Seymour, and Daw (2006) рассматриваются взаимодействия между классическим и инструментальным обусловливанием, особенно ситуации, когда классически и инструментально обусловленные реакции вступают в конфликт. Авторы предложили подход к моделированию аспектов этого взаимодействия на основе Q-обучения. В работе Modayil and Sutton (2014) описывается использование подвижного робота для демонстрации эффективности метода управления, сочетающего фиксированные отклики с онлайн-обучением предсказанию. Называя это *павловским управлением*, авторы подчеркивают, что оно отличается от обычных для обучения с подкреплением методов управления, поскольку основано на предсказанном выполнении фиксированных действий, а не на максимизации вознаграждения. Электромеханическая машина Росса (Ross, 1933) и в особенности самообучающаяся версия черепахи Уолтера (Walter, 1951) – очень ранние иллюстрации павловского управления.

14.2.1 В работе Kamin (1968) впервые сообщается о блокировке в классическом обусловливании, которая теперь известна под названием «блокировка Кэмина». В работе Moore and Schmajuk (2008) приведено великолепное краткое описание феномена блокировки, порожденных им исследований и его продолжающегося влияния на теорию обучения животных. В работе Gibbs, Cool, Land, Kehoe, and Gormezano (1991) описано обусловливание второго порядка реакции мигательной перепонки у кролика и его связь с обусловливанием серийными составными стимулами. В работе Finch

¹ Чистый лист (лат.). – Прим. перев.

and Culler (1934) сообщается о получении обусловливания пятого порядка отдергивания передней лапы собакой, «когда мотивация животного под-держивается различными командами».

- 14.2.2 Встроенная в модель Рескорлы–Вагнера идея о том, что обучение имеет место, когда животное удивлено, является выводом из работы Kamin (1969). Помимо модели Рескорлы–Вагнера, к моделям классического обусловливания относятся модели, описанные в работах Klopff (1988), Grossberg (1975), Mackintosh (1975), Moore and Stickney (1980), Pearce and Hall (1980) и Courville, Daw, and Touretzky (2006). В работе Schmajuk (2008) приведен обзор моделей классического обусловливания. Вагнер (Wagner, 2008) предлагает взгляд на модель Рескорлы–Вагнера и подобные ей элементарные теории обучения с точки зрения современной психологии.
- 14.2.3 Ранний вариант TD-модели классического обусловливания встречается в работе Sutton and Barto (1981a), куда также включено предсказание этой ранней модели о том, что первичность во времени подавляет блокировку; позднее в работе Kehoe, Schreurs, and Graham (1987) было показано, что именно так обстоит дело с подготовкой мигательной перепонки у кролика. В статье Sutton and Barto (1981a) содержится одна из ранних констатаций почти полной тождественности модели Рескорлы–Вагнера и правила обучения на основе наименьших средних квадратов (LMS), или правила Уидроу–Хоффа (Widrow and Hoff, 1960). Эта ранняя модель была пересмотрена после разработки Саттоном TD-алгоритма (Sutton, 1984, 1988) и впервые представлена в виде TD-модели в работе Sutton and Barto (1987), а затем более полно в работе Sutton and Barto (1990), которая в значительной степени и легла в основу этого раздела. Дополнительное исследование TD-модели и ее возможной нейронной реализации было выполнено Муром с сотрудниками (Moore, Desmond, Berthier, Blazis, Sutton, and Barto, 1986; Moore and Blazis, 1989; Moore, Choi, and Brunzell, 1998; Moore, Marks, Castagna, and Polewan, 2001). Теория классического обусловливания на основе побудительного подкрепления (drive-reinforcement) Клопфа (Klopff, 1988) развивает TD-модель, включая в нее такие экспериментальные детали, как S-образная форма кривых приобретения. В некоторых из вышеупомянутых публикаций TD расшифровывается как Time Derivative (производная по времени), а не Temporal Difference (временное различие).
- 14.2.4 В работе Ludvig, Sutton, and Kehoe (2012) оценивается качество TD-модели в ранее неисследованных задачах, включающих классическое обусловливание, и изучается влияние различных представлений стимулов, в т. ч. предложенного ими ранее (Ludvig, Sutton, and Kehoe, 2008) микростимульного представления. Более ранние исследования влияния представлений стимулов и их возможных нейронных реализаций на временные характеристики и топографию реакции в контексте TD-модели были проведены Муром с сотрудниками (см. выше). Хотя и не в контексте TD-модели, различные представления, похожие на микростимульное (Ludvig et al. 2012), были предложены и изучены в работах Grossberg and Schmajuk (1989), Brown, Bullock, and Grossberg (1999), Buhusi and Schmajuk (1999) и Machado (1997). Рисунки на стр. 406–409 заимствованы из работы Sutton and Barto (1990).

- 14.3** В разделе 1.7 имеются замечания об истории обучения методом проб и ошибок и о законе эффекта. Идея о том, что кошки Торндайка, возможно, занимались исследованием в соответствии с инстинктивным контекстно-зависимым упорядочением действий, а не просто выбирали один из множества инстинктивных импульсов, была высказана Питером Дайаном (частное сообщение). В работе Selfridge, Sutton, and Barto (1985) иллюстрируется эффективность формирования в задаче о балансировании стержня. Другие примеры формирования в обучении с подкреплением имеются в работах Gullapalli and Barto (1992), Mahadevan and Connell (1992), Mataric (1994), Dorigo and Colombette (1994), Saksida, Raymond, and Touretzky (1997) и Randløv and Alstrøm (1998). В работах Ng (2003) и Ng, Harada, and Russell (1999) термин «формирование» употребляется в несколько ином смысле, чем у Скиннера; акцент делается на том, как изменить сигнал вознаграждения, не изменяя множество оптимальных стратегий.
- В работе Dickinson and Balleine (2002) обсуждается сложность взаимодействия между обучением и мотивацией. В работе Wise (2004) приведен обзор обучения с подкреплением и его связи с мотивацией. В работе Daw and Shohamy (2008) мотивация и обучение связываются с различными аспектами теории обучения с подкреплением. См. также McClure, Daw, and Montague (2003), Niv, Joel, and Dayan (2006), Rangel, Camerer, and Montague (2008) и Dayan and Berridge (2014). В работах McClure et al. (2003), Niv, Daw, and Dayan (2006) и Niv, Daw, Joel, and Dayan (2007) представлены теории поведенческой энергичности, тесно связанные с инфраструктурой обучения с подкреплением.
- 14.4** Спенс, ученик Халла и сотрудник Йельского университета, подробно изучал роль подкрепления высшего порядка в решении проблемы отложенного подкрепления (Spence, 1947). Обучение с очень длительными задержками, например в экспериментах по обусловливанию вкусового отвращения с задержками до нескольких часов, привело к теориям интерференции как альтернативе теориям затухания следа (Revusky and Garcia, 1970; Boakes and Costa, 2014). В других интерпретациях обучения с отложенным подкреплением акцент делается на роли осведомленности и рабочей памяти (например, Clark and Squire, 1998; Seo, Barraclough, and Lee, 2007).
- 14.5** Работа Thistlethwaite (1951) представляет собой подробный обзор экспериментов по латентному обучению вплоть до момента ее публикации. В работе Ljung (1998) приведен обзор методов обучения модели, или идентификации системы в технике. В работе Gopnik, Glymour, Sobel, Schulz, Kushnir, and Danks (2004) представлена байесовская теория того, как дети обучаются моделям.
- 14.6** Предположение о наличии связи между привычным и целеустремленным поведением и безмодельным и основанным на модели обучением с подкреплением впервые высказано в работе Daw, Niv, and Dayan (2005). Задача о гипотетическом лабиринте, использованная для объяснения контроля над привычным и целеустремленным поведением, основана на объяснении из работы Niv, Joel, and Dayan (2006). В обзоре Dolan and Dayan (2013)

рассматриваются четыре поколения экспериментальных исследований, относящихся к этому вопросу, и обсуждается, как можно продвинуться вперед на основе различия между безмодельным и основанным на модели обучении с подкреплением. В работах Dickinson (1980, 1985) и Dickinson and Balleine (2002) обсуждаются экспериментальные факты, относящиеся к этому различию. В работе Donahoe and Burgos (2000) содержатся альтернативные аргументы в пользу того, что безмодельные процессы могут объяснить результаты экспериментов по снижению ценности ожидаемого исхода. В работе Dayan and Berridge (2014) доказываемое, что классическое обусловливание имеет прямое отношение к основанным на модели процессам. В обзоре Rangel, Camerer, and Montague (2008) упоминаются многие нерешенные вопросы, касающиеся привычного, целеустремленного и павловского режимов управления.

Замечания о терминологии. Традиционное значение слова *подкрепление* в психологии – усиление паттерна поведения (путем увеличения его интенсивности или частоты) в результате того, что животное получает стимул (или испытывает отсутствие стимула), находящийся в соответствующей временной связи с другим стимулом или с реакцией. Подкрепление рождает изменения, остающиеся в будущем поведении. Иногда в психологии под подкреплением понимается процесс порождения устойчивых изменений в поведении вне зависимости от того, усиливают эти изменения или ослабляют некоторый поведенческий паттерн (Mackintosh, 1983). Тот факт, что подкрепление может означать ослабление, а не только усиление, противоречит обыденному смыслу слова и его традиционному употреблению в психологии, но это полезное обобщение, которое мы здесь принимаем. В любом случае стимул, рассматриваемый как причина изменения в поведении, называется *подкрепителем*.

Психологи обычно не используют словосочетание *обучение с подкреплением*. Пионеры обучения животных, вероятно, считали обучение и подкрепление синонимами, а употребление обоих слов – избыточностью. Наше использование этого термина следует традиции, сложившейся в компьютерных и технических исследованиях, под влиянием в первую очередь работы Minsky (1961). Однако эта фраза получает все большее распространение в современной психологии и нейронауках, по-видимому, из-за наличия очевидных параллелей между алгоритмами обучения с подкреплением и обучением животных – параллелей, описанных в этой и следующей главах.

В соответствии с общепринятым словоупотреблением *вознаграждением* называется предмет или событие, которое животное стремится приблизить и ради которого готово трудиться. Вознаграждение может быть предоставлено животному в знак признания его «хорошего» поведения или для того, чтобы сделать его поведение «лучше». Аналогично *наказание* – это предмет или событие, которого животное обычно избегает и которое является следствием «плохого» поведения, обычно с целью это поведение изменить. *Первичным вознаграждением* называется вознаграждение, встроенное в нервную систему животного в результате эволюции, чтобы повысить его шансы на выживание и воспроизводство, т. е. вознаграждение, порождаемое вкусом питательной еды, сексуальным контактом, успешным спасением и многими другими стимулами и событиями, которые предсказывали репродуктивный успех на протяжении истории вида. Как объяснялось в разде-

ле 14.2.1, *вознаграждение высшего порядка* – это вознаграждение, доставляемое стимулами, которые предвещают первичное вознаграждение прямо или косвенно, предшествуя другим стимулам, предвещающим первичное вознаграждение. Вознаграждение называется *вторичным*, если его вознаграждающее качество есть результат прямого предвосхищения первичного вознаграждения.

В этой книге мы называем R_t «сигналом вознаграждения в момент t », а иногда просто «вознаграждением в момент t », но не рассматриваем это как предмет или событие в окружающей агента среде. Поскольку R_t – число, а не предмет или событие, то оно больше похоже на сигнал вознаграждения в нейронауках, т. е. внутренний сигнал в мозге, например активность нейронов, который влияет на принятие решений и обучение. Этот сигнал может быть запущен, когда животное воспринимает приятный (или неприятный) объект, но его могут запускать и вещи, которые физически не существуют в окружающей животное среде, например воспоминания, идеи или галлюцинации. Поскольку в нашем случае R_t может быть положительным, нулевым и отрицательным, наверное, было бы лучше называть отрицательное R_t наказанием, а равное нулю – нейтральным сигналом, но для простоты мы обычно избегаем этих терминов.

В обучении с подкреплением процесс, генерирующий все R_t , определяет ту задачу, которую агент пытается решить. Цель агента – максимизировать величину R_t во времени. В этом смысле R_t напоминает первичное вознаграждение для животного, если мы рассматриваем задачу, стоящую перед животным, как задачу получения максимально большого вознаграждения за всю жизнь (и тем самым благодаря предполагаемой «мудрости» эволюции повысить шансы на решение настоящей проблемы – передать свои гены будущим поколениям). Однако в главе 15 мы предполагаем маловероятным, что в мозге животного существует один такой «главный» сигнал вознаграждения, как R_t .

Не все подкрепители являются вознаграждениями или наказаниями. Иногда подкрепление не есть результат получения животным стимула, который оценивает его поведение в терминах плохое–хорошее. Поведенческий паттерн можно подкрепить стимулом, который поступает животному вне зависимости от его поведения. В разделе 14.1 объяснено, что зависимость или независимость подкрепителя от предшествующего поведения определяет различие между экспериментами по инструментальному (оперантному) и классическому (павловскому) обусловливаниям. Подкрепление работает в обоих видах экспериментов, но только в первом оно является обратной связью, оценивающей прошлое поведение. (Впрочем, часто отмечалось, что даже в тех случаях, когда подкрепляющий БС в эксперименте по классическому обусловливанию не контингентен предшествующему поведению, это поведение может влиять на его подкрепляющую ценность – например, если глаз закрыт, то воздушный удар по глазу не столь неприятен.)

Различие между сигналами вознаграждения и сигналами подкрепления станет решающим, когда мы будем обсуждать корреляции этих сигналов с нейронами в следующей главе. Мы считаем, что сигнал подкрепления, как и сигнал вознаграждения, в любой момент времени может быть положительным, отрицательным или нулевым числом. Сигнал подкрепления – это основной фактор, определяющий направление изменений, вносимых алгоритмом обучения в стратегию агента, оценки ценностей или модели окружающей среды. Для нас наиболее полезно определение сигнала подкрепления в некоторый момент времени как числа, на

которое умножается (возможно, вместе с другими постоянными) вектор, описывающий обновление параметров в некотором алгоритме обучения.

Для некоторых алгоритмов один лишь сигнал вознаграждения уже является критическим множителем в уравнении обновления параметров. В этих алгоритмах сигнал подкрепления совпадает с сигналом вознаграждения. Но в большинстве алгоритмов, обсуждаемых в этой книге, сигналы подкрепления включают еще и дополнительные члены; примером может служить TD-ошибка $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$, которая является сигналом подкрепления для TD-обучения ценности состояний (и аналогичные TD-ошибки для обучения ценности действий). В этом сигнале подкрепления R_{t+1} – вклад *первичного подкрепления*, а временное различие предсказанных значений, $\gamma V(S_{t+1}) - V(S_t)$ (или аналогичное временное различие для ценностей действий), – вклад *обусловленного подкрепления*. Таким образом, когда $\gamma V(S_{t+1}) - V(S_t) = 0$, δ_t сигнализирует о «чистом» первичном подкреплении, а когда $R_{t+1} = 0$ – о «чистом» обусловленном подкреплении. Но часто δ_t сигнализирует о смеси того и другого. Заметим, что, как было отмечено в разделе 6.1, δ_t недоступно до момента $t + 1$. Поэтому мы рассматриваем δ_t как сигнал подкрепления в момент $t + 1$, что вполне логично, потому что этот сигнал подкрепляет предсказания и (или) действия, сделанные раньше, на шаге t .

Возможный источник путаницы – терминология, принятая знаменитым психологом Б. Ф. Скиннером и его последователями. Для Скиннера положительное подкрепление имеет место, когда последствия некоторого поведения животного увеличивают частоту такого поведения, а наказание – когда последствия поведения уменьшают его частоту. Отрицательное подкрепление имеет место, когда поведение влечет за собой устранение негативных стимулов (которые животному не нравятся), увеличивая тем самым частоту такого поведения. С другой стороны, отрицательное наказание имеет место, когда поведение влечет за собой устранение позитивных стимулов (которые животному нравятся), уменьшая тем самым частоту такого поведения. Мы не ощущаем необходимости в таких различиях, поскольку наш подход более абстрактный, а сигналам вознаграждения и подкрепления разрешено принимать как положительные, так и отрицательные значения. (Однако особо отметим, что отрицательное значение нашего сигнала подкрепления – не то же самое, что отрицательное подкрепление по Скиннеру.)

С другой стороны, часто отмечалось, что использование одного числа в качестве сигнала вознаграждения или наказания, различие между которыми зависит только от знака числа, вступает в противоречие с тем фактом, что системы позитивного и негативного восприятий¹ у животных имеют качественно различные свойства, и в них задействованы различные механизмы мозга. Это указывает, в каком направлении инфраструктура обучения с подкреплением может развиваться в будущем в попытке воспользоваться вычислительными преимуществами данных систем, но сейчас мы не будем останавливаться на этих возможностях.

Еще одно терминологическое расхождение связано с нашим использованием слова *действие*. Для многих ученых-когнитивистов действие целенаправленно в том смысле, что является результатом знания животного о связи между поведением и его последствиями. Действие ориентировано на какую-то цель и является результатом принятого решения, в отличие от реакции в ответ на стимул – она

¹ В психологии применяются термины *аппетитивная* и *аверсивная* система. – Прим. перев.

является результатом рефлекса или привычки. Мы употребляем слово «действие», не различая то, что другие авторы называют действиями, решениями и реакциями. Это важные различия, но для нас они уже являются частью различий между безмодельными и основанными на модели алгоритмами обучения с подкреплением, которые мы обсуждали в разделе 14.6 в связи с привычным и целеустремленным поведением. В работе Dickinson (1985) обсуждается различие между реакцией и действием.

В этой книге часто употребляется термин *управление*. Мы понимаем под этим нечто совершенно иное, чем специалисты по психологии обучения животных. Говоря об управлении, мы имеем в виду, что агент влияет на свою окружающую среду, чтобы вызвать появление предпочтительных для него состояний или событий: агент управляет своей средой. Такой смысл вкладывают в это слово специалисты по системам управления. С другой стороны, в психологии под управлением обычно понимают, что на поведение животного влияют – управляют им – стимулы, получаемые животным (стимульный контроль), или режим подкрепления. Здесь окружающая среда управляет агентом. Управление в этом смысле – основа терапии, направленной на модификацию поведения. Разумеется, когда агент взаимодействует с окружающей средой, управление осуществляется в обоих направлениях, но нас в качестве управляющей стороны больше интересует агент, а не окружающая среда. Нашей точке зрения эквивалентен, а быть может, даже более понятен взгляд на агента как на объект, управляющий входными сигналами, получаемыми от окружающей среды (Powers, 1973). Это не то, что психологи называют стимульным контролем.

Иногда обучение с подкреплением понимают как формирование стратегий обучения непосредственно на основе вознаграждений (и наказаний) без участия функций ценности и моделей окружающей среды. Это то, что в психологии называется обучением типа стимул–реакция, или С–Р. Но для нас, как и для большинства современных психологов, обучение с подкреплением гораздо шире и включает, помимо обучения типа С–Р, методы, в которых есть место функциям ценности, моделям окружающей среды, планированию и другим процессам, которые принято относить к более когнитивной стороне функционирования мозга.

Глава 15

Нейронауки

Нейронауки – это междисциплинарная область знаний, занимающаяся изучением нервной системы: как она регулирует функции организма, управляет поведением, изменяется со временем в результате развития, обучения и старения и как клеточные и молекулярные механизмы обеспечивают все эти функции. Один из наиболее захватывающих аспектов обучения с подкреплением – растущее число фактов, добытых нейронауками и свидетельствующих о том, что нервная система человека и многих других животных реализует алгоритмы, удивительным образом соответствующие алгоритмам обучения с подкреплением. Главная цель этой главы – объяснить эти параллели и рассказать, какие выводы они позволяют сделать о нейронных основах вознаграждения в обучении животных.

Самая поразительная точка соприкосновения нейронаук и обучения с подкреплением связана с дофамином – химическим веществом, принимающим важное участие в обработке вознаграждения в мозге млекопитающих. Похоже, что дофамин передает ошибки временных различий (TD-ошибки) в структуры мозга, ответственные за обучение и принятие решений. Эта параллель выражается *гипотезой об ошибке предсказания вознаграждения, передаваемой дофаминовыми нейронами*, гипотезой, которая стала результатом конвергенции вычислительного обучения с подкреплением и итогов нейронаучных экспериментов. В этой главе мы обсудим эту гипотезу и приведшие к ней открытия нейронаук, а также объясним, почему так велик ее вклад в понимание мозговых систем вознаграждения. Мы также обсудим другие, не столь яркие параллели между обучением с подкреплением и нейронауками, которые тем не менее дают полезные средства для концептуального осмысления обучения на основе вознаграждения у животных. У обучения с подкреплением есть и другие элементы, потенциально способные оказать влияние на изучение нервной системы, но их связи с нейронауками пока еще недостаточно развиты. Мы обсудим несколько таких просматривающихся связей, важность которых, на наш взгляд, со временем возрастет.

В разделе вводной главы, посвященном истории вопроса (раздел 1.7), мы отметили, что многие аспекты обучения с подкреплением развивались под влиянием нейронаук. Вторая цель этой главы – познакомить читателей с теми идеями о функционировании мозга, которые внесли вклад в наш подход к обучению с подкреплением. Некоторые элементы обучения с подкреплением проще понять, рассматривая их в свете теорий работы мозга. Особенно это относится к идее следа приемлемости – одного из базовых механизмов обучения с подкреплением, предложенного в качестве гипотетического свойства синапсов – структур, посредством которых нервные клетки – нейроны – взаимодействуют между собой.

В этой главе мы не станем очень глубоко вдаваться в невообразимо сложные детали нервных систем, отвечающие за основанное на вознаграждении обучение животных: глава слишком коротка, а мы не являемся нейрочеными. Мы не будем пытаться описать – или хотя бы перечислить – многочисленные мозговые структуры и нервные пути, а также молекулярные механизмы, которые, как полагают, вовлечены в эти процессы. Не пытаемся мы и обосновать гипотезы и модели, альтернативные тем, что так хорошо согласуются с обучением с подкреплением. Неудивительно, что между узкими специалистами имеются расхождения во мнениях. Мы можем только бросить мимолетный взгляд на эту чарующую и развивающуюся историю. Надеемся, однако, что эта глава убедит вас в том, что появился очень плодотворный канал, соединяющий обучение с подкреплением и его теоретические основания с нейронаучными представлениями об основанном на вознаграждении обучении животных.

Есть много прекрасных публикаций, посвященных связям между обучением с подкреплением и нейронауками, некоторые из них упомянуты в последнем разделе этой главы. Наша трактовка отличается от принятой в большинстве из них, поскольку мы предполагаем знакомство с обучением с подкреплением в том виде, в котором оно изложено в предыдущих главах, но не предполагаем знакомства с нейронауками. Начнем с краткого введения в нейронаучные концепции, необходимые для понимания изложенного ниже на элементарном уровне.

15.1. ОСНОВЫ НЕЙРОНАУК

Базовая информация о нервной системе будет полезна для понимания материала этой главы. Встречающиеся термины выделены курсивом при первом упоминании. Если вы хоть немного знакомы с нейронауками, то можете пропустить этот раздел.

Нейроны, основные элементы нервной системы, – это клетки, специализированные для обработки и передачи информации с помощью электрических и химических сигналов. Они встречаются в разных формах, но обычно у нейрона имеется клеточное тело, *дендриты* и единственный *аксон*. Дендриты – это разветвленные отростки, по которым поступает входная информация от других нейронов (или внешние сигналы в случае рецепторных нейронов). Аксон нейрона – это волокно, по которому передается выходной сигнал другим нейронам (или мускулам либо железам). Выходной сигнал нейрона состоит из последовательности электрических импульсов, называемых *потенциалами действия*, которые передаются через аксон. Потенциалы действия называют также *спайками* и говорят, что нейрон, генерирующий спайк, *пульсирует*. В моделях нейронных сетей принято использовать вещественные числа для представления *частоты пульсации* нейрона – среднего числа спайков в единицу времени. Аксон может широко разветвляться, так что потенциал нейрона достигает многих целей. Разветвленная структура аксона называется *аксональной сетью* (axonal arbor). Проведение потенциала действия – активный процесс, в чем-то похожий на перегорание предохранителя, поэтому, когда потенциал действия достигает точки разветвления аксона, он «зажигает» потенциалы действия на всех исходящих ветвях (хотя иногда распространения по ветви может не быть). В результате активность нейрона с сильным аксональным разветвлением может оказать влияние на многие конечные цели.

Синапс – это структура, которая обычно находится на конце аксональной ветви и опосредует взаимодействие одного нейрона с другим. Синапс передает информацию от аксона *предсинаптического* нейрона дендриту или клеточному телу *постсинаптического* нейрона. За некоторыми исключениями, синапс высвобождает химический *нейромедиатор*, как только от предсинаптического нейрона приходит потенциал действия. (Исключениями являются случаи прямой электрической связи между нейронами, но здесь они нас не интересуют.) Молекулы нейромедиатора, высвобожденные на предсинаптической стороне синапса, распространяются через *синаптическую щель* – очень узкое пространство между предсинаптическим окончанием и постсинаптическим нейроном, – а затем прикрепляются к рецепторам на поверхности постсинаптического нейрона, чтобы возбудить или затормозить его активность по генерации спайков или для модуляции его поведения иными способами. Конкретный нейромедиатор может прикрепляться к нескольким разным типам рецепторов, каждый из которых оказывает различное влияние на постсинаптический нейрон. Например, существует по меньшей мере пять разных типов рецепторов, с помощью которых нейромедиатор дофамин может воздействовать на постсинаптический нейрон. Определено много химических веществ, играющих роль нейромедиаторов в нервных системах животных.

Фоновая активность нейрона – это тот уровень активности, обычно частота пульсации, при котором нейрон не выглядит находящимся под влиянием синаптического входного сигнала, связанного с задачей, интересующей экспериментатора, например когда активность нейрона не коррелирует со стимулом, доставляемым подопытному в ходе эксперимента. Фоновая активность может быть нерегулярной из-за сигналов, поступающих от широкой сети, или из-за шумов внутри нейрона или его синапсов. Иногда фоновая активность является результатом динамических процессов внутри нейрона. В отличие от фоновой, *фазическая* активность нейрона состоит из всплесков спайковой активности, обычно вызванной синаптическим входным сигналом. Активность, которая изменяется медленно и зачастую ступенчато, не важно, фоновая она или нет, называется *тонической активностью* нейрона.

Сила, или эффективность, с которой высвобождение нейромедиатора в синапсе влияет на постсинаптический нейрон, называется *силой* синапса. Один из способов изменения нервной системы благодаря опыту – посредством изменения силы синапсов в результате комбинации активностей предсинаптического и постсинаптического нейронов, иногда в присутствии *нейромодулятора*, т. е. нейромедиатора, который оказывает какие-то эффекты вместо или дополнительно к прямому быстрому возбуждению либо торможению.

В мозге имеется несколько систем нейромодуляции, состоящих из скоплений нейронов с широко разветвленной аксональной сетью, и каждая система пользуется своим нейромедиатором. Нейромодуляция может изменять функции нейронных контуров, влиять на мотивацию, возбуждение, внимание, память, настроение, эмоции, сон и температуру тела. Здесь важно, что нейромодуляторная система может распространять нечто похожее на скалярный сигнал, например сигнал подкрепления, с целью изменить работу синапсов в сильно распределенных участках, критических для обучения.

Способность силы синапса к изменению называется *синаптической пластичностью*. Это один из главных механизмов, отвечающих за обучение. Параметры,

или веса, корректируемые алгоритмами обучения, соответствуют силам синапсов. Ниже мы покажем, что модуляция синаптической пластичности посредством нейромодулятора дофамина – вероятный механизм, с помощью которого мозг мог бы реализовать алгоритмы обучения, похожие на описанные в этой книге.

15.2. СИГНАЛЫ ВОЗНАГРАЖДЕНИЯ, СИГНАЛЫ ПОДКРЕПЛЕНИЯ, ЦЕННОСТИ И ОШИБКИ ПРЕДСКАЗАНИЯ

Связи между нейронами и вычислительным обучением с подкреплением начинались как параллели между сигналами в мозге и сигналами, играющими важную роль в теории и алгоритмах обучения с подкреплением. В главе 3 мы говорили, что любую проблему обучения целеустремленному поведению можно свести к трем сигналам, представляющим действия, состояния и вознаграждения. Однако чтобы объяснить связи, установленные между нейронами и обучением с подкреплением, придется понизить уровень абстрактности и рассмотреть другие сигналы обучения с подкреплением, которые так или иначе соответствуют сигналам в мозге. Помимо сигналов вознаграждения, к таковым относятся сигналы подкрепления (которые, на наш взгляд, отличны от сигналов вознаграждения), сигналы ценности и сигналы, передающие ошибки предсказания. Маркируя сигнал его функцией, мы делаем это в контексте теории обучения с подкреплением, в которой сигнал соответствует некоторому члену в уравнении или алгоритме. С другой стороны, говоря о сигнале в мозге, мы имеем в виду физиологическое событие, например всплеск потенциалов действий или секрецию нейромедиатора. Маркируя нервный сигнал его функцией, например называя фазическую активность дофаминавого нейрона сигналом подкрепления, мы имеем в виду, что нервный сигнал ведет себя – и предположительно функционирует – как соответствующий теоретический сигнал.

Для выявления свидетельств в пользу таких параллелей необходимо преодолеть много трудностей. Нервную активность, связанную с обработкой вознаграждения, можно обнаружить почти в любой части мозга, и интерпретировать результаты однозначно очень сложно, потому что представления различных относящихся к вознаграждению сигналов зачастую сильно коррелируют друг с другом. Необходимо тщательно планировать эксперимент, чтобы более-менее уверенно отделить какой-то один тип относящегося к вознаграждению сигнала от других – или от множества других сигналов, никак не связанных с обработкой вознаграждения. Но, несмотря на эти трудности, было поставлено много экспериментов с целью согласовать различные аспекты теории и алгоритмов обучения с подкреплением с нервными сигналами. При этом удалось установить ряд очень интересных связей. Чтобы подготовить читателя к изучению этих связей, в оставшейся части данного раздела мы напомним, что означают различные относящиеся к вознаграждению сигналы в теории обучения с подкреплением.

В замечаниях о терминологии в конце предыдущей главы мы говорили, что R_t можно сравнить с сигналом вознаграждения в мозге животного, а не с предметом или событием в окружающей среде. В обучении с подкреплением сигнал вознаграждения (наряду с окружающей агента средой) определяет задачу,

которую обучающийся агент пытается решить. В этом отношении R_t похоже на сигнал в мозге животного, который распространяет первичное вознаграждение в различные участки мозга. Но маловероятно, что в мозге животного существует единый главный сигнал вознаграждения типа R_t . Лучше считать R_t абстракцией, суммирующей общий эффект нескольких нервных сигналов, генерируемых многочисленными системами мозга, которые оценивают качества ощущений и состояний с точки зрения вознаграждения или наказания.

Сигналы подкрепления в обучении с подкреплением – не то же самое, что сигналы вознаграждения. Функция сигнала подкрепления – определить направление изменений, которые алгоритм обучения вносит в стратегию агента, оценки ценностей и модели окружающей среды. Например, в случае TD-метода сигналом подкрепления в момент t является TD-ошибка $\delta_{t-1} = R_t + \gamma V(S_t) - V(S_{t-1})$ ¹. В некоторых алгоритмах сигнал подкрепления может совпадать с сигналом вознаграждения, но в большинстве он равен сигналу вознаграждения, скорректированному с учетом какой-то другой информации, например оценок ценностей в TD-ошибках.

Оценки ценности состояний или действий, т. е. V или Q , определяют, что для агента хорошо, а что плохо в долгосрочной перспективе. Это предсказания полного вознаграждения, которое агент может ожидать в будущем. Агенты принимают хорошие решения, выбирая действия, которые ведут в состояния с наивысшей оценкой ценности состояний, или действия, имеющие наивысшую оценку ценности действий.

Ошибки предсказания измеряют расхождение между ожидаемым и фактическим сигналом или ощущением. Конкретно ошибка предсказания вознаграждения (ОПВ) измеряет расхождение между ожидаемым и полученным сигналом вознаграждения; она положительна, если сигнал вознаграждения больше ожидаемого, и отрицательна в противном случае. TD-ошибки вида (6.5) – это частные случаи ОПВ, измеряющие расхождение между текущим и предыдущим ожиданиями долгосрочного вознаграждения. Говоря об ОПВ, нейроученые обычно (но не всегда) имеют в виду такие ОПВ в виде временных различий, которые мы в этой главе называем просто TD-ошибками. Кроме того, в этой главе TD-ошибка, как правило, не зависит от действий, в отличие от TD-ошибок, используемых при обучении ценностям действий в таких алгоритмах, как Sarsa и Q-обучение. Это связано с тем, что наиболее известные связи с нейронауками формулируются в терминах TD-ошибок, не содержащих действий, но мы не собираемся исключать из рассмотрения возможные похожие связи, включающие зависимость от действий TD-ошибки. (TD-ошибки полезны также для предсказания сигналов, отличных от вознаграждения, но мы этот случай здесь не рассматриваем. См., например, работу Modayil, White, and Sutton, 2014.)

Можно задать много вопросов о связях между данными, полученными в нейронауках, и этими теоретически определенными сигналами. На что больше похож наблюдаемый сигнал: на сигнал вознаграждения, на сигнал ценности, на ошибку

¹ Как отмечалось в разделе 6.1, δ_t в нашей нотации определено как $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$, поэтому δ_t недоступно до момента $t + 1$. В момент t TD-ошибка в действительности равна $\delta_{t-1} = R_t + \gamma V(S_t) - V(S_{t-1})$. Поскольку мы считаем временные шаги очень малыми или даже бесконечно малыми промежутками, не стоит придавать чрезмерную важность несущественному сдвигу на один шаг.

предсказания, на сигнал подкрепления или на что-то совсем другое? И если это сигнал ошибки, то какой именно: ОПВ, TD-ошибки или чего-то более простого, например ошибки Рескорлы–Вагнера (14.3)? И если это TD-ошибка, то зависит ли она от действий, как TD-ошибка в алгоритмах Q-обучения или Sarsa? Как отмечено выше, ставить эксперименты над мозгом для получения ответов на подобные вопросы очень трудно. Но экспериментальные факты позволяют предположить, что один нейромедиатор, а конкретно дофамин, сигнализирует об ОПВ и, более того, что фазическая активность продуцирующих дофамин нейронов на самом деле передает TD-ошибки (определение фазической активности см. в разделе 15.1). Эти факты привели к *гипотезе об ошибке предсказания вознаграждения, передаваемой дофаминовыми нейронами*, к которой мы и переходим.

15.3. ГИПОТЕЗА ОБ ОШИБКЕ ПРЕДСКАЗАНИЯ ВОЗНАГРАЖДЕНИЯ

Гипотеза об ошибке предсказания вознаграждения, передаваемой дофаминовыми нейронами, содержит предположение о том, что одна из функций фазической активности продуцирующих дофамин нейронов у млекопитающих состоит в доставке ошибки между старой и новой оценками ожидаемого будущего вознаграждения в целевые участки мозга. Эта гипотеза (хотя и в других терминах) впервые была явно сформулирована в работе Montague, Dayan, and Sejnowski (1996), которые показали, как концепция TD-ошибки из обучения с подкреплением объясняет многие особенности фазической активности дофаминовых нейронов у млекопитающих. Эксперименты, приведшие к этой гипотезе, были выполнены в 1980-х – начале 1990-х годов в лаборатории нейробиолога Вольфрама Шульца (Wolfram Schultz). В разделе 15.5 описаны эти оказавшие заметное влияние эксперименты. В разделе 15.6 объяснено, как их результаты согласуются с TD-ошибками, а в разделе «Библиографические и исторические замечания» в конце главы имеются ссылки на литературу, относящуюся к разработке данной гипотезы.

В работе Montague et al. (1996) сравниваются TD-ошибки в TD-модели классического обусловливания с фазической активностью дофаминовых нейронов в экспериментах по классическому обусловливанню. Напомним (см. раздел 14.2), что TD-модель классического обусловливания, по существу, представляет собой алгоритм полугradientного спуска TD(λ) с линейной аппроксимацией функций. Монтэгю с соавторами сделали несколько предположений, чтобы провести это сравнение. Во-первых, поскольку TD-ошибка может быть отрицательной, но нейрон не может иметь отрицательную частоту пульсации, они предположили, что величина, соответствующая активности дофаминового нейрона, равна $\delta_{t-1} + b_t$, где b_t – фоновая частота пульсации нейрона. Отрицательная TD-ошибка ведет к падению частоты пульсации дофаминового нейрона ниже фоновой величины¹.

Понадобилось также второе предположение о состояниях, посещаемых в каждом испытании по классическому обусловливанню, и об их представлении в виде входных данных алгоритма обучения. Это тот же вопрос, который мы обсуждали

¹ В литературе, когда речь заходит о соотношении TD-ошибки с активностью дофаминовых нейронов, под δ_t понимается то, что мы обозначаем $\delta_{t-1} = R_t + \gamma V(S_t) - V(S_{t-1})$.

в разделе 14.2.4 для TD-модели. Монтэгу с соавторами выбрали полное серийное составное (ПСС) представление, показанное в левом столбце на рис. 14.1, но такое, что последовательность кратковременных внутренних сигналов продолжается до начала БС, каковым в данном случае является поступление ненулевого сигнала вознаграждения. Такое представление позволяет отразить в TD-ошибке тот факт, что активность дофаминавого нейрона не только предсказывает будущее вознаграждение, но и чувствительна к тому, когда это вознаграждение ожидается после предвещающего ориентира. Должен существовать какой-то способ запоминать время между сенсорными ориентирами и поступлением вознаграждения. Если стимул инициирует последовательность внутренних сигналов, которая продолжается после окончания стимула, и если существует другой сигнал для каждого временного шага, следующего за стимулом, то каждый временной шаг после стимула представляется отдельным состоянием. Таким образом, TD-ошибка, будучи независимой от состояния, может быть чувствительна к хронометражу событий в течение испытания.

В имитированных испытаниях при таких предположениях о частоте фоновой пульсации и представлении входных данных TD-ошибки в TD-модели удивительно похожи на фазическую активность дофаминавых нейронов. Предвосхищая описание этого сходства в разделе 15.5, скажем, что можно провести параллели между TD-ошибками и следующими особенностями активности дофаминавых нейронов: 1) фазический отклик дофаминавого нейрона имеет место, только когда вознаграждающее событие не было предсказано; 2) на ранних стадиях обучения нейтральные ориентиры, предшествующие вознаграждению, не вызывают значительного фазического отклика дофаминавого нейрона, но по мере продолжения обучения эти ориентиры набирают предсказательную ценность и приводят к выраженному фазическому отклику; 3) если ориентир, который уже набрал предсказательную ценность, предшествует еще более ранний ориентир, то фазический отклик дофаминавого нейрона перемещается к более раннему ориентиру, а для более позднего прекращается; 4) если после обучения предсказанное вознаграждающее событие не происходит, то отклик дофаминавого нейрона падает ниже базового уровня вскоре после ожидаемого момента вознаграждающего события.

Хотя в экспериментах Шульца с сотрудниками не каждый обслуживаемый нейрон демонстрировал все описанные выше особенности поведения, разительное соответствие между активностью большинства отслеживаемых нейронов и TD-ошибками стало весомым аргументом в пользу гипотезы об ошибке предсказания вознаграждения. Однако имеются ситуации, когда предсказания, основанные на этой гипотезе, не совпадают с экспериментальными наблюдениями. От выбора входного представления очень сильно зависит, насколько хорошо TD-ошибки описывают некоторые детали активности дофаминавых нейронов, особенно те, что касаются хронометража откликов нейрона. Выдвигались различные идеи (некоторые мы обсудим ниже) о том, какие входные представления и другие аспекты TD-обучения позволяют приблизить TD-ошибки к данным, но главные параллели проявляются при выборе ПСС-представления, использованного в работах Монтэгу с сотрудниками. В общем и целом гипотеза об ошибке предсказания вознаграждения завоевала широкое признание у нейрочученых, изучающих обучение, основанное на вознаграждении, и доказала удивительную жизнеспособность перед лицом новых экспериментальных фактов.

Чтобы подготовить почву для описания экспериментов, поддерживающих гипотезу об ошибке предсказания вознаграждения, и предложить контекст, в котором можно будет оценить важность этой гипотезы, мы далее изложим некоторые известные факты о дофамине, о мозговых структурах, на которые он влияет, и о его роли в обучении, основанном на вознаграждении.

15.4. ДОФАМИН

Дофамин продуцируется в качестве нейромедиатора нейронами, клеточные тела которых расположены преимущественно в двух скоплениях в среднем мозге млекопитающих: компактной части черного вещества (*substantia nigra pars compacta* – SNpc) и вентральной области покрышки (*ventral tegmental area* – VTA). Дофамин играет существенную роль во многих процессах в мозге млекопитающих. Наиболее важные из них – мотивация, обучение, выбор действий, многие формы зависимости, а также такие расстройства, как шизофрения и болезнь Паркинсона. Дофамин называют нейромодулятором, потому что он выполняет многие функции, помимо прямого быстрого возбуждения и торможения целевых нейронов. Хотя о функциях дофамина и деталях его воздействия на клетки нам еще многое неизвестно, очевидно, что он играет фундаментальную роль в обработке вознаграждения в мозге млекопитающих. Дофамин – не единственный нейромодулятор, принимающий участие в обработке вознаграждения, и его роль в аверсивных ситуациях – в случае наказания – остается противоречивой. Кроме того, у животных, отличных от млекопитающих, дофамин может действовать по-другому. Но никто не сомневается, что дофамин очень важен в процессах, связанных с вознаграждением, у всех млекопитающих, включая человека.

Ранее превалировало представление о том, что дофаминовые нейроны распространяют сигнал вознаграждения в несколько областей мозга, вовлеченных в обучение и мотивацию. Это представление последовало за знаменитой работой Джеймса Олдса (James Olds) и Питера Милнера (Peter Milner) 1954 года, в которой были описаны эффекты электрической стимуляции некоторых участков мозга крысы. Они обнаружили, что электрическая стимуляция определенных областей действует как чрезвычайно мощное вознаграждение при управлении поведением крыс: «...контроль над поведением животного с помощью этого вознаграждения очень высокий, быть может, сильнее, чем воздействие любого другого вознаграждения, которое раньше использовалось в экспериментах с животными» (Olds and Milner, 1954). Более поздние исследования установили, что участки, стимуляция которых оказывается наиболее эффективной для порождения этого эффекта вознаграждения, возбуждали – прямо или косвенно – дофаминовые пути, которые обычно возбуждаются естественными вознаграждающими стимулами. Похожие эффекты наблюдались и у подопытных людей. Эти наблюдения позволяют с большой уверенностью предположить, что активность дофаминовых нейронов сигнализирует о вознаграждении.

Но если гипотеза об ошибке предсказания вознаграждения верна – даже если она объясняет только некоторые особенности активности дофаминовых нейронов, – то этот традиционный взгляд на активность дофаминовых нейронов не совсем правилен: фазические отклики дофаминовых нейронов сигнализируют об

ошибках предсказания вознаграждения, а не о самом вознаграждении. В терминах обучения с подкреплением фазический отклик дофаминового нейрона в момент t соответствует величине $\delta_{t-1} = R_t + \gamma V(S_t) - V(S_{t-1})$, а не R_t .

Теория и алгоритмы обучения с подкреплением помогают примирить взгляд, основанный на ошибке предсказания вознаграждения, с традиционным представлением о том, что дофамин сигнализирует о вознаграждении. Во многих обсуждаемых в этой книге алгоритмах δ играет роль сигнала подкрепления, т. е. является главной движущей силой обучения. Например, δ – критический фактор в TD-модели классического обусловливания и δ – сигнал подкрепления при обучении функции ценности и стратегии в архитектуре исполнитель–критик (разделы 13.5 и 15.7). Зависящие от действий формы δ являются сигналами подкрепления в алгоритмах Q-обучения и Sarsa. Сигнал вознаграждения R_t – важнейший компонент δ_{t-1} , но не только он определяет подкрепляющий эффект в этих алгоритмах. Дополнительный член $\gamma V(S_t) - V(S_{t-1})$ – часть δ_{t-1} , описывающая подкрепление высшего порядка, и даже если вознаграждение имеет место ($R_t \neq 0$), TD-ошибка может ничего не сообщать, если вознаграждение полностью предсказано (мы объясним это в разделе 15.6 ниже).

На самом деле при более внимательном изучении статьи Олдса и Милнера 1954 года выясняется, что она главным образом о подкреплении эффекта электрической стимуляции в задаче инструментального обусловливания. Электрическая стимуляция не только возбуждала крыс – благодаря влиянию дофамина на мотивацию, – но и приводила к тому, что крысы быстро обучались стимулировать себя сами, нажимая на рычаг, и могли делать это часто в течение длительного периода времени. Активность дофаминовых нейронов, запущенная электрической стимуляцией, подкрепляла нажатие на рычаг.

В поставленных позже экспериментах с применением оптогенетических методов окончательно подтверждена роль фазических откликов дофаминовых нейронов как сигналов подкрепления. Эти методы позволяют нейроученым с миллисекундной точностью контролировать активность нейронов выбранного типа у бодрствующего животного. Оптогенетический метод заключается во введении светочувствительных белков в нейроны выбранного типа, так что эти нейроны можно активировать или заглушить с помощью лазерных вспышек. Первый эксперимент по применению оптогенетических методов для изучения дофаминовых нейронов показал, что оптогенетической стимуляции, порождающей фазическую активацию дофаминовых нейронов мыши, достаточно для выработки у мыши условного рефлекса, заставляющего ее предпочитать ту часть камеры, где она получала эту стимуляцию, а не часть, где она не получала стимуляции вовсе или получала стимуляцию более низкой частоты (Tsai et al. 2009). В другом эксперименте, описанном в работе Steinberg et al. (2013), оптогенетическая активация дофаминовых нейронов использовалась для создания искусственных всплесков их активности у крыс в те моменты, когда вознаграждающие стимулы ожидалось, но отсутствовали, – т. е. в моменты, когда активность дофаминовых нейронов обычно приостанавливается. После того как паузы были заменены искусственными всплесками, реагирование продолжалось и тогда, когда в обычных условиях должно было бы уменьшаться из-за отсутствия подкрепления (в испытаниях на затухание), а обучение было возможно тогда, когда обычно блокировалось из-за того, что вознаграждение уже предсказано (парадигма блокировки, раздел 14.2.1).

Дополнительное свидетельство в пользу подкрепляющей функции дофамина дают оптогенетические эксперименты с дрозофилами, только на этих насекомых дофамин действует противоположным образом: оптически инициируемые всплески активности дофаминовых нейронов действуют как электрошоковое раздражение лапок и подкрепляют избегание, по крайней мере для популяции с активированными дофаминовыми нейронами (Claridge-Chang et al. 2009). Хотя ни один из этих экспериментов не показал, что фазическая активность дофаминовых нейронов – точный аналог TD-ошибки, они убедительно продемонстрировали, что фазическая активность действует в точности как δ (или $-\delta$ в случае дрозофил) в роли сигнала подкрепления в алгоритмах предсказания (классическое обусловливание) и управления (инструментальное обусловливание).

Дофаминовые нейроны особенно хорошо приспособлены к распространению сигнала подкрепления на многие участки мозга. У этих нейронов огромная аксональная сеть, так что каждое высвобождение дофамина достигает в 100–1000 раз больше рецепторов, чем могут достичь аксоны типичных нейронов. На рисунке справа показана аксональная сеть одного дофаминового нейрона, клеточное тело которого находится в области SNpc мозга крысы. Каждый аксон дофаминового нейрона, находящегося в SNpc или VTA, создает приблизительно 5000 синаптических контактов с дендритами нейронов в целевых участках мозга.

Если дофаминовый нейрон распространяет сигнал подкрепления, похожий на δ в обучении с подкреплением, то, поскольку это скалярный сигнал, т. е. одно число, можно ожидать, что все дофаминовые нейроны в областях SNpc и VTA активируются более или менее одинаково, так что все будут действовать почти синхронно, посылая один и тот же сигнал всем рецепторам, которых могут достичь их аксоны. Хотя принято считать, что дофаминовые нейроны именно так

и действуют совместно, современные исследования указывают на более сложную картину, когда различные субпопуляции дофаминовых нейронов реагируют на входной сигнал по-разному в зависимости от структур, которым посылают сигналы, и, кроме того, сигналы по-разному воздействуют на целевые структуры. У дофамина имеются и другие функции, помимо сигнализации об ОПВ, но даже тем дофаминовым нейронам, которые сигнализируют об ОПВ, имеет смысл посылать разные ОПВ разным структурам в зависимости от ролей, которые эти структуры играют в порождении подкрепленного поведения. Такие детали выходят за рамки этой книги, но все же скажем, что векторные сигналы ОПВ осмыслены с точки зрения обучения с подкреплением, когда решения можно разло-



Аксональная сеть одного нейрона, продуцирующего дофамин в качестве нейромедиатора. Эти аксоны создают синаптические контакты с огромным количеством дендритов нейронов в целевых участках мозга

Заимствовано из статьи Matsuda, Furuta, Nakamura, Hioki, Fujiyama, Arai, and Kaneko в журнале «The Journal of Neuroscience», volume 29, 2009, page 451

жить на отдельные подрешения, и вообще как способ разрешить *структурный* вариант проблемы назначения уровней доверия: как распределить поощрения за успех (или порицание на неудачу) решения между многими подструктурами, которые могли принимать участие в его выработке? Мы еще вернемся к этому вопросу в разделе 15.10.

Аксоны большинства дофаминовых нейронов устанавливают синаптический контакт с нейронами лобных долей и подкорковых ядер – участков мозга, отвечающих за сознательные движения, принятие решений, обучение и такие когнитивные функции, как планирование. Поскольку в большинстве идей, связывающих дофамин с обучением с подкреплением, акцент делается на подкорковых ядрах, где связи, идущие от дофаминовых нейронов, особенно густые, мы тоже сконцентрируемся на подкорковых ядрах. Подкорковые ядра – это совокупности групп нейронов (ядра), лежащие в основании переднего мозга. Главная входная структура подкоркового ядра называется полосатым телом. Практически вся кора головного мозга, да и другие структуры, предоставляет входную информацию полосатому телу. Активность кортикальных нейронов передает очень много информации о сенсорных входах, внутренних состояниях и двигательной активности. Аксоны кортикальных нейронов устанавливают синаптический контакт с дендритами главных входных и выходных нейронов полосатого тела, которые называются средними шипиковыми нейронами. Выход полосатого тела подается обратно с помощью других подкорковых ядер и таламуса в передние зоны коры и в двигательные зоны, что дает возможность полосатому телу оказывать влияние на движение, процессы принятия абстрактных решений и обработку вознаграждений. Для обучения с подкреплением важны две основные части полосатого тела: дорсальная, которая в основном отвечает за выбор действий, и вентральная, которая, как полагают, критична для различных аспектов обработки вознаграждения, в т. ч. отвечает за назначение эмоциональной ценности чувственным ощущениям.

Дендриты средних шипиковых нейронов покрыты шипиками, с кончиками которых устанавливают синаптический контакт аксоны кортикальных нейронов. С этими шипиками – но только с их стволами – устанавливают контакт также аксоны дофаминовых нейронов (рис. 15.1). При такой конфигурации объединяются предсинаптическая активность кортикальных нейронов, постсинаптическая активность средних шипиковых нейронов и входная информация от дофаминовых нейронов. Что именно происходит в этих шипиках, пока не вполне понятно. Рисунок 15.1 позволяет составить представление о сложности взаимодействия. Здесь показаны два типа рецепторов дофамина, рецепторы глутамата – нейромедиатора кортикальных входных сигналов – и различные способы взаимодействия входных сигналов. Но накапливаются факты в пользу того, что изменения в силе синапсов на пути от коры к полосатому телу (нейроученые называют их *кортико-стриарными синапсами*) очень сильно зависят от поступающих в нужное время дофаминовых сигналов.

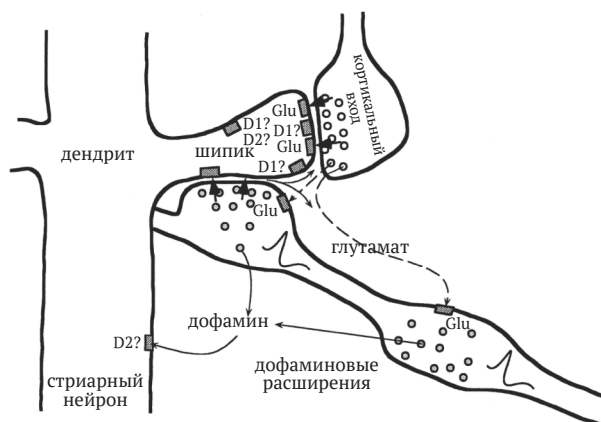


Рис. 15.1 ❖ Шипик стриарного нейрона и входы кортикальных и дофаминовых нейронов. Аксоны кортикальных нейронов оказывают воздействие на стриарные нейроны с помощью кортико-стриарных синапсов, высвобождающих нейромедиатор глутамат на кончиках шипиков, покрывающих дендриты стриарных нейронов. Видно, что аксон дофамина, находящегося в зоне VTA или SNpc, подходит к шипику (из правого нижнего угла). «Дофаминовые расширения» на этом аксоне высвобождают дофамин на поверхности ствола шипика или рядом с ней. При такой конфигурации объединяются предсинаптический вход от коры, постсинаптическая активность стриарного нейрона и дофамин, в результате чего несколько типов правил обучения получают возможность регулировать пластичность кортико-стриарных синапсов. Каждый аксон дофамина устанавливает синаптический контакт со стволами приблизительно 500 000 шипиков. На рисунке частично показаны сложности, опущенные в основном тексте: другие пути распространения нейромедиаторов и несколько типов рецепторов, в частности дофаминовые рецепторы D1 и D2, с помощью которых дофамин может продуцировать различные воздействия на шипики и другие постсинаптические участки. Заимствовано из «Journal of Neurophysiology», W. Schultz, vol. 80, 1998, page 10

15.5. ЭКСПЕРИМЕНТАЛЬНОЕ ПОДТВЕРЖДЕНИЕ ГИПОТЕЗЫ ОБ ОШИБКЕ ПРЕДСКАЗАНИЯ ВОЗНАГРАЖДЕНИЯ

Дофаминовые нейроны реагируют всплесками активности на интенсивные, новые или неожиданные зрительные и слуховые стимулы, вызывая движения глаз и тела, но лишь очень малая часть их активности связана с самими движениями. Это удивительно, потому что дегенерация дофаминовых нейронов является причиной болезни Альцгеймера, к симптомам которой относятся моторные нарушения, а в особенности дефицит движений по собственной инициативе. Заинтересовавшись слабой связью между активностью дофаминовых нейронов и движениями глаз и тела под воздействием стимулов, Ромо и Шульц (Romo and Schultz, 1990 и Schultz and Romo, 1990) предприняли шаги, подсказываемые гипотезой об ошибке предсказания вознаграждения, и решили зарегистрировать активность дофаминовых нейронов и мышечную активность обезьян, двигающих руками.

Они обучили двух обезьян тянуть руку из расслабленного положения в лоток, содержащий кусок яблока, печенье или изюм, при виде или звуке открывающейся двери. Затем обезьяна могла схватить пищу и поднести ее ко рту. После освоения этого задания обезьяну обучали двум другим. Цель первого – увидеть, что делают дофаминовые нейроны, когда движения предприняты по собственной инициативе. Лоток оставляли открытым, но накрывали сверху, чтобы обезьяна не видела, что находится внутри, но могла достать это снизу. Никаких пусковых стимулов не было, а после того как обезьяна доставала и съедала лакомство, экспериментатор обычно (но не всегда) тихо и незаметно для обезьяны заменял пищу в лотке, накатывая ее на жесткую проволоку. Здесь также активность дофаминовых нейронов, за которыми следили Ромо и Шульц, не была связана с движениями обезьяны, но значительная часть этих нейронов продуцировала фазические отклики всякий раз, как обезьяна впервые прикасалась к лакомству. Эти нейроны не реагировали, когда обезьяна касалась только проволоки или исследовала лоток, в котором не было пищи. Это подтверждало, что нейроны реагировали на пищу, а не на другие аспекты задания.

Цель второго задания состояла в том, чтобы узнать, что происходит, когда движения инициируются стимулами. Для этого использовался другой лоток с подвижной крышкой. Вид и звук открывающегося лотка инициировали движения в его сторону. Ромо и Шульц обнаружили, что после периода обучения дофаминовые нейроны переставали реагировать на прикосновение к пище, но реагировали на вид и звук открывающейся крышки лотка. Фазические отклики этих нейронов перешли от самого вознаграждения к стимулам, предвещающим доступность вознаграждения. В дополнительном исследовании Ромо и Шульц обнаружили, что большинство дофаминовых нейронов, активность которых они регистрировали, не реагируют на вид и звук открывающейся крышки вне контекста обуславливающего задания. Эти наблюдения подтверждали, что дофаминовые нейроны не реагируют ни на инициирование движения, ни на сенсорные свойства стимула, а сигнализируют об ожидании вознаграждения.

Группа Шульца провела много дополнительных исследований, относящихся к дофаминовым нейронам в зонах SNpc и VTA. Одна из серий экспериментов особенно убедительно продемонстрировала, что фазические отклики дофаминовых нейронов соответствуют именно TD-ошибкам, а не более простым ошибкам типа встречающихся в модели Рескорлы–Вагнера (14.3). В первом эксперименте (Ljungberg, Apicella, and Schultz, 1992) обезьян обучали нажимать рычаг после зажигания света, выступавшего в роли «спускового ориентира» для получения порции апельсинового сока. Как и в прежних наблюдениях Ромо и Шульца, многие дофаминовые нейроны первоначально реагировали на вознаграждение – порцию сока (рис. 15.2 сверху). Но по мере продолжения обучения многие из этих нейронов утрачивали реакцию на вознаграждение, а начинали реагировать на зажигание света, предшествующее вознаграждению (рис. 15.2 в середине). Далее в процессе обучения нажатия на рычаг стали быстрее, а количество дофаминовых нейронов, реагирующих на спусковой ориентир, уменьшилось.

После этого эксперимента тех же обезьян обучили новому заданию (Schultz, Apicella, and Ljungberg, 1993). Перед обезьянами было расположено два рычага и над каждым лампочка. Зажигание одной из лампочек было «инструктивным ориентиром», показывающим, какой из рычагов выдаст яблочного сока. В этом

задании инструктивный ориентир предшествовал спусковому ориентиру в предыдущем задании с фиксированным интервалом 1 секунда. Обезьяны обучились не тянуть руку, пока не возникнет спусковой ориентир и активность дофаминовых нейронов не увеличится, но теперь наблюдаемые дофаминовые нейроны реагировали почти исключительно на более ранний инструктивный ориентир, а не на спусковой (рис. 15.2 снизу). И в этом случае количество дофаминовых нейронов, реагирующих на инструктивный ориентир, значительно уменьшилось после завершения обучения. В ходе обучения этим двум заданиям активность дофаминовых нейронов сместилась от первоначального реагирования на вознаграждение к реагированию на более ранний из предвещающих стимулов – сначала на спусковой ориентир, а затем на еще более ранний инструктивный ориентир. По мере смещения во времени реагирование на более поздний стимул прекращалось. Это смещение реакции к более ранним предвестникам вознаграждения и прекращение реакции на более поздние – отличительный признак TD-обучения (см., например, рис. 14.2).

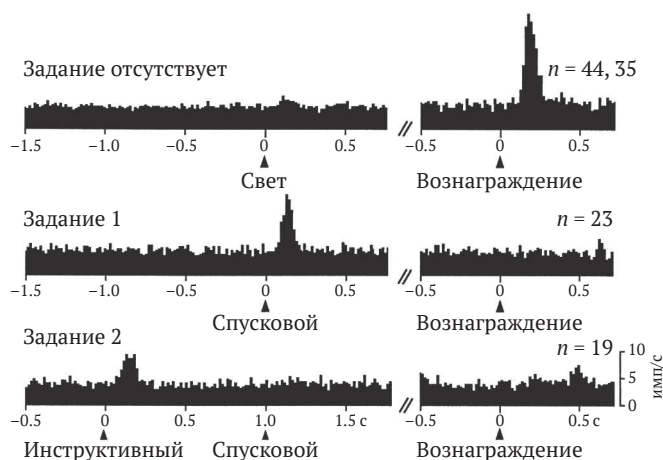


Рис. 15.2 ❖ Реакция дофаминовых нейронов смещается от начальных откликов на первичное вознаграждение к более ранним предвещающим стимулам. На графиках показано количество потенциалов действия, порожденных наблюдаемыми дофаминовыми нейронами в течение коротких промежутков времени, усредненное по всем наблюдаемым нейронам (их количество менялось от 23 до 44). **Сверху:** дофаминовые нейроны активируются непредвиденной доставкой порции яблочного сока. **В середине:** после обучения дофаминовые нейроны развили реакцию на спусковой ориентир, предвещающий вознаграждение, и перестали реагировать на саму доставку вознаграждения. **Снизу:** после добавления инструктивного ориентира, появляющегося на 1 секунду раньше спускового, реакция дофаминовых нейронов сместилась со спускового ориентира на более ранний инструктивный.

Заимствовано из Schultz et al. (1995), MIT Press

Только что описанное задание раскрыло еще одно свойство активности дофаминовых нейронов, роднящее ее с TD-обучением. Обезьяны иногда нажимали не тот рычаг, на который указывал инструктивный ориентир, и, как следствие, не получали вознаграждения. В таких испытаниях частота пульсации многих до-

фаминовых нейронов резко снижалась ниже базового уровня вскоре после срока обычной доставки вознаграждения, и это происходило без какого-либо внешнего ориентира, отмечающего обычный срок доставки (рис. 15.3). Каким-то образом обезьяны умудрялись запоминать хронометраж вознаграждения. (Хронометраж реагирования – один из тех аспектов, где простейший вариант TD-обучения необходимо модифицировать, чтобы объяснить некоторые детали времени реакции дофаминарных нейронов. Мы рассмотрим этот вопрос в следующем разделе.)

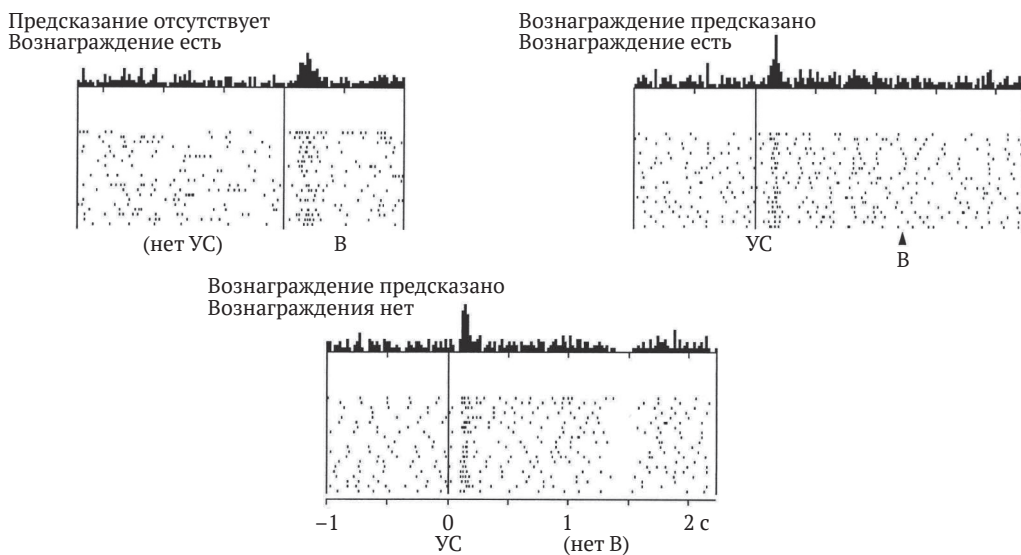


Рис. 15.3 ❖ Реакция дофаминовых нейронов падает ниже базового уровня вскоре после истечения срока ожидаемой доставки вознаграждения. **Сверху:** дофаминовые нейроны активированы непредвиденной доставкой порции яблочного сока. **В середине:** дофаминовые нейроны реагируют на условный стимул (УС), который предвещает вознаграждение, и не реагируют на само вознаграждение. **Снизу:** если вознаграждение, предсказанное УС, не поступает, то активность дофаминовых нейронов падает ниже базового уровня вскоре после ожидаемого момента вознаграждения. Над каждым рисунком показано среднее количество потенциалов действий, порожденных наблюдаемыми дофаминовыми нейронами в течение коротких промежутков времени вокруг отмеченных моментов. На растровых графиках ниже показаны паттерны активности отдельных дофаминовых нейронов: каждая точка представляет один потенциал действия. Заимствовано из работы Schultz, Dayan, and Montague «A Neural Substrate of Prediction and Reward», Science, vol. 275, issue 5306, pages 1593–1598, March 14, 1997. Печатается с разрешения AAAS

Описанные выше наблюдения позволили Шульцу и его группе сделать вывод, что дофаминовые нейроны реагируют на непредвиденные вознаграждения и на самых ранних предвестников вознаграждения и что их активность падает ниже базового уровня, если вознаграждение или его предвестник не поступает в ожидаемое время. Исследователи, знакомые с обучением с подкреплением, быстро поняли, что эти результаты поразительно похожи на то, как ведет себя TD-ошибка в качестве сигнала подкрепления в TD-алгоритме. В следующем разделе мы изучим это сходство, детально разобрав конкретный пример.

15.6. ПАРАЛЛЕЛЬ МЕЖДУ TD-ОШИБКОЙ И ДОФАМИНОМ

В этом разделе получают объяснение параллели между TD-ошибкой δ и фазическими откликами дофаминовых нейронов, наблюдавшимися в вышеописанных экспериментах. Мы рассмотрим, как δ изменяется в процессе обучения в задаче, похожей на только что описанную, – когда обезьяна сначала видит инструктивный ориентир, а спустя фиксированное время должна правильно отреагировать на спусковой ориентир, чтобы получить вознаграждение. Мы будем использовать простой идеализированный вариант этой задачи, но углубимся в детали сильнее, чем обычно, потому что хотим подчеркнуть теоретическую основу параллели между TD-ошибками и активностью дофаминовых нейронов.

Первое упрощающее предположение состоит в том, что агент уже обучился действиям, необходимым для получения вознаграждения. Тогда его задача – просто научиться точно предсказывать будущее вознаграждение по опытной последовательности состояний. Следовательно, это задача предсказания или, точнее, задача оценивания стратегии: обучить функцию ценности для фиксированной стратегии (разделы 4.1 и 6.1). Подлежащая обучению функция ценности сопоставляет каждому состоянию ценность, предсказывающую доход, который будет получен, если в этом состоянии агент выберет действие согласно заданной стратегии, где под доходом понимается сумма (возможно, обесцененная) всех будущих вознаграждений. Это нереалистичная модель для описания поведения обезьяны, поскольку обезьяна, скорее всего, обучается этим предсказаниям одновременно с обучением правильно действовать (как в алгоритме обучения с подкреплением, который обучает как стратегии, так и функции ценности, например в алгоритме типа исполнитель–критик), но этот сценарий проще описать, чем тот, в котором стратегия и функция ценности обучаются одновременно.

Теперь представим себе, что опыт агента разделен на несколько испытаний, в каждом из которых повторяется одна и та же последовательность состояний, и на каждом временном шаге испытания возникает отличное от других состояние. Представим далее, что предсказываемый доход ограничен доходом в одном испытании, что делает испытание аналогом определенного нами эпизода в обучении с подкреплением. Конечно, в действительности предсказываемые доходы не ограничены одиночными испытаниями, а временной интервал между испытаниями играет важную роль в определении того, чему именно обучается животное. Это справедливо и для TD-обучения, но здесь мы предполагаем, что доходы не накапливаются на протяжении нескольких испытаний. При таких условиях испытание в экспериментах типа тех, что были поставлены Шульцем с сотрудниками, эквивалентно эпизоду обучения с подкреплением. (Но в обсуждении ниже мы будем использовать термин «испытание», а не «эпизод», чтобы связь с экспериментами была нагляднее.)

Как обычно, нам еще нужно сделать предположение о представлении состояний, подаваемом на вход алгоритма обучения, и от этого предположения будет зависеть, насколько близко TD-ошибка соответствует активности дофаминовых нейронов. Позже мы обсудим этот вопрос, а пока остановимся на том же ПСС-представлении, которое было использовано в работе Montague et al. (1996), когда имеется отдельный внутренний стимул для каждого состояния, посещенного на каждом временном шаге испытания. Это сводит процесс к табличному случаю,

рассмотренному в первой части книги. Наконец, предположим, что агент использует алгоритм TD(0) для обучения функции ценности V , хранящейся в таблице соответствия, и эта функция для всех состояний инициализирована нулями. И еще предположим, что это детерминированная задача, а коэффициент обесценивания γ очень близок к 1, так что его можно игнорировать.

На рис. 15.4 показана динамика R , V и δ на нескольких этапах обучения в этой задаче оценивания стратегии. По оси времени откладывается временной интервал, в течение которого в испытании посещается последовательность состояний (чтобы не загромождать рисунок, мы опускаем показ отдельных состояний). Сигнал вознаграждения равен нулю в течение всего испытания, за исключением тех моментов, когда агент достигает вознаграждающего состояния. Они находятся ближе к правому концу временной оси, где сигнал вознаграждения становится положительным числом, скажем R^* . Цель TD-обучения – предсказать доход для каждого состояния, посещенного в испытании. В нашем случае, когда обесценивание отсутствует и предсказания, по предположению, ограничены отдельными испытаниями, этот доход просто равен R^* в каждом состоянии.



Рис. 15.4 ❖ Поведение TD-ошибки δ в процессе TD-обучения, согласованного с особенностями фазической активации дофаминовых нейронов. (Здесь δ – это TD-ошибка, доступная в момент t , т. е. δ_{t-1}). **Сверху:** последовательность состояний, показанная как интервал регулярных предсказателей, сопровождается ненулевым доходом R^* . **На ранних этапах обучения:** начальная функция ценности V и начальная δ , равная R^* . **По завершении обучения:** функция ценности точно предсказывает будущее вознаграждение, δ положительна в самом раннем предсказывающем состоянии и $\delta = 0$ в момент ненулевого вознаграждения. **R отсутствует:** в момент, когда предсказанное вознаграждение отсутствует, δ становится отрицательной.

Подробное объяснение происходящего см. в тексте

Вознаграждающему состоянию предшествует последовательность состояний, предсказывающих вознаграждение, *самое раннее* из них показано на левом краю временной оси. Это похоже на состояние на начальной стадии испытания, например на состояние, помеченное инструктивным ориентиром в эксперименте

с обезьянами, описанном выше и в работе Schultz et al. (1993). Это первое состояние, которое надежно предсказывает вознаграждение в этом испытании. (Конечно, в действительности состояния, посещенные в предыдущих испытаниях, являются еще более ранними предсказывающими состояниями, но, поскольку мы ограничиваемся предсказаниями в отдельных испытаниях, они не считаются предсказателями вознаграждения. Ниже приведено более точное, хотя и более абстрактное, описание самого раннего предсказывающего вознаграждение состояния.) *Самое позднее предсказывающее вознаграждение состояние* в испытании – это состояние, непосредственно предшествующее вознаграждающему состоянию. На рис. 15.4 оно показано справа на временной оси. Отметим, что вознаграждающее состояние испытания не предсказывает доход в этом испытании: ценность этого состояния должна была бы предсказывать доход по всем *последующим* испытаниям, который мы в этой эпизодической постановке считаем равным нулю.

На рис. 15.4 показана динамика V и δ в первом испытании в виде графиков с пометкой «На ранних этапах обучения». Поскольку сигнал вознаграждения равен нулю на протяжении всего испытания, кроме вознаграждающего состояния, и все V -ценности нулевые, то TD-ошибка остается равной нулю, пока не станет равной R^* в вознаграждающем состоянии. Это следует из того, что $\delta_{t-1} = R_t + V_t - V_{t-1} = R_t + 0 - 0 = R_t$, а эта величина равна нулю, пока не станет равна R^* в момент получения вознаграждения. Здесь V_t и V_{t-1} – оценки ценностей посещенных состояний в момент t и $t - 1$ соответственно. TD-ошибка на этой стадии обучения аналогична реакции дофаминового нейрона на непредвиденное вознаграждение, например на порцию яблочного сока в начале обучения.

На протяжении этого первого и всех последующих испытаний при каждом переходе состояний имеет место обновление TD(0), описанное в главе 6. Это последовательно увеличивает ценности предсказывающих вознаграждение состояний, причем увеличение распространяется назад от вознаграждающего состояния, пока ценности не сойдутся к правильным предсказаниям дохода. В данном случае (поскольку мы предполагаем, что обесценивание отсутствует) правильные предсказания равны R^* для всех предсказывающих вознаграждение состояний. Это можно видеть на графике V с пометкой «По завершении обучения» (рис. 15.4), где ценности всех предсказывающих вознаграждение состояний от самого раннего до самого позднего равны R^* . Ценности состояний, предшествующих самому раннему из предсказывающих вознаграждение, остаются малыми (на рис. 15.4 они показаны как нулевые), потому что они не являются надежными предсказателями вознаграждения.

По завершении обучения, т. е. когда V выходит на правильные значения ценностей, TD-ошибки, ассоциированные с переходами из любого предсказывающего вознаграждение состояния, равны нулю, т. к. теперь предсказания точны. Так происходит, потому что для перехода из предсказывающего вознаграждение состояния в другое такое же мы имеем $\delta_{t-1} = R_t + V_t - V_{t-1} = 0 + R^* - R^* = 0$, а для перехода из самого позднего предсказывающего вознаграждение состояния в вознаграждающее имеем: $\delta_{t-1} = R_t + V_t - V_{t-1} = R^* + 0 - R^* = 0$. С другой стороны, TD-ошибка при переходе из любого состояния в самое раннее предсказывающее вознаграждение состояние положительна из-за несовпадения низкой ценности этого состояния и более высокой ценности следующего за ним предсказывающего вознаграждение состояния. Действительно, если бы ценность состояния, пред-

шествующего самому раннему предсказывающему вознаграждение состоянию, была нулевой, то после перехода в самое раннее предсказывающее вознаграждение состояние мы имели бы $\delta_{t-1} = R_t + V_t - V_{t-1} = 0 + R^* - 0 = R^*$. На графике δ с пометкой «По завершении обучения» (рис. 15.4) эта положительная ценность показана в самом раннем предсказывающем вознаграждение состоянии, а во всех остальных она равна нулю.

Положительная TD-ошибка после перехода в самое раннее предсказывающее вознаграждение состояние аналогична длительному существованию дофаминовых откликов на самые ранние стимулы, предвещающие вознаграждение. Рассуждая таким же образом, получаем, что по завершении обучения переход из самого позднего предсказывающего вознаграждение состояния в вознаграждающее состояние порождает нулевую TD-ошибку, потому что ценность самого позднего предсказывающего вознаграждение состояния, будучи правильной, погашает вознаграждение. Это очевидная параллель с тем наблюдением, что количество дофаминовых нейронов, продуцирующих фазический отклик на полностью предсказанное вознаграждение, меньше, чем на непредвиденное.

После обучения, если вознаграждение почему-то не доставлено, TD-ошибка становится отрицательной в момент, когда обычно доставлялось вознаграждение, т. е. ценность последнего предсказывающего вознаграждение состояния слишком высока: $\delta_{t-1} = R_t + V_t - V_{t-1} = 0 + 0 - R^* = -R^*$. Это показано справа на графике с пометкой «R отсутствует». Ситуация такая же, как при падении активности дофаминовых нейронов ниже базового уровня в момент, когда ожидаемое вознаграждение не получено, – мы видели это в эксперименте Шульца, который был описан выше и показан на рис. 15.3.

Идея *самого раннего предсказывающего вознаграждение состояния* заслуживает более пристального внимания. В описанном выше сценарии, поскольку опыт разделен на испытания и мы предположили, что предсказания ограничены отдельными испытаниями, самое раннее предсказывающее вознаграждение состояние всегда является первым в испытании. Очевидно, что это искусственное ограничение. Есть более общий способ – интерпретировать самое раннее предсказывающее вознаграждение состояние как *непредсказанный предсказатель* вознаграждения, и таких состояний может быть много. В жизни животного может быть много разных состояний, предшествующих самому раннему предсказывающему вознаграждение состоянию. Однако, поскольку за этими состояниями чаще следуют другие состояния, не предсказывающие вознаграждения, то их предсказательная способность, т. е. их ценность, остается низкой. Если бы TD-алгоритм правил жизнью животных, то он обновлял бы и ценности этих состояний тоже, но обновления не накапливались бы устойчиво, потому что, по предположению, ни для одного из этих состояний нельзя с уверенностью сказать, что оно непосредственно предшествует самому раннему предсказывающему вознаграждение состоянию. Если бы для какого-нибудь из них это было так, то оно само было бы предсказывающим вознаграждение. Это может объяснить, почему при избыточном обучении уменьшаются дофаминовые отклики даже на самый ранний предсказывающий вознаграждение стимул. В случае избыточного обучения стоило бы ожидать, что даже ранее не предсказанное предсказывающее состояние должно было бы стать предсказанным в результате стимулов, ассоциированных с более ранними состояниями: взаимодействие животного с окружающей средой как

внутри, так и вне экспериментального задания стало бы привычным делом. Но если нарушить эту рутинную процедуру, введя новое задание, то TD-ошибки снова появятся, что и наблюдается на примере активности дофаминовых нейронов.

Описанный выше пример объясняет, почему у TD-ошибок так много общего с фазической активностью дофаминовых нейронов, когда животное обучается решению задачи, похожей на идеализированную задачу в нашем примере. Но не всякое свойство фазической активности дофаминовых нейронов так удачно совпадает со свойствами δ . Одно из самых удручающих расхождений проявляется, когда вознаграждение доставляется раньше, чем ожидалось. Мы видели, что отсутствие ожидаемого вознаграждения порождает отрицательную ошибку предсказания в момент ожидаемой доставки, что соответствует падению активности дофаминовых нейронов ниже базового уровня. Если вознаграждение поступает позже, чем ожидалось, то это неожиданное вознаграждение, и оно генерирует положительную ошибку предсказания. Это справедливо как для TD-ошибок, так и для реакции дофаминовых нейронов. Но если вознаграждение поступает раньше, то дофаминовые нейроны ведут себя иначе, чем TD-ошибка, – по крайней мере, в случае ПСС-представления, использованного в работе Montague et al. (1996) и в нашем примере. Дофаминовые нейроны реагируют на преждевременное вознаграждение, что согласуется с положительной TD-ошибкой, поскольку появление вознаграждения в этот момент не предсказывалось. Однако позднее, когда вознаграждение ожидается, но не поступило, TD-ошибка становится отрицательной, тогда как активность дофаминовых нейронов не падает ниже базового уровня, как должна была бы в соответствии с TD-моделью (Hollerman and Schultz, 1998). В мозге животного происходит что-то более сложное, чем просто TD-обучение с ПСС-представлением.

Некоторые противоречия между TD-ошибкой и активностью дофаминовых нейронов можно разрешить, выбирая подходящие значения параметров TD-алгоритма и применяя представления стимулов, отличные от ПСС. Например, чтобы устранить только что описанное расхождение в случае раннего вознаграждения, в работе Suri and Schultz (1999) предложено ПСС-представление, в котором последовательности внутренних сигналов, инициированные более ранними стимулами, сводятся на нет с получением вознаграждения. Другое предложение (Daw, Courville, and Touretzky, 2006) заключается в том, что TD-система в мозге пользуется представлениями, порождаемыми в результате статистического моделирования, которое выполняется в сенсорной зоне коры головного мозга, а не более простыми представлениями, основанным на исходной сенсорной информации. В работе Ludvig, Sutton, and Kehoe (2008) установлено, что TD-обучение с микростимульным (MC) представлением (рис. 14.1) соответствует активности дофаминовых нейронов при преждевременном вознаграждении и в других ситуациях лучше, чем при использовании ПСС-представления. В работе Pan, Schmidt, Wickens, and Hyland (2005) показано, что даже при ПСС-представлении пролонгированные следы приемлемости приближают TD-ошибку к некоторым аспектам активности дофаминовых нейронов. Вообще говоря, многие детали поведения TD-ошибки зависят от тонких взаимодействий между следами приемлемости, обесцениванием и представлением стимулов. Подобные открытия уточняют и развивают гипотезу об ошибке предсказания вознаграждения, не отказываясь от ее основного утверждения о том, что фазическую активность дофаминовых

нейронов можно хорошо охарактеризовать как сигнализирование о TD-ошибках.

С другой стороны, существуют противоречия между теорией TD и экспериментальными данными, которые не удастся устранить путем выбора значений параметров и представления стимулов (некоторые из них упомянуты в разделе «Библиографические и исторические замечания»). И вполне вероятно, что по мере постановки новых, более точных экспериментов количество таких противоречий будет расти. Но гипотеза об ошибке предсказания вознаграждения была – и остается – очень эффективной в качестве катализатора, улучшившего наше понимание того, как работает система вознаграждения в мозге. Были придуманы хитроумные эксперименты, имеющие целью подтвердить или опровергнуть предсказания, вытекающие из этой гипотезы. А их результаты, в свою очередь, привели к уточнению и развитию гипотезы о связи TD-ошибки и дофамина.

Удивительно во всем этом то, что теория и алгоритмы обучения с подкреплением, так хорошо согласующиеся с характеристиками дофаминовой системы, были разработаны с вычислительной точки зрения при полном отсутствии знаний о соответствующих свойствах дофаминовых нейронов – напомним, что TD-обучение и его связи с оптимальным управлением и динамическим программированием были разработаны за много лет до постановки экспериментов, которые раскрыли TD-подобную природу активности дофаминовых нейронов. Эта никем не запланированная аналогия, пусть даже не совершенная, наводит на мысль, что между TD-ошибкой и дофамином имеются параллели, которые улавливают что-то важное о процессах вознаграждения, протекающих в мозге.

Помимо объяснения многих особенностей фазической активности дофаминовых нейронов, гипотеза об ошибке предсказания вознаграждения связывает нейронауки с другими аспектами обучения с подкреплением, в частности с алгоритмами обучения, в которых TD-ошибки играют роль сигналов подкрепления. Нейронауки все еще далеки от полного понимания электрических цепей, молекулярных механизмов и функций фазической активности дофаминовых нейронов, но факты, подтверждающие гипотезу об ошибке предсказания вознаграждения наряду со свидетельствами в пользу того, что фазические отклики дофаминовых нейронов являются сигналами подкрепления при обучении, позволяют предположить, что мозг, возможно, реализует нечто вроде алгоритма исполнитель–критик, в котором TD-ошибки играют решающую роль. Правдоподобными кандидатами являются и другие алгоритмы обучения с подкреплением, но алгоритмы типа исполнитель–критик особенно хорошо согласуются с анатомией и физиологией мозга млекопитающих, как будет описано в следующих двух разделах.

15.7. Нейронный исполнитель–критик

Алгоритмы типа исполнитель–критик обучают одновременно стратегии и функции ценности. «Исполнителем» называется компонент, который обучается стратегии, а «критиком» – компонент, который узнает о том, какой стратегии сейчас следует исполнителю, чтобы «критиковать» его выбор действий. Критик пользуется TD-алгоритмом, чтобы обучить функцию ценности состояний для текущей стратегии исполнителя. Функция ценности позволяет критику критиковать выбор действий исполнителем, посылая ему TD-ошибки δ . Положительная δ озна-

чает, что действие было «хорошим», поскольку привело в состояние, ценность которого лучше ожидаемой, отрицательная – что «плохим», потому что ценность нового состояния хуже ожидаемой. Основываясь на этих критических замечаниях, исполнитель постоянно обновляет свою стратегию.

У алгоритмов исполнитель – критик есть две особенности, которые позволяют думать, что мозг, возможно, реализует подобный алгоритм. Во-первых, наличие двух компонентов – исполнитель и критик – наводит на мысль, что две части полосатого тела – дорсальная и вентральная (раздел 15.4), играющие важнейшую роль в обучении, основанном на вознаграждении, – могут функционировать примерно как исполнитель и критик. Второе свойство заключается в том, что TD-ошибка играет двоякую роль – является сигналом подкрепления как для исполнителя, так и для критика, хотя и по-разному влияет на обучение в них. Это хорошо согласуется с несколькими свойствами сети нейронов: аксоны дофаминовых нейронов подходят к нейронам дорсальной и вентральной частей полосатого тела; дофамин, похоже, критически важен для модулирования синаптической пластичности в обеих структурах; характер воздействия нейромодулятора, в т. ч. дофамина, на целевую структуру зависит от свойств этой структуры, а не только от свойств нейромодулятора.

В разделе 13.5 алгоритмы исполнитель – критик представлены в виде методов градиента стратегии, но алгоритм, описанный в работе Barto, Sutton, and Anderson (1983), был проще и представлен в виде искусственной нейронной сети (ИНС). Здесь мы описываем похожую реализацию ИНС и следуем работе Takahashi, Schoenbaum, and Niv (2008), описывая схематичное предположение о том, как эта ИНС могла бы быть реализована реальными сетями нейронов в мозге. Обсуждение правил обучения исполнителя и критика мы отложим до раздела 15.8, где представим их в виде частных случаев формулировки на основе градиента стратегии и поговорим о том, какие они позволяют высказать гипотезы о механизме модулирования синаптической пластичности дофамином.

На рис. 15.5а показана реализация алгоритма исполнитель – критик в виде ИНС, которая состоит из отдельных сетей, реализующих исполнителя и критика. Критик состоит из одного нейроноподобного блока V , выход которого представляет ценности состояний, и компонента, изображенного в виде ромба с меткой TD, который вычисляет TD-ошибки, объединяя выход V с сигналами вознаграждения и предыдущими ценностями состояний (на что указывает петля, ведущая из блока TD в него же). Сеть исполнителя состоит из одного слоя, содержащего k блоков, помеченных A_i , $i = 1, \dots, k$. Выходом каждого блока является элемент k -мерного вектора действий. Альтернативой было бы k отдельных действий, по одному для каждого блока исполнителя, которые конкурируют друг с другом за право исполнения, но здесь мы предпочитаем рассматривать весь вектор A как одно действие.

Обе сети – исполнителя и критика – получают входные данные в виде признаков, представляющих состояние окружающей агента среды. (Напомним: в главе 1 мы отмечали, что среда, окружающая агента обучения с подкреплением, включает компоненты, находящиеся как внутри, так и вне «организма», содержащего агента.) На рисунке эти признаки показаны окружностями с пометками $x_1, x_2, \dots, x_n$, причем дважды, чтобы не усложнять рисунок. С каждой связью любого признака x_i с блоком критика V и с блоками действий A_i ассоциирован вес, представляющий силу синапса. Веса в сети критика параметризуют функцию ценности, а веса

в сети исполнителя – стратегию. Сети обучаются по мере того, как эти веса изменяются в соответствии с правилами обучения критика и исполнителя, которые будут описаны в следующем разделе.

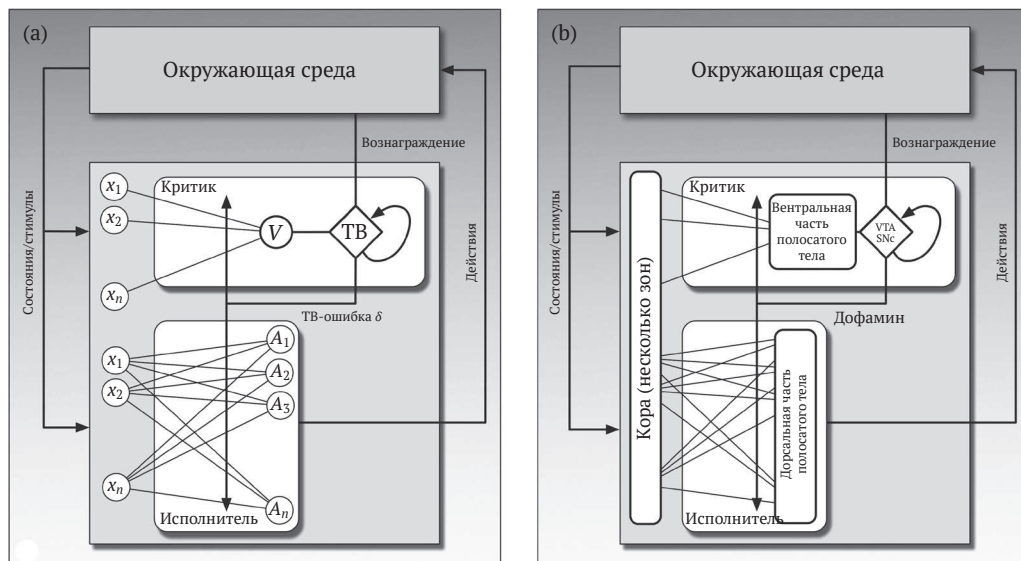


Рис. 15.5 ❖ ИНС исполнитель–критик и ее гипотетическая реализация биологическими нейронами мозга. а) Алгоритм исполнитель–критик в виде ИНС. Исполнитель корректирует стратегию на основе TD-ошибки δ , которую получает от критика. Критик корректирует параметры ценности состояний, используя ту же δ . Критик генерирует TD-ошибку, зная сигнал вознаграждения R и текущее изменение своей оценки ценностей состояний. У исполнителя нет прямого доступа к сигналу вознаграждения, а у критика – прямого доступа к действию. б) Гипотетическая реализация алгоритма исполнитель–критик биологическими нейронами. Исполнитель и та часть критика, которая обучается ценностям, находятся соответственно в дорсальной и вентральной частях полосатого тела. TD-ошибка передается дофаминовыми нейронами, расположенными в зонах VTA и SNpc, чтобы модулировать изменения синаптической силы входов от зон коры в дорсальную и вентральную части полосатого тела. Заимствовано из Y. Takahashi, G. Schoenbaum, and Y. Niv «Silencing the critics: Understanding the effects of cocaine sensitization on dorsolateral and ventral striatum in the context of an Actor/Critic model», *Frontiers in Neuroscience*, vol. 2 (1), 2008

TD-ошибка, порождаемая сетью критика, является сигналом подкрепления для изменения весов в сетях критика и исполнителя. Это показано на рис. 15.5а стрелкой с пометкой «TD-ошибка δ », которая пересекает все связи в обеих сетях. Данный аспект реализации сети в сочетании с гипотезой об ошибке предсказания вознаграждения и тем фактом, что активность дофаминовых нейронов так широко распространяется развитыми аксональными сетями этих нейронов, наводит на мысль, что нечто, подобное сети исполнитель–критик, может оказаться не такой уж притянутой за уши гипотезой о мозговых процессах, относящихся к обучению на основе вознаграждения.

На рис. 15.5b очень схематично показано, как ИНС на левом рисунке могла бы отображаться на структуры мозга согласно гипотезе, высказанной в работе Takahashi et al. (2008). Эта гипотеза помещает исполнителя и ту часть критика, которая обучается ценностям, соответственно в дорсальную и вентральную части полосатого тела – входной структуры подкорковых ядер. Напомним (см. раздел 15.4), что дорсальная часть полосатого тела в основном участвует в выборе действий, а вентральная считается критически важной для различных аспектов обработки вознаграждения, включая придание аффективного значения ощущениям. Кора головного мозга наряду с другими структурами передает полосатому телу входную информацию о стимулах, внутренних состояниях и двигательной активности.

В этой гипотетической реализации исполнителя – критика мозгом вентральная часть полосатого тела посылает ценностную информацию зонам VTA и SNpc, где дофаминовые нейроны объединяют ее с информацией о вознаграждении с целью породить активность, соответствующую TD-ошибке (хотя как именно дофаминэргические нейроны вычисляют эти ошибки, пока не ясно). Линия с пометкой «TD-ошибка δ » на рис. 15.5a превращается в линию с пометкой «Дофамин» на рис. 15.5b, которая представляет широко разветвленные аксоны дофаминовых нейронов, клеточные тела которых находятся в зонах VTA и SNpc. Возвращаясь к рис. 15.1, мы видим, что эти аксоны устанавливают синаптический контакт с шипиками на дендритах средних шипиковых нейронов – основных входных и выходных нейронов дорсальной и вентральной частей полосатого тела. Аксоны кортикальных нейронов, отправляющие входную информацию полосатому телу, устанавливают синаптический контакт с кончиками этих шипиков. Согласно гипотезе, именно в этих шипиках силы синапсов, идущих от областей коры к полосатому телу, изменяются в соответствии с правилами обучения, которые критически зависят от сигнала подкрепления, переносимого дофамином.

На рис. 15.5b показано важное следствие данной гипотезы, а именно что дофаминовый сигнал не является «главным» сигналом вознаграждения, как скалярный R_t в обучении с подкреплением. На самом деле из гипотезы вытекает, что совершенно необязательно вставлять электроды в мозг и измерять сигналы, подобные R_t , порождаемые каждым отдельным нейроном. Многие взаимосвязанные нервные системы генерируют относящуюся к вознаграждению информацию, причем какие именно структуры привлекаются к этой задаче, зависит от типа вознаграждения. Дофаминовые нейроны получают информацию из многих разных участков мозга, поэтому вход в зоны SNpc и VTA, помеченный «Вознаграждение», следует рассматривать как вектор относящейся к вознаграждению информации, поступающий к нейронам этих зон по нескольким входным каналам. Тогда теоретический скалярный сигнал вознаграждения R_t мог бы соответствовать суммарному вкладу всей относящейся к вознаграждению информации в активность дофаминовых нейронов. Это результат характерной активности, охватывающей много нейронов в разных участках мозга.

Хотя биологическая реализация исполнителя – критика, показанная на рис. 15.5b, в каких-то чертах, возможно, и правильна, очевидно, что она нуждается в уточнении, обобщении и модификации, чтобы считаться полноценной моделью функции фазической активности дофаминовых нейронов. В разделе «Библиографические и исторические замечания» в конце главы приведены ссылки на публикации, где более подробно обсуждается как эмпирическое подтверждение этой

гипотезы, так и места, где она недотягивает. А мы сейчас разберемся, какие выводы алгоритмы обучения критика и исполнителя позволяют сделать о правилах, управляющих изменениями синаптической силы кортико-стриарных синапсов.

15.8. ПРАВИЛА ОБУЧЕНИЯ КРИТИКА И ИСПОЛНИТЕЛЯ

Если мозг действительно реализует что-то, подобное алгоритму исполнитель–критик – и в предположении, что популяции нейронов транслируют общий сигнал подкрепления кортико-стриарным синапсам дорсальной и вентральной частей полосатого тела, как показано на рис. 15.5b (который, вероятно, является упрощением, о чем мы уже упоминали), – то этот сигнал подкрепления воздействует на синапсы двух указанных структур по-разному. В правилах обучения критика и исполнителя используется один и тот же сигнал, TD-ошибка δ , но ее влияние на обучение каждого из них различно. TD-ошибка (в сочетании со следами приемлемости) говорит исполнителю, как следует обновить вероятности действий, чтобы достичь состояний с более высокой ценностью. Обучение исполнителя можно сравнить с инструментальным обусловливанием с применением правила обучения типа закона эффекта (раздел 1.7): исполнитель стремится, чтобы δ было положительно и как можно больше по абсолютной величине. С другой стороны, TD-ошибка (в сочетании со следами приемлемости) сообщает критику направление и абсолютную величину изменения параметров функции ценности, которое приведет к повышению ее предсказательной точности. Критик стремится уменьшить величину δ , как можно сильнее приблизив ее к нулю, используя правило обучения типа TD-модели классического обусловливания (раздел 14.2). Различие между правилами обучения критика и исполнителя сравнительно простое, но оно оказывает глубокое влияние на обучение и принципиально для способа работы алгоритма исполнитель–критик. Различие коренится исключительно в следах приемлемости, используемых в правилах обучения обоих типов.

В нейронных сетях исполнитель–критик типа показанной на рис. 15.5b можно использовать более одного набора правил обучения, но для определенности мы ограничимся алгоритмом исполнитель–критик для непрерывных задач со следами приемлемости, описанными в разделе 13.6. При каждом переходе из состояния S_t в состояние S_{t+1} с выбором действия A_t и получением вознаграждения R_{t+1} этот алгоритм вычисляет TD-ошибку (δ), а затем обновляет векторы следов приемлемости ($\mathbf{z}_t^w$ и $\mathbf{z}_t^\theta$) и параметры критика и исполнителя ($\mathbf{w}$ и θ) по следующим правилам:

$$\begin{aligned}\delta_t &= R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w}); \\ \mathbf{z}_t^w &= \lambda^w \mathbf{z}_{t-1}^w + \nabla \hat{v}(S_t, \mathbf{w}); \\ \mathbf{z}_t^\theta &= \lambda^\theta \mathbf{z}_{t-1}^\theta + \nabla \ln \pi(A_t | S_t, \theta); \\ \mathbf{w} &\leftarrow \mathbf{w} + \alpha^w \delta_t \mathbf{z}_t^w; \\ \theta &\leftarrow \theta + \alpha^\theta \delta_t \mathbf{z}_t^\theta,\end{aligned}$$

где $\gamma \in [0, 1)$ – коэффициент обесценивания, $\lambda^w \in [0, 1]$ и $\lambda^\theta \in [0, 1]$ – параметры бутстрэппинга для критика и исполнителя соответственно, а $\alpha^w > 0$ и $\alpha^\theta > 0$ – размеры шагов.

Будем интерпретировать приближенную функцию ценности $\hat{v}$ как выход одного линейного нейроноподобного блока, называемого *блоком критика* и помеченного буквой V на рис. 15.5а. Тогда функция ценности является линейной функцией от вектора признаков, представляющего состояние s , $\mathbf{x}(s) = (x_1(s), \dots, x_n(s))^T$, параметризованной вектором весов $\mathbf{w} = (w_1, \dots, w_n)^T$:

$$\hat{v}(s, \mathbf{w}) = \mathbf{w}^T \mathbf{x}(s). \quad (15.1)$$

Каждый элемент $x_i(s)$ – аналог предсинаптического сигнала, поступающего синапсу нейрона силой w_i . Веса критика увеличиваются, согласно приведенному выше правилу, на $\alpha^w \delta_t \mathbf{z}_t^w$, где сигнал подкрепления δ_t соответствует дофаминовому сигналу, транслируемому всем синапсам блока критика. Вектор следа приемлемости $\mathbf{z}_t^w$ для блока критика является следом (усреднением недавних значений) вектора $\nabla \hat{v}(S_t, \mathbf{w})$. Поскольку $\hat{v}(s, \mathbf{w})$ линейно зависит от весов, то $\nabla \hat{v}(S_t, \mathbf{w}) = \mathbf{x}(S_t)$.

В терминах нейронов это означает, что у каждого синапса есть собственный след приемлемости, являющийся одним из элементов вектора $\mathbf{z}_t^w$. След приемлемости синапса накапливается в соответствии с уровнем активности на входе в этот синапс, т. е. уровнем предсинаптической активности, который здесь представлен элементом вектора признаков $\mathbf{x}(S_t)$. В остальном след стремится к нулю со скоростью, определяемой параметром λ^w . Синапс считается *приемлемым для модификации*, если его след приемлемости отличен от нуля. Как именно модифицируется сила синапса, зависит от сигналов подкрепления, которые поступают, когда синапс приемлем. Подобные следы приемлемости мы называем *неконтингентными следами приемлемости* синапсов блока, поскольку они зависят только от предсинаптической активности и никоим образом не контингентны постсинаптической активности.

Неконтингентные следы приемлемости синапсов блока критика означают, что правило обучения блока критика по существу является TD-моделью классического обусловливания, описанной в разделе 14.2. При том определении блока критика и его правила обучения, которое мы дали выше, критик на рис. 15.5а – то же самое, что критик в ИНС исполнитель–критик, описанной в работе Barto et al. (1983). Очевидно, что такой критик, состоящий всего из одного линейного нейроноподобного блока, – простейшая отправная точка; этот блок занимает место более сложной нейронной сети, способной обучать более сложные функции ценности.

Исполнитель на рис. 15.5а представляет собой одноуровневую сеть с k нейроноподобными блоками исполнителя, каждый из которых получает в момент t тот же вектор признаков $\mathbf{x}(S_t)$, что и блок критика. У каждого блока исполнителя j , $j = 1, \dots, k$, имеется собственный вектор весов θ_j , но, поскольку все блоки исполнителя идентичны, мы описываем только один из них и опускаем нижний индекс. Выписанным выше уравнениям алгоритма исполнитель–критик удовлетворяют, в частности, *логистические блоки Бернулли*. Это означает, что выход любого блока исполнителя в любой момент времени – случайная величина A_j , принимающая значение 0 или 1. Можно считать, что 1 соответствует пульсации нейрона, т. е. испусканию потенциала действия. Взвешенная сумма $\theta^T \mathbf{x}(S_t)$ элементов входного вектора блока определяет вероятности действий блока, имеющие экспоненциальное распределение softmax (13.2), которое в случае двух действий превращается в логистическую функцию:

$$\pi(1|s, \theta) = 1 - \pi(0|s, \theta) = \frac{1}{1 + \exp(-\theta^\top \mathbf{x}(s))}. \quad (15.2)$$

Весы блоков исполнителя увеличиваются, как показано выше, по правилу $\theta \leftarrow \theta + \alpha^\theta \delta_t \mathbf{z}_t^\theta$, где δ снова соответствует дофаминовому сигналу: одному и тому же сигналу подкрепления, который посылается всем синапсам блока критика. На рис. 15.5а показано, что δ_t транслируется всем синапсам всех блоков исполнителя (что превращает сеть исполнителя в команду агентов обучения с подкреплением, идею которой мы будем обсуждать в разделе 15.10 ниже). Вектор следа приемлемости исполнителя $\mathbf{z}_t^\theta$ – это след (усреднение недавних значений) вектора $\nabla \ln \pi(A_t | S_t, \theta)$. Чтобы понять этот след приемлемости, обратитесь к упражнению 13.5, где определяется этот вид блока и предлагается вывести для него правило обучения. В том упражнении ставилась задача выразить $\nabla \ln \pi(a | s, \theta)$ в терминах a , $\mathbf{x}(s)$ и $\pi(a | s, \theta)$ (для произвольного состояния s и действия a), вычислив градиент. Если действие и состояние относятся к моменту t , то ответ имеет вид:

$$\nabla \ln \pi(A_t | S_t, \theta) = (A_t - \pi(1 | S_t, \theta)) \mathbf{x}(S_t). \quad (15.3)$$

В отличие от неконтингентного следа приемлемости синапса критика, который накапливает только предсинаптическую активность $\mathbf{x}(S_t)$, след приемлемости синапса блока исполнителя дополнительно зависит от активности самого блока исполнителя. Мы называем его *контингентным следом приемлемости*, потому что он контингентен этой постсинаптической активности. След приемлемости в каждом синапсе постепенно затухает, но увеличивается или уменьшается в зависимости от активности предсинаптического нейрона и от того, пульсирует или нет постсинаптический нейрон. Множитель $\pi(1 | S_t, \theta)$ в формуле (15.3) положителен, когда $A_t = 1$, и отрицателен в противном случае. *Постсинаптическая контингентность следов приемлемости блоков исполнителя – единственное различие между правилами обучения критика и исполнителя.* Сохраняя информацию о том, какие действия были предприняты в каких состояниях, контингентные следы приемлемости позволяют распределять поощрение за вознаграждение (положительное δ) или порицание за наказание (отрицательное δ) между параметрами стратегии (силами синапсов блоков исполнителя) в соответствии со вкладами этих параметров в выходы блоков, которые могли бы повлиять на более поздние значения δ . Контингентные следы приемлемости маркируют синапсы в соответствии с тем, как их нужно было бы модифицировать, чтобы повлиять на будущие отклики блоков в пользу положительных значений δ .

Какие предположения правила обучения критика и исполнителя позволяют сделать о том, как изменяются силы кортико-стриарных синапсов? Оба правила обучения связаны с классическим постулатом Дональда Хебба о том, что всякий раз, как предсинаптический сигнал участвует в активации постсинаптического нейрона, сила синапса увеличивается (Hebb, 1949). Общей для правил обучения критика и исполнителя и постулата Хебба является идея о том, что изменения в силе синапса зависят от взаимодействия нескольких факторов. В правиле обучения критика имеет место взаимодействие между сигналом подкрепления δ и следами приемлемости, которые зависят только от предсинаптических сигналов. Нейроученые называют это *двухфакторным правилом обучения*, поскольку взаимодействуют два сигнала, или величины. С другой стороны, правило обучения ис-

полнителя *трехфакторное*, потому что помимо зависимости от δ его следы приемлемости зависят от предсинаптической и постсинаптической активности. Но, в отличие от постулата Хебба, характер изменения синаптической силы критически зависит от временных соотношений факторов – следы приемлемости должны вклиниться, чтобы сигнал подкрепления смог воздействовать на синапсы, которые были активны в недавнем прошлом.

Некоторые тонкости хронометража сигналов в правилах обучения исполнителя и критика заслуживают более пристального внимания. При определении нейроноподобных блоков исполнителя и критика мы игнорировали то небольшое время, которое необходимо входу синапса, чтобы оказать влияние на пульсацию реального нейрона. Когда потенциал действия от предсинаптического нейрона достигает синапса, высвобождающиеся молекулы нейромедиатора диффундируют через синаптическую щель в сторону постсинаптического нейрона, где прикрепляются к рецепторам на поверхности последнего: это активирует молекулярный механизм, вызывающий пульсацию постсинаптического нейрона (или торможение его пульсации в случае тормозящего синаптического входа). Этот процесс может занимать несколько десятков миллисекунд. Но согласно формулам (15.1) и (15.2), поступление сигнала на вход блока критика или исполнителя немедленно порождает выходной сигнал. Подобное игнорирование времени активации общепринято в абстрактных моделях пластичности типа хеббовской, когда синаптическая сила изменяется как простое произведение одновременной пред- и постсинаптической активности. В более реалистичных моделях время активации необходимо принимать во внимание.

Время активации особенно важно для более реалистичного блока исполнителя, поскольку влияет на то, как должны работать контингентные следы приемлемости, чтобы правильно распределять поощрение за подкрепление между синапсами. Выражение $(A_t - \pi(1|S_t, \theta)x(S_t))$, определяющее контингентные следы приемлемости для приведенного выше правила обучения блока исполнителя, включает постсинаптический множитель $(A_t - \pi(1|S_t, \theta))$ и предсинаптический множитель $x(S_t)$. Это работает, потому что вследствие игнорирования времени активации предсинаптическая активность $x(S_t)$ является одной из *причин* постсинаптической активности, входящей в выражение $(A_t - \pi(1|S_t, \theta))$. Чтобы правильно распределить поощрение за подкрепление, предсинаптический множитель, определяющий след приемлемости, должен быть причиной постсинаптического множителя, который также определяет след. Контингентные следы приемлемости для более реалистичного блока исполнителя должны были бы учитывать время активации. (Не следует путать время активации со временем, необходимым нейрону, чтобы получить сигнал подкрепления, на которое влияет активность данного нейрона. Назначение следов приемлемости состоит в том, чтобы охватить этот временной интервал, который в общем случае гораздо дольше времени активации. Мы еще обсудим это в следующем разделе.)

Нейронауки дают подсказку о том, как этот процесс мог бы протекать в мозге. Нейроученые открыли форму хеббовской пластичности, названной *пластичность, зависящая от момента времени спайка* (spike-timing-dependent plasticity – STDP), которая делает правдоподобным существование в мозге синаптической пластичности по типу исполнителя. STDP – пластичность хеббовского типа, но изменение силы синапсов зависит от относительного хронометража пред- и постсинаптиче-

ского потенциала действия. Зависимость может принимать различные формы, но в одной из наиболее изученных сила синапса возрастает, если спайки, проходящие по этому синапсу, поступают незадолго до пульсации постсинаптического нейрона. Если временное соотношение обратное, т. е. предсинаптический спайк поступает спустя короткое время после пульсации постсинаптического нейрона, то сила синапса убывает. STDP – тип хеббовской пластичности, учитывающий время активации нейрона, т. е. одну из обязательных составных частей обучения по типу исполнителя.

Открытие STDP побудило нейроученых исследовать возможность трехфакторной формы STDP, при которой нейромодуляторный входной сигнал должен сопровождаться пред- и постсинаптическими спайками в соответствующие моменты времени. Эта форма синаптической пластичности, названная *модулированной вознаграждением STDP* (reward-modulated STDP), во многом напоминает обсуждаемое правило обучения исполнителя. Синаптические изменения, которые могли бы порождаться регулярной STDP, имеют место, только если нейромодуляторный входной сигнал поступает в течение временного окна после предсинаптического спайка, за которым почти сразу следует постсинаптический спайк. Появляется все больше фактов в пользу того, что модулированная вознаграждением STDP проявляется в стволах средних шипиковых нейронов дорсальной части полосатого тела, а дофамин играет роль нейромодулятора в тех участках, где происходит обучение исполнителя в гипотетической реализации алгоритма исполнитель–критик, показанной на рис. 15.5b. Эксперименты продемонстрировали, что модулированная вознаграждением STDP с устойчивыми изменениями силы кортико-стриарных синапсов возникает, только если нейромодуляторный импульс поступает в течение временного окна длительностью до 10 секунд, после того как предсинаптический спайк сопровождается почти сразу следующим за ним постсинаптическим спайком (Yagishita et al. 2014). Несмотря на косвенность свидетельства, эти эксперименты указывают на существование контингентных следов приемлемости с пролонгированной динамикой. Молекулярные механизмы, порождающие эти следы, а также гораздо более короткие следы, которые, вероятно, лежат в основе STDP, пока неясны, но исследования зависящей от времени и нейромодулятора синаптической пластичности продолжаются.

Описанный выше нейроноподобный блок исполнителя с правилом обучения в духе закона эффекта в более простой форме встречался в сети исполнитель–критик в работе Barto et al. (1983). Эта сеть была инспирирована гипотезой о «гедонистическом нейроне», предложенной физиологом А. Г. Клопфом (А. Н. Klopff, 1972, 1982). Не все детали гипотезы Клопфа согласуются с тем, что известно о синаптической пластичности, но открытие STDP и растущий массив свидетельств в пользу модулированной вознаграждением STDP наводят на мысль, что идеи Клопфа не так уж далеки от истины. Далее мы обсудим гипотезу Клопфа о гедонистическом нейроне.

15.9. Гедонистические нейроны

Формулируя гипотезу о гедонистическом нейроне, Клопф предположил, что отдельный нейрон стремится максимизировать разность между синаптическим входом, рассматриваемым как вознаграждение, и синаптическим входом, рас-

смаатриваемым как наказание, корректируя силы своих синапсов, исходя из последствий его собственных потенциалов действий. Иными словами, отдельные нейроны можно обучать с помощью контингентного отклику подкрепления, как животное в эксперименте по инструментальному обусловливанию. Гипотеза включала идею о том, что вознаграждения и наказания доставляются нейрону через один и тот же синаптический вход, который возбуждает или тормозит активность нейрона по генерации спайков. (Если бы Клопфу было известно то, что мы знаем сегодня о нейромодуляторных системах, то, возможно, он приписал бы подкрепляющую роль нейромодуляторному входу, но он хотел избежать любого централизованного источника информации для обучения.) Синаптически локальные следы прошлой пред- и постсинаптической активности обладали ключевой для гипотезы Клопфа функцией делать синапсы *приемлемыми* – этот термин ввел он – для модификации посредством более позднего вознаграждения или наказания. Он предположил, что эти следы реализуются молекулярными механизмами, локальными для каждого синапса и потому отличными от электрической активности пред- и постсинаптических нейронов. В разделе «Библиографические и исторические замечания» мы обратим внимание на похожие предположения других ученых.

Конкретно Клопф высказал предположение, что синаптическая сила изменяется следующим образом. Когда нейрон испускает потенциал действия, все его синапсы, которые вносили вклад в этот потенциал, становятся приемлемыми, т. е. готовы претерпеть изменение своей силы. Если спустя подходящее время за потенциалом действия следует увеличение вознаграждения, то силы всех приемлемых синапсов увеличиваются. Наоборот, если за потенциалом действия следует наказание, то силы приемлемых синапсов уменьшаются. Это реализуется срабатыванием следа приемлемости в синапсе при совпадении пред- и постсинаптической активности (точнее, при сопряжении предсинаптической активности с постсинаптической активностью, которой эта предсинаптическая активность поспособствовала) – что мы и называем контингентным следом приемлемости. По существу, это трехфакторное правило обучения блока исполнителя, описанное в предыдущем разделе.

Форма и динамика следа приемлемости в теории Клопфа отражают длительно-сти многих петель обратной связи, в которые вмонтирован нейрон; одни из них находятся целиком внутри мозга и тела организма, другие захватывают окружающую среду, опосредованные двигательной и сенсорной системой. Его идея состояла в том, что форма синаптического следа приемлемости похожа на гистограмму длительностей петель обратной связи, в которые вмонтирован нейрон. Тогда пик следа приемлемости должен приходиться на длительность самых распространенных петель обратной связи, в которых участвует нейрон. Следы приемлемости, используемые в описанных в этой книге алгоритмах, – это упрощенные варианты оригинальной идеи Клопфа, а именно экспоненциально (или геометрически) убывающие функции с параметрами λ и γ . Это упрощает теорию и имитационное моделирование, но мы рассматриваем эти простые следы приемлемости как замену следов, более близких исходной концепции Клопфа, которые давали бы вычислительные преимущества в сложных системах обучения с подкреплением благодаря уточненному процессу распределения поощрений.

Гипотеза Клопфа не так уж неправдоподобна, как может показаться на первый взгляд. Есть хорошо изученный пример одноклеточной бактерии *Escherichia coli*,

которая стремится к одним стимулам и избегает других. На перемещение этого одноклеточного организма влияют химические стимулы в окружающей среде, такое поведение называется хемотаксисом. Бактерия плавает в жидкой среде, поворачивая прикрепленные к поверхности волосовидные структуры, называемые жгутиками (да-да, она ими вращает!). Молекулы в окружающей бактерию среде прикрепляются к рецепторам на ее поверхности. События прикрепления модулируют частоту, с которой бактерия изменяет направление вращения жгутиков на противоположное. Каждое изменение приводит к тому, что бактерия поворачивается на месте, а затем начинает двигаться в новом случайном направлении. Небольшой объем химической памяти и вычислений позволяет уменьшать частоту инверсии жгутиков, когда бактерия плывет в сторону более высокой концентрации молекул, необходимых ей для выживания (аттрактантов), и увеличивать, когда впереди высокая концентрация вредных молекул (репеллентов). В результате бактерия стремится к существованию в окружении аттрактантов, избегая репеллентов.

Описанное выше хемотаксическое поведение называется клинокинезом. Это вид поведения на основе проб и ошибок, хотя маловероятно, что тут присутствует какое-то обучение: бактерии нужна лишь малая толика краткосрочной памяти, чтобы обнаруживать градиенты концентрации молекул, но, скорее всего, долгосрочной памяти у нее нет. Пионер искусственного интеллекта Оливер Селфридж (Oliver Selfridge) называл эту стратегию «беги и крутись», отмечая ее полезность в качестве базовой адаптивной стратегии: «продолжать движение в том же направлении, если ситуация улучшается, иначе повернуться» (Selfridge, 1978, 1984). По аналогии можно считать, что нейрон «плавает» (не буквально, конечно) в среде, состоящей из сложного набора петель обратной связи, в которые он вмонтирован, действуя так, чтобы получать входные сигналы одного типа и избегать остальных. Но, в отличие от бактерии, в синаптических силах нейрона сохраняется информация о его прошлом поведении. Если такой взгляд на поведение нейрона (или только одного типа нейронов) правдоподобен, то природа взаимодействия нейрона с окружающей средой, устроенная по типу контура с обратной связью, важна для понимания его поведения. Здесь окружающая среда нейрона состоит из всего остального организма животного и среды, с которой животное взаимодействует как целое.

Гипотеза о гедонистическом нейроне Клопфа не ограничивалась идеей о том, что отдельные нейроны являются агентами обучения с подкреплением. Он утверждал, что многие аспекты интеллектуального поведения можно интерпретировать как результат коллективного поведения популяции эгоистичных гедонистических нейронов, взаимодействующих друг с другом в громадном сообществе или экономической системе, каковую представляет собой нервная система животного. Полезен такой взгляд на нервную систему или нет, но у коллективного поведения агентов обучения с подкреплением имеются следствия для нейронауки. Этой темой мы и займемся далее.

15.10. Коллективное обучение с подкреплением

Поведение популяций агентов обучения с подкреплением имеет глубокие связи с изучением социальных и экономических систем, а если что-то, подобное гипотезе Клопфа о гедонистическом нейроне, справедливо, то и с нейронауками.

Описанное выше предположение о том, как алгоритм исполнитель–критик мог бы быть реализован в мозге, лишь рассматривает последствия того факта, что и дорсальная, и вентральная части полосатого тела – предположительные места расположения исполнителя и критика – содержат миллионы средних шипиковых нейронов, синапсы которых претерпевают изменения, модулированные физическими всплесками активности дофаминовых нейронов.

Посетителем на рис. 15.5а является однослойная сеть k блоков исполнителя. Действия, порождаемые этой сетью, – векторы $(A_1, A_2, \dots, A_k)^T$, предположительно управляющие поведением животного. Изменения силы синапсов всех этих блоков зависят от сигнала подкрепления δ . Поскольку блоки исполнителя стремятся максимально увеличить δ , эта величина, по существу, выступает для них в роли сигнала подкрепления (т. е. в этом случае подкрепление совпадает с вознаграждением). Таким образом, каждый блок исполнителя сам является агентом обучения с подкреплением – если угодно, гедонистическим нейроном. Теперь, чтобы до предела упростить ситуацию, предположим, что каждый блок получает один и тот же сигнал вознаграждения в одно и то же время (хотя, как указано выше, предположение о том, что дофамин высвобождается во всех кортико-стриарных синапсах при одинаковых условиях и одновременно, вероятно, является чрезмерным упрощением).

Что теория обучения с подкреплением говорит нам о том, что происходит, когда все члены популяции агентов обучаются на общем сигнале вознаграждения? В разделе, который называется *многоагентное обучение с подкреплением*, рассматриваются различные аспекты обучения таких популяций агентов. Хотя эта область выходит за рамки книги, мы полагаем, что некоторые ее основные концепции и результаты имеют прямое отношение к осмыслению диффузных нейромодуляторных систем мозга. В многоагентном обучении с подкреплением (и в теории игр) сценарий, при котором все агенты стремятся максимизировать общее одновременно получаемое вознаграждение, называется *кооперативной игрой*, или *командной задачей*.

Командная задача интересна тем, что общий сигнал вознаграждения, посылаемый каждому агенту, оценивает *паттерн* активности, порождаемой всей популяцией, т. е. *коллективное действие* членов команды. Это означает, что способность каждого отдельного агента влиять на сигнал вознаграждения ограничена, т. к. отдельный агент отвечает лишь за один компонент коллективного действия, оцениваемого общим сигналом вознаграждения. Для эффективного обучения в таких условиях необходимо решить *проблему структурного распределения поощрения*: какие члены команды или группы членов заслуживают поощрения за благоприятный сигнал вознаграждения или порицания за неблагоприятный сигнал? Это *кооперативная игра*, или командная задача, потому что агенты сообща стремятся увеличить один и тот же сигнал вознаграждения, между агентами нет конфликта интересов. Мы имели бы *конкурентную игру*, если бы разные агенты получали разные сигналы вознаграждения, каждый из которых оценивал бы коллективное действие популяции, а целью агента было бы увеличить собственный сигнал вознаграждения. В этом случае между агентами возможен конфликт интересов, выражающийся в том, что действия, хорошие для одних агентов, плохи для других. Даже решение о том, каким должно быть наилучшее коллективное действие, – нетривиальный аспект теории игр. Эта конкурентная постановка также

может представлять интерес для нейронаук (например, чтобы объяснить неоднородность активности дофаминовых нейронов), но сейчас нас занимает только кооперативный случай.

Как каждый агент обучения с подкреплением, входящий в команду, может обучиться «поступать правильно», чтобы коллективное действие команды получило высокое вознаграждение? Интересен следующий результат: если каждый агент может эффективно обучиться, несмотря на то, что его сигнал вознаграждения искажен сильным шумом, и несмотря на отсутствие доступа к полной информации о состоянии, то популяция в целом обучится предпринимать коллективные действия, которые улучшаются с точки зрения общего сигнала вознаграждения, даже в случае, когда агенты не могут взаимодействовать между собой. Каждый агент сталкивается со своей задачей обучения с подкреплением, в которой его влияние на сигнал вознаграждения замаскировано шумом, создаваемым влиянием других агентов. На самом деле для каждого агента все прочие агенты – часть окружающей среды, потому что поступающая ему информация – и та ее часть, которая несет информацию о состоянии, и часть, содержащая вознаграждение, – зависит от поведения других агентов. Кроме того, не имея доступа к действиям других агентов, точнее к параметрам, определяющим их стратегии, каждый агент может лишь частично наблюдать состояние окружающей среды. Из-за этого задача обучения каждого члена команды оказывается очень трудной, но если каждый применяет алгоритм обучения с подкреплением, способный увеличить сигнал вознаграждения даже в таких сложных условиях, то команда агентов обучения с подкреплением сможет обучиться коллективным действиям, которые будут улучшаться с точки зрения оценки, которую несет общий сигнал вознаграждения.

Если членами команды являются нейроноподобные блоки, то задача каждого блока – увеличить вознаграждение, получаемое со временем, – как поступает блок исполнителя, описанный в разделе 15.8. У алгоритма обучения каждого блока есть две важные особенности. Во-первых, он должен использовать контингентные следы приемлемости. Напомним, что в терминах биологических нейронов контингентный след приемлемости иницируется (или увеличивается) в синапсе, когда предсинаптический вход является одной из причин пульсации постсинаптического нейрона. Наоборот, неконтингентный след приемлемости иницируется или увеличивается предсинаптическим входом вне зависимости от того, что делает постсинаптический нейрон. В разделе 15.8 было объяснено, что, сохраняя информацию о том, какие действия были предприняты в каких состояниях, контингентные следы приемлемости позволяют распределять поощрение за вознаграждение или порицание за наказание между параметрами стратегии агента в соответствии со вкладом значений этих параметров в определение действия агента. Если рассуждать подобным образом, то член команды должен запоминать свое недавнее действие, так чтобы он мог увеличить или уменьшить вероятность совершения этого действия в зависимости от полученного впоследствии сигнала вознаграждения. Компонент действия, являющийся частью контингентного следа приемлемости, реализует такое запоминание действия. Но в силу трудности задачи обучения контингентная приемлемость – лишь предварительный шаг процесса распределения поощрения: связь между действием одного члена команды и изменениями командного сигнала вознаграждения.

раждения представляет собой статистическую корреляцию, для оценки которой нужно много испытаний. Контингентная приемлемость – важный, но предварительный шаг этого процесса.

Обучение с неконтингентными следами приемлемости вообще не работает в командной задаче, потому что в нем отсутствует способ корреляции действий с последующими изменениями сигнала вознаграждения. Неконтингентные следы приемлемости подходят для обучения предсказанию в качестве критика в алгоритме исполнитель–критик, но не поддерживают обучение управлению, как должен делать исполнитель. Члены популяции агентов, подобных критику, по-прежнему могут получать общий сигнал подкрепления, но все они должны были бы обучиться предсказывать одну и ту же величину (каковой в случае метода исполнитель–критик был бы ожидаемый доход для текущей стратегии). Насколько успешен был бы каждый член популяции, обучаемой предсказывать ожидаемый доход, зависело бы от получаемой им информации, которая может сильно различаться для разных членов популяции. У популяции не было бы нужды порождать дифференцированные паттерны активности. Это не совпадает с данным нами определением командной задачи.

Второе требование к коллективному обучению в командной задаче заключается в том, что в действиях членов команды должна присутствовать изменчивость, чтобы команда могла исследовать пространство коллективных действий. Простейший способ добиться этого – сделать так, чтобы каждый член команды мог независимо исследовать собственное пространство действий, обеспечив изменчивость своего выхода. В результате действия команды как целого тоже будут варьироваться. Например, команда блоков исполнителей, описанных в разделе 15.8, исследует пространство коллективных действий, потому что выход каждого блока, будучи логистическим блоком Бернулли, вероятностным образом зависит от взвешенной суммы элементов входного вектора. Взвешенная сумма смещает вероятность испускания импульса вверх или вниз, но изменчивость присутствует всегда. Поскольку каждый блок применяет алгоритм градиента стратегии REINFORCE (глава 13), он корректирует свои веса с целью максимизировать средний темп вознаграждения, наблюдаемый в процессе стохастического исследования своего пространства действий. Можно показать (см. работу Williams, 1992), что команда логистических блоков Бернулли с алгоритмом REINFORCE реализует алгоритм градиента стратегии как *единое целое* – относительно среднего темпа общего сигнала вознаграждения команды, – когда под действиями понимаются коллективные действия команды.

Далее Уильямс показал, что команда логистических блоков Бернулли с алгоритмом REINFORCE осуществляет подъем по среднему градиенту вознаграждения, если составляющие ее блоки взаимосвязаны и образуют многослойную ИНС. В этом случае сигнал вознаграждения транслируется всем блокам сети, хотя вознаграждение может зависеть только от коллективных действий выходных блоков сети. Это означает, что многоуровневая команда логистических блоков Бернулли с алгоритмом REINFORCE ведет себя как многослойная сеть, обучаемая популярным методом обратного распространения ошибки, только в этом случае процесс обратного распространения заменен трансляцией сигнала вознаграждения. На практике метод обратного распространения ошибки заметно быстрее, но применение биологическими нейронами метода командного обучения с подкреплени-

нием более правдоподобно, особенно в свете наших знаний о модулированной вознаграждением STDP (см. раздел 15.8).

Независимое исследование членами команды – лишь простейший вид командного исследования; возможны и более сложно организованные методы, если члены команды координируют свои действия, чтобы сконцентрироваться на конкретных частях пространства коллективных действий – путем взаимодействия друг с другом или путем ответа на общие входные сигналы. Существуют также более сложные, чем контингентные следы приемлемости, механизмы решения проблемы структурного распределения поощрения, которое оказывается проще в командной задаче, когда множество возможных коллективных действий как-то ограничено. Предельный случай – конфигурация «победитель забирает все» (например, результат латерального торможения в мозге), которая ограничивает коллективные действия такими, в которые вносит вклад только один или небольшое число членов команды. В таком случае победители получают поощрение или порицание за полученное в результате вознаграждение либо наказание.

Детали обучения в кооперативных играх (или командных задачах) и некооперативных играх выходят за рамки данной книги. В разделе «Библиографические и исторические замечания» приведена подборка публикаций на эту тему, включая многочисленные ссылки на исследования, посвященные влиянию коллективного обучения с подкреплением на нейронауки.

15.11. ОСНОВАННЫЕ НА МОДЕЛИ МЕТОДЫ В МОЗГЕ

Имеются факты в пользу того, что различие между основанными на модели и безмодельными алгоритмами обучения с подкреплением полезно при осмыслении процессов обучения и принятия решений у животных. В разделе 14.6 обсуждаются параллели между ним и различием между привычным и целеустремленным поведением животных. Высказанная выше гипотеза о том, как мозг мог бы реализовать алгоритм исполнитель–критик, относится только к привычному поведению, потому что метод исполнитель–критик в основе своей безмодельный. Какие нейронные механизмы отвечают за возникновение целеустремленного поведения и как они взаимодействуют с теми, что отвечают за привычное поведение?

Один из способов изучения вопросов о мозговых структурах, участвующих в этих видах поведения, состоит в том, чтобы отключить какую-то зону в мозге крысы, а затем понаблюдать за тем, что делает крыса в эксперименте по снижению ценности желаемого исхода (раздел 14.6). Результаты подобных экспериментов показывают, что описанная выше гипотеза об исполнителе и критике, согласно которой исполнитель помещается в дорсальную часть полосатого тела, чрезмерно упрощенная. Отключение одной зоны дорсальной части полосатого тела, а именно дорсолатерального стриатума (ДЛС), нарушает привычное обучение, заставляя животное больше полагаться на целеустремленные процессы. С другой стороны, отключение дорсомедиального стриатума (ДМС) нарушает целеустремленные процессы, заставляя животное больше полагаться на привычное обучение. Такие результаты подтверждают точку зрения, согласно которой ДЛС у грызунов сильнее вовлечен в безмодельные процессы, тогда как ДМС – в процессы, основанные на модели. Подобные эксперименты с людьми с применением функциональной

нейровизуализации, а также с высшими приматами, свидетельствуют, что аналогичные структуры в мозге приматов по-разному участвуют в привычном и целеустремленном режимах поведения.

В других исследованиях активность, связанная с основанными на модели процессами, выявлена в префронтальной коре человеческого мозга – самой передней части фронтальной коры, участвующей в исполнительных функциях, в т. ч. планировании и принятии решений. Конкретно участвует орбитофронтальная кора (ОФК) – часть префронтальной коры, расположенная прямо над глазами. Функциональная нейровизуализация мозга человека, а также регистрация активности одиночных нейронов у обезьян свидетельствуют о сильной активности в зоне ОФК, относящейся к субъективной оценке вознаграждения биологически значимых стимулов, а также к активности, связанной с вознаграждением, ожидаемым в качестве следствия действий. Хотя эти результаты и не свободны от противоречий, они все же подтверждают участие ОФК в целенаправленном выборе. Это может быть критически важно для той части модели окружающей среды у животного, которая связана с вознаграждением.

Еще одна структура, вовлеченная в поведение, основанное на модели, – гиппокамп, отвечающий за память и навигацию в пространстве. Гиппокамп крысы играет важнейшую роль в ее способности целенаправленно проходить лабиринт, это привело Толмена к выводу о том, что животные пользуются моделями, или когнитивными картами, при выборе действий (раздел 14.5). Гиппокамп может также быть одним из основных компонентов способности человека формировать мысленный образ нового опыта (Hassabis and Maguire, 2007; Ólafsdóttir, Barry, Saleem, Hassabis, and Spiers, 2015).

Открытия, которые позволяют прямо предположить участие гиппокампа в планировании – процессе, необходимом для привлечения модели окружающей среды к принятию решений, – были сделаны в экспериментах по декодированию активности нейронов гиппокампа с целью выяснить, какая часть пространственной активности гиппокампа представлена в каждый момент времени. Когда крыса останавливается в раздумье в лабиринте, представление пространства в гиппокампе разворачивается вперед (а не назад) вдоль возможных путей, которые животное может выбрать в данной точке (Johnson and Redish, 2007). Более того, пространственные траектории, представленные такими разворотами, соответствуют последующему навигационному поведению крысы (Pfeiffer and Foster, 2013). Эти результаты позволяют предположить, что гиппокамп играет важнейшую роль в той части модели окружающей среды животного, которая связана с переходом состояний, и что он является частью системы, которая применяет модель для имитации возможных будущих последовательностей состояний с целью оценить последствия того или иного образа действий, т. е. формы планирования.

Описанные выше результаты пополняют обширную литературу по нейронным механизмам, лежащим в основе целеустремленного, или основанного на модели, обучения и принятия решений, но многие вопросы еще остаются без ответа. Например, как области, настолько структурно похожие, как ДЛС и ДМС, могут быть существенными компонентами столь непохожих режимов обучения и поведения, как безмодельные и основанные на модели алгоритмы? Верно ли, что за компоненты модели окружающей среды, связанные с переходами состояний (так мы их называем) и вознаграждением, отвечают разные структуры? Все ли планирование

осуществляется в момент принятия решений посредством имитации возможных в будущем вариантов действий, как можно предположить из наблюдений за развитием представления пространства вперед в гиппокампе? Иными словами, все ли планирование похоже на алгоритм разыгрывания (раздел 8.10)? Или иногда модели привлекаются в фоновом режиме, чтобы уточнить или пересчитать информацию о ценности, как делается в архитектуре Дупа (раздел 8.2)? Как мозг осуществляет выбор между системами привычного и целеустремленного поведения? И есть ли вообще четкое разделение между нейронными субстратами этих систем?

Фактов, подтверждающих положительный ответ на последний вопрос, нет. Резюмируя ситуацию, Doll, Simon, and Daw (2012) писали, что «основанные на модели влияния, похоже, присутствуют чуть ли не везде, где мозг обрабатывает информацию о вознаграждении». Сюда относятся сами дофаминовые сигналы, которые могут свидетельствовать о влиянии основанной на модели информации, помимо ошибок предсказания вознаграждения, которые, как полагают, лежат в основе безмодельных процессов.

Продолжающиеся нейронаучные исследования, в которых учитывается различие между безмодельным и основанным на модели обучении с подкреплением, потенциально способны улучшить наше понимание привычных и целеустремленных процессов в мозге. А это, в свою очередь, может привести к алгоритмам, объединяющим безмодельные и основанные на модели методы способами, еще неизвестными в вычислительном обучении с подкреплением.

15.12. НАРКОТИЧЕСКАЯ ЗАВИСИМОСТЬ

Понимание нейронных основ наркотической зависимости – одна из самых важных задач нейронаук. Ее решение может привести к разработке новых подходов к этой серьезной проблеме здравоохранения. Есть мнение, что пристрастие к наркотикам – результат тех же процессов мотивации и обучения, которые побуждают нас искать естественные пути вознаграждения, отвечающие нашим биологическим потребностям. Наркотические вещества оказывают сильный подкрепляющий эффект и тем самым привлекают на свою сторону наши естественные механизмы обучения и принятия решений. Это правдоподобное объяснение, учитывая, что многие – хотя и не все – вещества, вызывающие пристрастие, прямо или косвенно повышают уровень дофамина в областях вокруг окончаний аксонов дофаминовых нейронов в полосатом теле – структуре мозга, принимающей важное участие в нормальном обучении на основе вознаграждения (раздел 15.7). Но ведущее к самоуничтожению поведение, ассоциируемое с наркотической зависимостью, нехарактерно для нормального обучения. В чем отличие опосредованного дофамином обучения, когда вознаграждение является результатом приема наркотика? Является ли зависимость результатом нормального обучения в ответ на вещества, которые не были широко доступны на протяжении нашей эволюции, из-за чего в процессе естественного отбора не выработалась защита от их разрушительных последствий? Или наркотики как-то препятствуют нормальному опосредованному дофамином обучению?

Гипотеза об ошибке предсказания вознаграждения в результате активности дофаминовых нейронов и ее связь с TD-обучением – основа принадлежащей Реди-

шу (Redish, 2004) модели некоторых – но, конечно, не всех – особенностей наркотической зависимости. Модель основана на том наблюдении, что прием кокаина и других наркосодержащих веществ вызывает транзиторное повышение уровня дофамина. Предполагается, что этот выброс дофамина увеличивает TD-ошибку δ , так что это нельзя уравновесить изменением функции ценности. Иными словами, в то время как обычно δ уменьшается до такого уровня, при котором предшествующими событиями предсказывается нормальное вознаграждение (раздел 15.6), вклад, привнесенный в δ наркотическим стимулом, не уменьшается, когда сигнал вознаграждения становится предсказанным, т. е. вознаграждение вследствие наркотика нельзя «погасить предсказанием». Модель достигает этого тем, что предотвращает падение δ ниже нуля, когда сигнал вознаграждения обусловлен наркотиком, и, следовательно, сводит на нет характерную особенность TD-обучения – исправление ошибок – для состояний, ассоциированных с приемом наркотика. В результате ценности этих состояний растут неограниченно, а стало быть, ведущие в эти состояния действия оказываются предпочтительнее всех остальных.

Зависимое поведение гораздо сложнее, чем этот результат, следующий из модели Редиша, но основная идея этой модели, возможно, является кусочком головоломки. А возможно, модель ведет в никуда. Не похоже, что дофамин играет критическую роль во всех формах наркозависимости, и не все люди одинаково подвержены развитию зависимого поведения. К тому же модель не учитывает изменения во многих нервных цепях и участках мозга, сопровождающие хронический прием наркотиков, например изменения, которые приводят к уменьшению действия вещества при продолжающемся использовании. Вероятно также, что в формировании зависимости участвуют основанные на модели процессы. Тем не менее модель Редиша иллюстрирует, как теорию обучения с подкреплением можно привлечь к пониманию важной проблемы здравоохранения. Эта теория внесла также значительный вклад в развитие новой дисциплины – вычислительной психиатрии, цель которой – лучше понять психические расстройства за счет применения математических и вычислительных методов.

15.13. РЕЗЮМЕ

Нервные пути, участвующие в мозговой системе вознаграждения, сложны и не вполне понятны, но нейронаучные исследования, направленные на понимание этих путей и их роли в поведении, быстро прогрессируют. В ходе этих исследований открываются поразительные параллели между мозговой системой вознаграждения и представленной в этой книге теорией обучения с подкреплением.

Гипотеза об ошибке предсказания вознаграждения в результате активности дофаминовых нейронов предложена учеными, осознавшими удивительную параллель между поведением TD-ошибок и активностью нейронов, продуцирующих дофамин – нейромедиатор, играющий важную роль в ориентированном на вознаграждение обучении и поведении млекопитающих. Эксперименты, поставленные в конце 1980-х и в 1990-х годах в лаборатории Вольфрама Шульца, показали, что дофаминовые нейроны реагируют на события вознаграждения резкими всплесками активности, которые получили название «фазический отклик», только если

животное не ожидает этих событий. Это позволяет предположить, что дофаминовые нейроны сигнализируют об ошибках предсказания вознаграждения, а не о самом вознаграждении. Кроме того, данные эксперименты показали, что по мере того, как животное обучается предсказывать событие вознаграждения на основе предшествующих сенсорных ориентиров, фазическая активность дофаминовых нейронов смещается в сторону более ранних предсказательных ориентиров, а для более поздних уменьшается. Это соответствует эффекту обратного обновления TD-ошибки по мере того, как агент обучения с подкреплением обучается предсказывать вознаграждение.

Другие экспериментальные результаты уверенно подтверждают, что фазическая активность дофаминовых нейронов является сигналом подкрепления для обучения, который доходит до нескольких участков мозга посредством широко разветвленной аксональной сети у продуцирующих дофамин нейронов. Эти результаты согласуются с различием, которое мы проводим между сигналом вознаграждения R_t и сигналом подкрепления, каковым в большинстве описанных алгоритмов является TD-ошибка δ_t . Фазические отклики дофаминовых нейронов – это сигналы подкрепления, а не вознаграждения.

Преобладает гипотеза о том, что мозг реализует нечто, подобное алгоритму исполнитель–критик. Две мозговые структуры (дорсальная и вентральная части полосатого тела), играющие важную роль в обучении на основе вознаграждения, могут функционировать как исполнитель и критик. Тот факт, что TD-ошибка является сигналом подкрепления и для исполнителя, и для критика, хорошо согласуется с тем, что аксоны дофаминовых нейронов подходят к дорсальной и вентральной частям полосатого тела, с тем, что дофамин, похоже, очень важен для модулирования синаптической пластичности в обеих структурах, и с тем, что влияние на целевую структуру такого нейромодулятора, как дофамин, зависит от свойств самой целевой структуры, а не только от свойств нейромодулятора.

Исполнителя и критика можно реализовать с помощью ИНС, состоящей из нейроноподобных блоков, с правилами обучения, основанными на методе градиента стратегии исполнитель–критик, описанном в разделе 13.5. Каждую связь в этих сетях можно уподобить синапсу, соединяющему нейроны в мозге, а правила обучения соответствуют правилам, регулирующим изменение силы синапсов как функции от активности пред- и постсинаптических нейронов и нейромодуляторного входа, являющегося аналогом входа от дофаминовых нейронов. При такой формулировке у каждого синапса есть свой след приемлемости, который запоминает прошлую активность с участием этого синапса. Единственное различие между правилами обучения исполнителя и критика состоит в использовании разных следов приемлемости: следы блока критики неkontингентны, потому что не зависят от выхода этого блока, а следы блока исполнителя контингентны, потому что зависят не только от входа, но и от выхода блока исполнителя. В гипотетической реализации системы исполнитель–критик в мозге эти правила обучения соответствуют правилам, регулирующим пластичность кортико-стриарных синапсов, которые проводят сигналы от коры мозга к главным нейронам в дорсальной и вентральной частях полосатого тела, – синапсам, которые получают также сигналы от дофаминовых нейронов.

Правило обучения блока исполнителя в сети исполнитель–критик хорошо соответствует *модулированной вознаграждением пластичности, зависящей от мо-*

мента времени спайка (STDP). В случае обычной STDP соотношение моментов пред- и постсинаптической пластичности определяет направление синаптического изменения. В случае модулированной вознаграждением STDP изменения в синапсах зависят также от нейромодулятора, например дофамина, который поступает в течение временного окна длительностью до 10 секунд после выполнения условий, необходимых для STDP. Появляются все новые факты в пользу того, что модулированная вознаграждением STDP имеет место в кортико-стриарных синапсах, где происходит обучение в гипотетической нейронной реализации системы исполнитель–критик. Они усиливают правдоподобность гипотезы о том, что в мозге некоторых животных существует нечто, подобное системе исполнитель–критик.

Идея синаптической приемлемости и основные особенности правила обучения исполнителя вытекают из гипотезы Клопфа о «гедонистическом нейроне» (Klopf, 1972, 1981). Клопф предполагал, что отдельные нейроны стремятся получить вознаграждение и избежать наказания, изменяя силу синапсов в соответствии с тем, какие последствия имели их потенциалы действия: вознаграждение или наказание. Активность нейрона может повлиять на поступающий к нему впоследствии сигнал, потому что нейрон вмонтирован во многие петли обратной связи, одни из которых находятся целиком внутри нервной системы и тела животного, а другие захватывают окружающую его среду. Идея приемлемости Клопфа заключается в том, что синапсы временно маркируются как приемлемые для модификации, если участвовали в пульсации нейрона (тем самым это является контингентной формой следа приемлемости). Сила синапса модифицируется, если подкрепляющий сигнал поступает, когда синапс приемлем. Мы привели хемотаксическое поведение бактерии как пример одноклеточного организма, направляющего свои движения, так чтобы встретиться с одними молекулами и избежать встречи с другими.

Обращает на себя внимание особенность дофаминовой системы, состоящая в том, что волокна, высвобождающие дофамин, проникают в несколько участков мозга. Конечно, есть вероятность, что только некоторые популяции дофаминовых нейронов транслируют один и тот же сигнал подкрепления, но если этот сигнал достигает синапсов многих нейронов, участвующих в обучении по типу исполнителя, то ситуация моделируется как командная задача. В такой задаче каждый член совокупности агентов обучения с подкреплением получает один и тот же сигнал подкрепления, зависящий от активности всех членов совокупности – команды. Если каждый член команды применяет достаточно мощный алгоритм обучения, то команда сможет коллективно обучиться улучшать результаты своей деятельности, оцениваемые глобально транслируемым сигналом подкрепления, даже если члены команды не взаимодействуют друг с другом напрямую. Это согласуется с широким распространением дофаминовых сигналов в мозге и дает биологически правдоподобную альтернативу популярному методу обратного распространения ошибки для обучения многослойных сетей.

Различие между безмодельным и основанным на модели обучением с подкреплением помогает нейрочеловеку исследовать биологические основы привычного и целеустремленного обучения и принятия решений. Современные результаты указывают на существование зон мозга, в большей степени вовлеченных в один из этих процессов, нежели в другой, но общая картина остается неясной, пото-

му что между этими процессами, похоже, нет четкой границы в мозге. Многие вопросы пока не имеют ответов. Быть может, самым интригующим является тот факт, что гиппокамп, который традиционно принято связывать с памятью и навигацией в пространстве, похоже, участвует в моделировании возможных в будущем способов действия как части процесса принятия решений у животных. Это позволяет предположить, что он является частью системы, которая использует модель окружающей среды для планирования.

Теория обучения с подкреплением также оказывает влияние на исследование нейронных процессов, лежащих в основе наркотической зависимости. Модель некоторых черт зависимости основана на гипотезе об ошибке предсказания вознаграждения. Предполагается, что вызывающее зависимость вещество, например кокаин, дестабилизирует TD-обучение, приводя к неограниченному росту ценности действий, ассоциированных с приемом наркотика. Это еще далеко от полной модели зависимости, но показывает, как вычислительная точка зрения может подсказывать теории, которые можно проверить в будущих экспериментах. Новая дисциплина, вычислительная психиатрия, аналогичным образом фокусируется на применении вычислительных моделей, в т. ч. заимствованных из обучения с подкреплением, с целью лучше понять психические расстройства.

В этой главе мы лишь слегка коснулись вопроса о взаимном влиянии нейронаук и развития обучения с подкреплением в информатике и технике. Своим дизайном алгоритмы обучения с подкреплением в большинстве своем обязаны чисто вычислительным соображениям, но на некоторые аспекты оказали влияние гипотезы о механизмах обучения животных. Удивительно, что по мере накопления экспериментальных фактов о мозговых процессах вознаграждения многие особенности алгоритмов обучения с подкреплением, возникшие по чисто вычислительным причинам, оказались согласованы с нейронаучными данными. Может статься, что и у других аспектов обучения с подкреплением, таких как следы приемлемости и способность команды агентов обучаться коллективным действиям под влиянием глобально транслируемого сигнала подкрепления, могут появиться параллели с экспериментальными данными, по мере того как нейроученые продолжают раскрывать тайны основанного на вознаграждении обучения и поведения животных.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

Количество публикаций, в которых рассматриваются параллели между обучением и принятием решений в нейронауках и подходом к обучению с подкреплением, излагаемым в этой книге, огромно. Мы можем упомянуть только небольшую часть. Хорошей отправной точкой могут послужить работы Niv (2009), Dayan and Niv (2008), Glimcher (2011), Ludvig, Bellemaire, and Pearson (2011) и Shah (2012). Теория обучения с подкреплением наряду с экономикой, эволюционной биологией и математической психологией помогает формулировать количественные модели нейронных механизмов выбора у людей и других приматов. Эта глава посвящена в основном обучению, нейронаучный подход к вопросам принятия решений в ней затронут лишь вскользь. Книга Glimcher (2003) является введением в нейроэкономику – дисциплину, в которой обучение с подкреплением применяется

для изучения нейронных основ принятия решений с точки зрения экономики. См. также Glimcher and Fehr (2013). Учебником по вычислительному и математическому моделированию в нейронауках является книга Dayan and Abbott (2001), в которой рассматривается в том числе роль обучения с подкреплением. В книге Sterling and Laughlin (2015) исследуются нейронные основы обучения в терминах общих принципов проектирования, обеспечивающих эффективное адаптивное поведение.

- 15.1 Есть много хороших изложений основ нейронаук. Книга Kandel, Schwartz, Jessell, Siegelbaum, and Hudspeth (2013) – авторитетный и очень полный источник.
- 15.2 В работе Berridge and Kringelbach (2008) приведен обзор неврологических основ вознаграждения и удовольствия и отмечается, что обработка вознаграждения многогранна и задействует много нейронных систем. Нехватка места не позволяет нам обсудить влиятельное исследование Berridge and Robinson (1998), где проводится различие между гедонистическим воздействием стимула, который они называют «склонностью», и его мотивационным эффектом, названным «желанием». В работе Hare, O'Doherty, Camerer, Schultz, and Rangel (2008) исследованы нейронные основы относящихся к ценности сигналов с точки зрения экономики и проводится различие между ценностью цели, ценностью решения и ошибками предсказания. Ценность решения – это ценность цели за вычетом стоимости действия. См. также Rangel, Camerer, and Montague (2008), Rangel and Hare (2010) и Peters and Büchel (2010).
- 15.3 Гипотеза об ошибке предсказания вознаграждения в результате активности дофаминовых нейронов наиболее полно обсуждается в работе Schultz, Dayan, and Montague (1997). Впервые эта гипотеза была явно сформулирована в работе Montague, Dayan, and Sejnowski (1996). Они говорили об ошибках предсказания вознаграждения (ОПВ), но не конкретно о TD-ошибках, однако из обоснования гипотезы ясно следует, что имеются в виду именно TD-ошибки. Первое известное нам упоминание о связи между TD-ошибками и дофамином – работа Montague, Dayan, Nowlan, Pouget, and Sejnowski (1993), где предложено модулированное TD-ошибкой хеббовское правило обучения, подсказанное результатами по дофаминовой сигнализации, полученными группой Шульца. Указание на эту связь имеется также в реферате Quartz, Dayan, Montague, and Sejnowski (1992). В работе Montague and Sejnowski (1994) подчеркнута важность предсказания в мозге и в общих чертах описано, как предсказательное хеббовское обучение, модулированное TD-ошибками, можно было бы реализовать посредством диффузной нейромодуляторной системы типа дофаминовой. В работе Friston, Tononi, Reeke, Sporns, and Edelman (1994) представлена модель зависящего от ценности обучения в мозге, в которой изменения синапсов опосредованы TD-подобной ошибкой в виде глобального нейромодуляторного сигнала (хотя они не выделяли конкретно дофамин). В работе Montague, Dayan, Person, and Sejnowski (1995) представлена модель поиска пищи медоносной пчелой, в которой используется TD-ошибка. Эта модель основана на исследованиях Хаммера, Менцеля и их сотрудников (Hammer and Menzel,

1995; Hammer, 1997), доказавших, что нейромодулятор октопамин действует как сигнал подкрепления у медоносной пчелы. В работе Montague et al. (1995) указано, что дофамин играет аналогичную роль в мозге позвоночных. В статье Barto (1995a) проведена параллель между архитектурой исполнитель–критик и цепями в подкорковых ядрах и обсуждается связь между TD-обучением и основными результатами группы Шульца. В работе Houk, Adams, and Barto (1995) высказано предположение о том, как TD-обучение и архитектура исполнитель–критик могут отображаться на анатомию, физиологию и молекулярные механизмы подкорковых ядер. В работе Doya and Sejnowski (1998) более раннее исследование тех же авторов распространено на обучение птиц пению (Doya and Sejnowski, 1995); для этого включена TD-подобная ошибка, идентифицируемая дофамином, чтобы подкрепить выбор подлежащего запоминанию слухового входного сигнала. В работах O'Reilly and Frank (2006) и O'Reilly, Frank, Hazy, and Watz (2007) приводятся аргументы в пользу того, что фазические дофаминовые сигналы – это ОПВ, но не TD-ошибки. В подтверждение своей теории авторы приводят результаты с переменным межстимульным интервалом, которые не согласуются с предсказаниями простой TD-модели, а также тот факт, что обусловливание выше второго порядка наблюдается редко, тогда как TD-обучение не столь ограничено. В работе Dayan and Niv (2008) обсуждается «хорошо, плохо и отвратительно» в том, что касается согласования теории обучения с подкреплением и гипотезы об ошибке предсказания вознаграждения с экспериментальными данными. В работе Glimcher (2011) приведен обзор эмпирических фактов в пользу гипотезы об ошибке предсказания вознаграждения и подчеркнута важность этой гипотезы в современных нейронауках.

- 15.4** Работа Graybiel (2000) представляет собой краткое введение в подкорковые ядра. Упомянутые в ней эксперименты, включающие оптогенетическую активацию дофаминовых нейронов, были описаны в работах Tsai, Zhang, Adamantidis, Stuber, Bonci, de Lecea, and Deisseroth (2009), Steinberg, Keiin, Boivin, Witten, Deisseroth, and Janak (2013) и Claridge-Chang, Roorda, Vrontou, Sjulson, Li, Hirsh, and Miesenböck (2009). В работах Fiorillo, Yun, and Song (2013), Lammel, Lim, and Malenka (2014) и Saddoris, Cacciapaglia, Wightman, and Carelli (2015) показано, что сигнальные свойства дофаминовых нейронов специализированы для различных целевых зон. Нейроны, сигнализирующие об ОПВ, могут принадлежать к одной из нескольких популяций дофаминовых нейронов с разными целями и разными функциями. В работе Eshel, Tian, Bukwich, and Uchida (2016) обнаружена однородность откликов дофаминовых нейронов, несущих информацию об ошибке предсказания вознаграждения, в латеральной зоне VTA в ходе экспериментов по классическому обусловливанию мышей, хотя полученные результаты не указывают на разнообразие откликов при рассмотрении более широких областей мозга. В работе Gershman, Pesaran, and Daw (2009) изучались задачи обучения с подкреплением, которые можно разложить на независимые компоненты с отдельными сигналами вознаграждения, и в данных нейровизуализации человека были найдены свидетельства в пользу того, что мозг пользуется такого рода структурой.

- 15.5** Обзорная статья Шульца (Schultz, 1998) – прекрасное введение в очень обширную литературу по сигнализации предсказания вознаграждения дофаминовыми нейронами. В работах Berns, McClure, Pagnoni, and Montague (2001), Breiter, Aharon, Kahneman, Dale, and Shizgal (2001), Pagnoni, Zink, Montague, and Berns (2002) и O’Doherty, Dayan, Friston, Critchley, and Dolan (2003) описываются результаты функциональной нейровизуализации, подтверждающие существование сигналов, похожих на TD-ошибки, в мозге человека.
- 15.6** Этот раздел в основных чертах повторяет работу Barto (1995a), где объясняется, как TD-ошибки моделируют основные результаты группы Шульца по фазическим откликам дофаминовых нейронов.
- 15.7** Этот раздел основан преимущественно на работах Takahashi, Schoenbaum, and Niv (2008) и Niv (2009). Насколько нам известно, в работах Barto (1995a) и Houk, Adams, and Barto (1995) впервые опубликованы рассуждения о возможных реализациях алгоритмов исполнитель–критик в подкорковых ядрах. На основе результатов функциональной МРТ людей, участвовавших в эксперименте по инструментальному обусловливанию, O’Doherty, Dayan, Schultz, Deichmann, Friston, and Dolan (2004) высказали предположение, что исполнитель и критик, скорее всего, находятся соответственно в дорсальной и вентральной частях полосатого тела. Работа Gershman, Moustafa, and Ludvig (2014) посвящена представлению времени в моделях подкорковых ядер на основе обучения с подкреплением; обсуждаются факты в пользу различных вычислительных подходов к представлению времени и их следствия. Гипотетическая нейронная реализация архитектуры исполнитель–критик, описанная в этом разделе, содержит очень мало деталей известной анатомии и физиологии подкорковых ядер. Помимо более детальной гипотезы, описанной в работе Houk, Adams, and Barto (1995), существует ряд других гипотез, которые учитывают специфические связи с анатомией и физиологией и претендуют на объяснение дополнительных данных. К ним относятся гипотезы, предложенные в работах Suri and Schultz (1998, 1999), Brown, Bullock, and Grossberg (1999), Contreras-Vidal and Schultz (1999), Suri, Bargas, and Arbib (2001), O’Reilly and Frank (2006), и O’Reilly, Frank, Hazy, and Watz (2007). В работе Joel, Niv, and Ruppin (2002) содержится критический анализ анатомической правдоподобности некоторых из этих моделей и представлена альтернатива, призванная включить кое-какие особенности нейронных связей в подкорковых ядрах, которым ранее не уделяли внимания.
- 15.8** Обсуждаемое здесь правило обучения исполнителя более сложное, чем то, что было предложено в ранней сети исполнитель–критик из работы Barto et al. (1983). В той сети следами приемлемости блока исполнителя были следы одного лишь $A_t \times x(S_t)$, а не полного $(A_t - \pi(A_t | S_t)) x(S_t)$. В оригинальной работе не использовались достижения теории градиента стратегии, представленные в главе 13, и результаты Уильямса (Williams, 1986, 1992), который показал, как ИНС, состоящая из логистических блоков Бернулли, могла бы реализовать метод градиента стратегии.
- В работе Reynolds and Wickens (2002) предложено трехфакторное правило для синаптической пластичности на кортико-стриарном тракте, в котором

дофамин модулирует изменения силы кортико-стриарных синапсов. Обсуждаются экспериментальные подтверждения такого правила обучения и его возможная молекулярная основа. Убедительная демонстрация пластичности, зависящей от момента времени спайка (STDP), приписывается работе Markram, Lübke, Frotscher, and Sakmann (1997), в которой использованы добытые в более ранних экспериментах Levy and Steward (1983) и других факты, свидетельствующие о том, что относительное время пред- и постсинаптических спайков критически важно для индуцирования изменений синаптической эффективности. В работе Rao and Sejnowski (2001) высказано предположение о том, что STDP могла бы быть результатом TD-подобного механизма в синапсах с неkontингентными следами приемлемости длительностью примерно 10 миллисекунд. В работе Dayan (2002) отмечено, что для этого понадобилась бы такая ошибка, какая описана в ранней модели классического обусловливания Sutton and Barto's (1981a), а не настоящая TD-ошибка. Упомянем некоторые репрезентативные публикации из обширного массива литературы по модулированной вознаграждением STDP: Wickens (1990), Reynolds and Wickens (2002 и Calabresi, Picconi, Tozzi and Di Filippo (2007). В работе Pawlak and Kerr (2008) показано, что дофамин необходим для индуцирования STDP в кортико-стриарных синапсах средних шипиковых нейронов. См. также Pawlak, Wickens, Kirkwood, and Kerr (2010). В работе Yagishita, Hayashi-Takagi, Ellis-Davies, Urakubo, Ishii, and Kasai (2014) установлено, что дофамин вызывает увеличение шипика средних шипиковых нейронов у мышей только в течение временного окна длительностью от 0.3 до 2 секунд после стимуляции STDP. В работе Izhikevich (2007) предложена и исследована идея об использовании временных условий STDP для активизации контингентных следов приемлемости. В работе Frémaux, Sprekeler, and Gerstner (2010) предложены теоретические условия успешного обучения по правилам, которые основаны на модулированной вознаграждением STDP.

- 15.9** Гипотеза Клопфа о гедонистическом нейроне (Klopf 1972, 1982) подсказала нам реализацию алгоритма исполнитель–критик в виде ИНС с одним нейроноподобным блоком исполнителя и правилом обучения, похожим на закон эффекта (Barto, Sutton, and Anderson, 1983). Идеи, связанные с синаптически локальной приемлемостью Клопфа, предлагались и другими авторами. В работе Crow (1968) высказано предположение, что изменения в синапсах кортикальных нейронов чувствительны к последствиям нейронной активности. Подчеркивая необходимость учитывать временную задержку между нейронной активностью и ее последствиями в модулированной вознаграждением форме синаптической пластичности, Кроу предложил контингентную форму приемлемости, ассоциированную, однако, с целым нейроном, а не его отдельными синапсами. Согласно его гипотезе, волна нейронной активности

приводит к краткосрочному изменению в клетках, участвующих в этой волне, отличающему их от фенотипа клеток, не подвергшихся такой активации... такие клетки в результате этого краткосрочного изменения становятся чувствительны к сигналу вознаграждения...

таким образом, если подобный сигнал поступает до окончания периода затухания изменения, синаптические связи между клетками становятся более эффективными (Crow, 1968).

Кроу возражал против более ранних предположений о том, что такую роль играют реверберирующие нейронные структуры, указывая, что воздействие сигнала вознаграждения на подобную структуру «...приводит к образованию синаптических связей, которые ведут к реверберации (т. е. участвовали в активности в момент сигнала вознаграждения), а не связей на тракте, который привел к адаптивному моторному выходу». Далее Кроу постулировал, что сигналы вознаграждения доставляются «отдельной системой нервных волокон», предположительно той, которую пунктировали Olds and Milner (1954) и которая преобразует синаптические связи «из краткосрочной формы в долгосрочную».

В еще одной прозорливой гипотезе Миллер предложил правило обучения по типу закона эффекта, включающее синаптически локальные контингентные следы приемлемости:

...можно представить себе, что в конкретной сенсорной ситуации нейрон В случайно запускает «значимый всплеск» активности, который затем транслируется в моторные акты, изменяющие ситуацию. Следует предположить, что этот значимый всплеск на уровне нейронов влияет на все синапсы нейрона, которые были активны в тот момент... тем самым производя предварительный выбор синапсов, подлежащих усилению, но еще не производя фактического усиления. ... Усиливающий сигнал ... производит окончательный выбор ... и фиксирует конечное изменение соответствующих синапсов (Miller, 1981, p. 81).

Гипотеза Миллера также включала механизм, подобный критику, который он называл «блоком сенсорного анализатора» и который работал в соответствии с принципами классического обусловливания, подавая сигналы подкрепления нейронам, так чтобы они обучались переходить из состояния низкой ценности в состояния более высокой ценности. Тем самым он предвосхитил использование TD-ошибки в качестве сигнала подкрепления в архитектуре исполнитель–критик. Идея Миллера не только сходна с идеей Клопфа (за исключением явного инициирования специального «усиливающего сигнала»), но и предвосхитила общие черты модулированной вознаграждением STDP.

Сюда же относится родственная, хотя и другая, идея, которая в работе Seung (2003) названа «гедонистическим синапсом». Заключается она в том, что синапсы индивидуально корректируют вероятность высвобождения нейромедиатора в духе закона эффекта: если за высвобождением следует вознаграждение, то вероятность высвобождения увеличивается, а если вознаграждается отсутствие высвобождения, то уменьшается. По существу, это та же схема обучения, которую Мински использовал в своей докторской диссертации 1954 года, защищенной в Принстонском университете, где подобный синапсу самообучающийся элемент был назван SNARC (Stochastic Neural-Analog Reinforcement Calculator – стохастический ней-

ро-аналоговый калькулятор с подкреплением). Контингентные следы приемлемости тоже встречались в его идеях, хотя речь шла о контингентности активности индивидуального синапса, а не постсинаптического нейрона. К той же тематике примыкает предложение Unnikrishnan and Venugopal (1994), в котором для корректировки весов ИНС используется корреляционный метод из работы Harth and Tzanakou (1974).

В работе Frey and Morris (1997) предложена идея «синаптической метки» для индуцирования длительного усиления синаптической силы. Эта идея не лишена сходства с приемлемостью Клопфа, но авторы предполагали, что метка состоит из временного усиления синапса, которое может быть преобразовано в длительное усиление последующей активацией нейрона. В модели, описанной в работах O'Reilly and Frank (2006) и O'Reilly, Frank, Nazy, and Watz (2007), используется рабочая память для связывания интервалов времени, и нет следов приемлемости. В работе Wickens and Kotter (1995) обсуждаются возможные механизмы синаптической приемлемости. В статье He, Huertas, Hong, Tie, Hell, Shouval, Kirkwood (2015) приведены экспериментальные факты в поддержку существования контингентных следов приемлемости в синапсах кортикальных нейронов, имеющих примерно такую динамику, как у следов приемлемости, постулированных Клопфом.

Метафора применяющего правило обучения нейрона, связанная с бактериальным хемотаксисом, обсуждалась в работе Barto (1989). Подробное исследование бактериального хемотаксиса, предпринятое Кошландом, отчасти было мотивировано сходством между некоторыми особенностями бактерий и нейронов (Koshland, 1980). См. также Berg (1975). В работе Shmansky (2009) предложено синаптическое правило обучения, напоминающее вышеупомянутое правило Сонга, в котором каждый синапс действует сам по себе как хемотаксическая бактерия. В этом случае совокупность синапсов «плывет» в сторону аттрактантов в многомерном пространстве значений синаптических весов. В работе Montague, Dayan, Person, and Sejnowski (1995) предложена устроенная по аналогии с хемотаксисом модель поведения пчелы при поиске пищи, в которой нейромодулятором является октопамин.

- 15.10 Исследования поведения агентов обучения с подкреплением в командных и игровых задачах имеют долгую историю, в которой можно выделить три этапа. Насколько нам известно, первый этап начался исследованиями советского математика и физика М. Л. Цетлина. Собрание его работ было опубликовано в виде сборника Tsetlin (1973) после его смерти в 1966 году. В разделах 1.7 и 4.8 нашей книги упоминаются его работы по самообучающимся автоматам в связи с задачами о бандите. Наследие Цетлина включает также исследования самообучающихся автоматов в командных и игровых задачах, которые впоследствии привели к работам по применению стохастических самообучающихся автоматов Narendra and Thathachar (1974, 1989), Viswanathan and Narendra (1974), Lakshmivarahan and Narendra (1982), Narendra and Wheeler (1983) и Thathachar and Sastry (2002). В работе Thathachar and Sastry (2011) приведен сравнительно недавний и более

полный перечень. Эти исследования в большинстве своем ограничивались неассоциативными самообучающимися автоматами, т. е. ассоциативные, или контекстуальные, задачи о бандитах (раздел 2.9) в них не рассматривались.

Второй этап начался с обобщения самообучающихся автоматов на ассоциативный, или контекстуальный, случай. В работах Barto, Sutton, and Brouwer (1981) и Barto and Sutton (1981b) описаны эксперименты с ассоциативными стохастическими самообучающимися автоматами в однослойных ИНС, которым транслировался глобальный сигнал подкрепления. В качестве алгоритма обучения использовалось ассоциативное обобщение алгоритма Аlorex, предложенного в работе Harth and Tzanakou (1974). В работе Барто с соавторами нейроноподобные элементы, реализующие этот вид обучения, были названы *элементами ассоциативного поиска* (ЭАС). В работе Barto and Anandan (1985) описан ассоциативный алгоритм обучения с подкреплением, названный алгоритмом *ассоциативного вознаграждения–наказания* (A_{R-P}). При доказательстве сходимости использовалось сочетание теории стохастических самообучающихся автоматов с теорией классификации образов. В работах Barto (1985, 1986) и Barto and Jordan (1987) описаны результаты для команд A_{R-P} -блоков, соединенных в многоуровневые ИНС, которые показали их способность к обучению нелинейным функциям, в частности XOR, с помощью глобально транслируемого сигнала подкрепления. В работе Barto (1985) приводится подробное обсуждение этого подхода к ИНС и связей между правилами обучения этого типа и другими встречавшимися в тогдашней литературе. В работе Williams (1992) проведен математический анализ и расширение этого класса правил обучения, а их применение соотнесено с методом обратного распространения ошибки для обучения многослойных ИНС. В работе Williams (1988) описано несколько способов совместного использования методов обратного распространения и обучения с подкреплением для обучения ИНС. В работе Williams (1992) показано, что алгоритм REINFORCE является частным случаем алгоритма A_{R-P} , хотя общий алгоритм дает лучшие результаты (Barto, 1985).

Третий этап интереса к командам агентов обучения с подкреплением был связан с возросшим пониманием роли дофамина как нейромодулятора с широким диапазоном трансляции и размышлениями о существовании модулированной вознаграждением STDP. В гораздо большей степени, чем прежде, исследования на этом этапе касаются деталей синаптической пластичности и других ограничений, пришедших из нейронаук. Из публикаций отметим следующие (в хронологическом и алфавитном порядке): Bartlett and Baxter (1999, 2000), Xie and Seung (2004), Baras and Meir (2007), Farries and Fairhall (2007), Florian (2007), Izhikevich (2007), Pecovski, Maass, and Legenstein (2008), Legenstein, Pecovski, and Maass (2008), Kolodziejcki, Porr, and Wörgötter (2009), Urbanczik and Senn (2009) и Vasilaki, Frémaux, Urbanczik, Senn, and Gerstner (2009). В работе Nowé, Vrancx, and De Hauwere (2012) приведен обзор других недавних достижений в более широкой области многоагентного обучения с подкреплением.

- 15.11** В работе Yin and Knowlton (2006) приведен обзор результатов экспериментов по снижению ценности желаемого исхода у грызунов. Они подтверждают мнение о том, что привычное и целеустремленное поведения (в том смысле, в котором эту фразу употребляют психологи) в большей степени ассоциированы соответственно с обработкой в дорсолатеральном стриатуме (ДЛС) и в дорсомедиальном стриатуме (ДМС). Результаты описанных в работе Valentin, Dickinson, and O'Doherty (2007) экспериментов с людьми по функциональной нейровизуализации, направленных на снижение ценности желаемого исхода, позволяют предположить, что орбитофронтальная кора (ОФК) – важный компонент целенаправленного выбора. Регистрация отдельных нейронов у обезьян в работе Padoa-Schioppa and Assad (2006) подтверждает роль ОФК в кодировании ценностей, направляющем выбор поведения. В работах Rangel, Camerer, and Montague (2008) и Rangel and Hare (2010) с точки зрения нейроэкономики обсуждаются открытия, касающиеся принятия мозгом целенаправленных решений. В работе Pezzulo, van der Meer, Lansink, and Pennartz (2014) приведен обзор нейронаучных результатов о внутренне генерируемых последовательностях и представлена модель, показывающая, как эти механизмы могут являться компонентами планирования, основанного на модели. В работе Daw and Shohamy (2008) высказано предположение, что дофаминовая сигнализация хорошо стыкуется с привычным, или безмодельным, поведением, но в целеустремленное, или основанное на модели, поведение вовлечены другие процессы. Данные экспериментов Bromberg-Martin, Matsumoto, Hong, and Nikosaka (2010) показывают, что дофаминовые сигналы содержат информацию, относящуюся как к привычному, так и к целеустремленному поведению. В работе Doll, Simon, and Daw (2012) доказывается, что в мозге может и не быть четкого разделения между механизмами, содействующими привычному и целеустремленному поведению и выбору.
- 15.12** В обзоре Keiin and Janak (2015) рассматриваются связи между TD-ошибками и наркозависимостью. В работе Nutt, Lingford-Hughes, Erritzoe, and Stokes (2015) подвергнута критическому анализу гипотеза о том, что наркозависимость обусловлена нарушениями в работе дофаминовой системы. В статье Montague, Dolan, Friston, and Dayan (2012) в общих чертах описаны цели и ранние работы в области вычислительной психиатрии, а в работе Adams, Huys, and Roiser (2015) – более поздние достижения.

Глава 16

Примеры и приложения

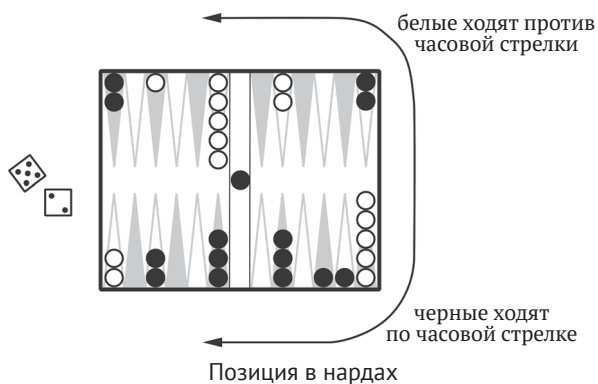
В этой главе мы расскажем о нескольких конкретных применениях обучения с подкреплением. Некоторые из них являются значительными приложениями, имеющими потенциальную ценность для экономики. Один, программа игры в шашки Сэмюэла, представляет в основном исторический интерес. Мы построили изложение, так чтобы проиллюстрировать некоторые компромиссы и проблемы, возникающие в реальных приложениях. Например, мы показываем, как знания о предметной области включаются в постановку и решение задачи. Мы также обращаем внимание на вопросы представления, которые часто оказываются решающими для успеха приложения. Алгоритмы, использованные в некоторых примерах, существенно сложнее, чем описано в основном тексте книги. Применения обучения с подкреплением пока еще далеки от рутины и обычно требуют искусства в не меньшей степени, чем науки. Упростить создание приложений – одна из целей современных исследований по обучению с подкреплением.

16.1. TD-GAMMON

На сегодня одно из самых впечатляющих применений обучения с подкреплением – программа игры в нарды Джеральда Тезауро (Tesauro, 1992, 1994, 1995, 2002). Эта программа, *TD-Gammon*, почти ничего не знала о нардах, но обучилась играть удивительно хорошо, на уровне сильнейших в мире гроссмейстеров. Алгоритм обучения в TD-Gammon устроен как сочетание алгоритма TD(λ) с нелинейной аппроксимацией функций многослойной искусственной нейронной сетью (ИНС), обученной методом обратного распространения TD-ошибок.

Нарды – одна из главных игр в том смысле, что в нее играют по всему миру, проводятся многочисленные турниры и регулярные матчи за звание чемпиона мира. Это частично азартная игра, на исход которой ставят большие деньги. Профессиональных игроков в нарды, наверное, больше, чем в шахматы. Игра ведется 15 белыми и 15 черными шашками на доске с 24 полями, которые называются *пунктами*. На следующей странице показана типичная позиция, возникающая в начале игры, изображенная с точки зрения белых. Белые только что бросили кости, на которых выпало 5 и 2. Это означает, что игрок может подвинуть одну из своих шашек на 5 шагов, а еще одну (быть может, ту же самую) на 2 шага. Например, можно было бы подвинуть две шашки, находящиеся в пункте 12: одну в пункт 17, а другую – в пункт 14. Цель белых – переместить все свои шашки в последний квадрант (пункты 19–24), а затем снять их с доски. Игрок, который первым снимет все свои

шашки, выигрывает. Сложность в том, что шашки взаимодействуют, когда минуют друг друга, двигаясь в противоположных направлениях. Например, если бы в этой позиции был ход черных, то они могли бы воспользоваться выпавшей на кости двойкой, чтобы передвинуть шашку из пункта 24 в пункт 22, «побив» находящуюся там белую шашку. Сбитые шашки помещаются на *бар* – вертикальную полосу в центре доски (мы видим там одну сбитую черную шашку), откуда они вновь вступают в игру из начальной позиции. Однако если в пункте находится две шашки, то противник не может пойти в этот пункт – шашки защищены от удара. Таким образом, белые не могут использовать выпавшую комбинацию 5-2, чтобы поставить какую-нибудь из своих шашек в пункт 3, потому что он занят группой черной шашек. Создание сплошных участков занятых пунктов с целью заблокировать ходы противника – одна из простейших стратегий в игре.



В нардах есть еще несколько правил, усложняющих игру, но основная идея ясна. При наличии 30 шашек и 24 мест их размещения (26, если считать бар и место вне доски) понятно, что количество возможных позиций огромно, гораздо больше, чем элементов памяти в любом физически реализуемом компьютере. Количество ходов в каждой позиции тоже велико. Для типичного броска костей может быть 20 способов продолжить игру. При рассмотрении будущих ходов, например ответа противника, следует также учитывать возможный результат бросания костей. В итоге эффективный коэффициент ветвления дерева игры равен примерно 400. Это слишком много, чтобы можно было использовать традиционные методы эвристического поиска, которые оказались так эффективны в играх типа шахмат и шашек.

С другой стороны, нарды вполне под стать возможностям TD-обучения. Хотя игра в высшей степени стохастическая, полное описание ее состояния доступно в любой момент времени. Игра проходит через последовательность ходов и позиций, пока не заканчивается победой одного или другого игрока. Исход можно интерпретировать как окончательное вознаграждение, подлежащее предсказанию. Но применить к этой задаче описанные выше теоретические результаты не удастся. Количество состояний настолько велико, что таблица соответствия не поможет, а противник является источником неопределенности и изменений во времени.

В программе TD-Gammon используется нелинейная форма TD(λ). Оценкой ценности $\hat{V}(s, w)$ любого состояния (позиции на доске) s является оценка вероятности

выигрыша при старте из состояния s . Для этого мы начисляем нулевое вознаграждение на любом временном шаге, кроме приведших к окончанию игры. Чтобы реализовать такую функцию ценности, в TD-Gammon используется многослойная ИНС типа той, что изображена на рисунке ниже. (В настоящей сети есть два дополнительных блока в последнем слое для оценки вероятности выигрыша каждым игроком при игре специальным способом, который называется «gammon» или «backgammon».) Сеть состоит из слоя входных блоков, слоя скрытых блоков и одного выходного блока. На вход сети подается представление позиции, а на выходе получается оценка ценности этой позиции.

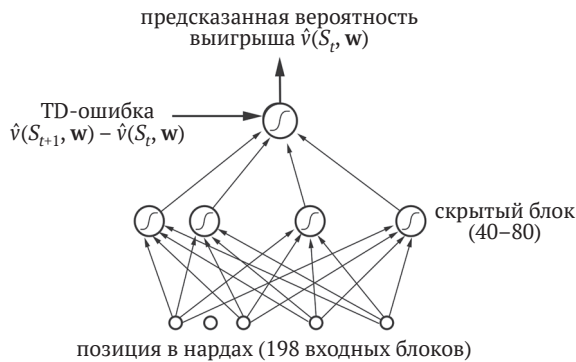


Рис. 16.1 ♠ ИНС в программе TD-Gammon

В первой версии программы, TD-Gammon 0.0, позиции передавались сети сравнительно простым способом, предполагавшим очень мало знаний о специфике игры в нарды. Однако он подразумевал хорошее знание о работе ИНС и о том, как лучше представлять информацию для них. Точное представление, выбранное Тезауро, очень поучительно. Всего в сети было 198 входных блоков. Для каждого пункта на доске четыре блока указывали количество белых шашек в нем. Если белых шашек не было, что все четыре блока принимали нулевое значение. Если была одна шашка, то первый блок принимал значение 1. Так кодировалась концепция «слабого места позиции», т. е. шашки, которую мог сбить противник. Если шашек было две или более, то второй блок устанавливался равным 1. Так кодировалась концепция «занятого пункта», в который противник не мог поставить свою шашку. Если в пункте находилось ровно три шашки, то третий блок устанавливался равным 1. Так кодировалась концепция «одной запасной», т. е. дополнительной, шашки вдобавок к тем двум, что уже находились в пункте. Наконец, если шашек было больше трех, то четвертому блоку присваивалось значение, пропорциональное количеству шашек сверх трех. Если обозначить n общее число шашек в пункте и $n > 3$, то четвертый блок принимает значение $(n - 3)/2$. Так кодировалось линейное представление «нескольких запасных» в данном пункте.

Четыре блока для белых и четыре блока для черных в каждом из 24 пунктов дают 192 блока. Еще два блока кодировали количество белых и черных шашек на баре (каждый принимает значение $n/2$, где n – количество шашек соответствующего цвета на баре), а еще два – количество черных и белых шашек, уже снятых с доски (они принимают значение $n/15$, где n – количество снятых шашек соот-

ветствующего цвета). Наконец, два блока служили для бинарного кодирования очередного хода: белых или черных. Логика, стоящая за таким выбором, ясна. Тезауро стремился представить позицию в лоб, так чтобы не слишком увеличивать число блоков. Он зарезервировал по одному блоку для каждой концептуально различной возможности, которая могла иметь отношение к делу, и привел их значения примерно к общему диапазону, в данном случае от 0 до 1.

Зная представление игровой позиции, сеть стандартным способом вычисляла оценку ее ценности. Каждой связи входного блока со скрытым назначался вещественный вес. Сигналы от каждого входного блока умножались на соответствующие веса и суммировались в скрытом блоке. Выходом $h(j)$ скрытого блока j была нелинейная сигмоидная функция от взвешенной суммы:

$$h(j) = \sigma\left(\sum_i w_{ij}x_i\right) = \frac{1}{1 + e^{-\sum_i w_{ij}x_i}},$$

где x_i – значение i -го входного блока, а w_{ij} – вес его связи с j -м выходным блоком (все веса, присутствующие в сети, образуют вектор параметров $\mathbf{w}$). Выход сигмоиды всегда является числом от 0 до 1 и естественно интерпретируется как вероятность, основанная на суммировании фактов. Вычисление вклада скрытых блоков в выходной аналогично. Связи между каждым скрытым блоком и выходным назначаются вес. Выходной блок вычисляет взвешенную сумму, которая передается такой же сигмоидной нелинейности.

В TD-Gammon использовалась полугradientная форма алгоритма $TD(\lambda)$, описанная в разделе 12.2; градиенты вычислялись алгоритмом обратного распространения ошибки (Rumelhart, Hinton, and Williams, 1986). Напомним, что общее правило обновления в этом случае имеет вид:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha[R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t)]\mathbf{z}_t, \quad (16.1)$$

где $\mathbf{w}_t$ – вектор всех модифицируемых параметров (в данном случае весов сети), а $\mathbf{z}_t$ – вектор следов приемлемости, по одному для каждого элемента $\mathbf{w}_t$, обновляемый по правилу

$$\mathbf{z}_t \doteq \gamma \lambda \mathbf{z}_{t-1} + \nabla \hat{v}(S_t, \mathbf{w}_t),$$

где $\mathbf{z}_0 \doteq 0$. Градиент в этом уравнении можно эффективно вычислить с помощью процедуры обратного распространения. В применении к нардам, где $\gamma = 1$, а вознаграждение равно нулю во всех случаях, кроме выигрыша, TD-ошибка в правиле обучения обычно равна просто $\hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})$, как показано на рис. 16.1.

Чтобы применить правило обучения, нам нужен источник партий в нарды. Тезауро получил бесконечную последовательность партий, заставив своего обучающегося игрока играть против себя же. При выборе ходов TD-Gammon рассматривала каждый из примерно 20 вариантов розыгрыша выпавшей на костях комбинации и получающиеся в результате позиции. Эти позиции – не что иное, как *последостояния*, обсуждавшиеся в разделе 6.8. У сети запрашивалась оценка их ценностей. Затем выбирался ход, который вел в позицию с наивысшей оценкой. Действуя таким образом, т. е. заставляя TD-Gammon совершать ходы за обе стороны, можно было легко сгенерировать много партий в нарды. Каждая партия

рассматривалась как эпизод, в котором последовательность позиций играла роль состояний $S_0, S_1, S_2, \dots$. Тезауро применял нелинейное правило TD (16.1) строго инкрементно, т. е. после каждого хода.

Весы сети инициализировались небольшими случайно выбранными значениями. Таким образом, начальные оценки были совершенно произвольными. Поскольку ходы выбирались на основе этих оценок, первые ходы неизбежно были плохими, а первые партии продолжались на протяжении сотен, а то и тысяч ходов, пока какая-то сторона не выигрывала, почти случайно. Но после нескольких десятков сыгранных партий качество игры начало быстро улучшаться.

Сыграв примерно 300 000 партий сама с собой, TD-Gammon 0.0 научилась играть примерно на уровне лучших из существовавших тогда программ игры в нарды. Этот результат был поразителен, потому что во все предыдущие программы были встроены обширные знания об игре. Например, чемпионом в то время была программа Neurogammon, тоже написанная Тезауро, в которой использовалась ИНС, но не было TD-обучения. Сеть Neurogammon была обучена на большом корпусе примеров ходов, подготовленном опытными игроками в нарды, а кроме того, работала со множеством признаков, специально сконструированных для этой игры. Neurogammon была высокооптимизированной и весьма эффективной программой, которая уверенно выиграла Всемирную олимпиаду по нардам в 1989 году. С другой стороны, у TD-Gammon 0.0 не было практически никаких знаний о нардах. То, что она смогла сражаться наравне с Neurogammon и всеми другими подходами, убедительно свидетельствует о потенциале методов обучения путем игры против себя же.

Турнирный успех программы TD-Gammon 0.0, обладающей нулевыми экспертными знаниями о нардах, натолкнул на мысль об очевидной модификации: добавить специализированные для нард признаки, сохранив метод TD-обучения на играх с собой. В результате родилась версия TD-Gammon 1.0, которая играла значительно лучше всех предшествующих программ, а серьезную конкуренцию ей могли составить только люди. Последующие версии, TD-Gammon 2.0 (40 скрытых блоков) и TD-Gammon 2.1 (80 скрытых блоков), были дополнены избирательной двухэтапной процедурой поиска. При выборе хода программа анализировала не только возникающие после него позиции, но рассматривала также возможные броски и ходы противника. В предположении, что противник всегда делает ход, который представляется наилучшим в данной позиции без заглядывания вперед, программа вычисляла ожидаемую ценность каждого потенциального хода и выбирала наилучший. Для экономии машинного времени второй этап поиска выполнялся только для тех потенциальных ходов, которые получили высокую оценку после первого этапа; в среднем это составляло четыре-пять ходов. Двухэтапный поиск влиял только на выбор ходов, процесс обучения происходил так же, как раньше. В последних версиях программы, TD-Gammon 3.0 и 3.1, было 160 скрытых блоков и избирательный трехэтапный поиск. TD-Gammon иллюстрирует сочетание обученных функций ценности, поиска на этапе принятия решений, как в эвристическом поиске, и поиска по дереву методом Монте-Карло. В последующей работе Tesauro and Galperin (1997) были исследованы методы траекторной выборки как альтернативы поиску на всю ширину, что кратно (в 4–6 раз) уменьшало частоту ошибок в игре при сохранении приемлемого времени обдумывания хода – порядка 5–10 секунд.

На протяжении 1990-х годов программы Тезауро сыграли очень много партий с соперниками – людьми мирового уровня. Результаты приведены в табл. 16.1. Согласно этим результатам и анализу, проведенному гроссмейстерами игры в нарды (Robertie, 1992; Tesauro, 1995), версия TD-Gammon 3.0 играла на уровне близком, а быть может, превышающем силу лучших мировых игроков. В статье (Tesauro, 2002) Тезауро опубликовал развернутый анализ решений о выборе ходов и удвоении ставок, принимаемых TD-Gammon в сравнении с ведущими игроками-людьми. Он пришел к выводу, что TD-Gammon 3.1 имела «несомненное преимущество» в решениях о перемещении шашки и «небольшой перевес» в решениях об удвоении.

Таблица 16.1. Итоговые результаты TD-Gammon

Программа	Скрытых блоков	Тренировочных игр	Противники	Результаты
TD-Gammon 0.0	40	300 000	Другие программы	Одна из лучших
TD-Gammon 1.0	80	300 000	Роберти, Магриэль, ...	–13 очков / 51 игра
TD-Gammon 2.0	40	800 000	Различные гроссмейстеры	–7 очков / 38 игр
TD-Gammon 2.1	80	1 500 000	Роберти	–1 очко / 20 игр
TD-Gammon 3.0	80	1 500 000	Казарос	+6 очков / 20 игр

TD-Gammon оказала большое влияние на манеру игры ведущих игроков-людей. Например, в некоторых начальных позициях она обучилась играть иначе, чем было принято у людей. Глядя на успех TD-Gammon и проведя анализ, лучшие игроки теперь разыгрывают эти позиции так, как TD-Gammon (Tesauro, 1995). Это влияние на игру людей заметно ускорилося, после того как стали широко доступны другие самообучающиеся программы игры в нарды на основе ИНС, вдохновленные примером TD-Gammon, например Jellyfish, Snowie и GNUBackgammon. Эти программы способствовали распространению новых знаний, сгенерированных ИНС, что привело к качественному повышению уровня игры на турнирах (Tesauro, 2002).

16.2. ПРОГРАММЫ ИГРЫ В ШАШКИ СЭМЮЭЛА

Важным предтечей программы TD-Gammon Тезауро была пионерская работа Артура Сэмюэла (Samuel, 1959, 1967) по конструированию программ, обучающихся игре в шашки. Сэмюэл одним из первых начал эффективно использовать методы эвристического поиска и то, что мы сейчас называем обучением на основе временных различий. Его шашечные программы представляют не только исторический интерес, но и поучительный пример. Мы всячески подчеркиваем связь методов Сэмюэла с современными методами обучения с подкреплением и пытаемся объяснить некоторые мотивы Сэмюэла.

Первую шашечную программу Сэмюэл написал для IBM 701 в 1952 году. Первая его самообучающаяся программа была закончена в 1955 году и показана по телевизору в 1956 году. Более поздние версии достигли хорошего, но не экспертного уровня игры. Игры привлекли внимание Сэмюэла как область, где можно

было изучать машинное обучение, поскольку игры не так сложны, как проблемы «из жизни», но открывают плодотворное поле деятельности для исследования совместного использования эвристических процедур и обучения. Он выбрал шашки, а не шахматы из-за относительной простоты, позволяющей в большей мере сосредоточиться на обучении.

Программы Сэмюэла были основаны на просмотре на несколько ходов вперед из каждой текущей позиции. В них использовалось то, что мы сейчас называем методами эвристического поиска, для решения о том, когда расширять дерево поиска, а когда остановиться. Для оценивания конечных позиций при поиске использовалась функция ценности, или «оценочный полином», и линейная аппроксимация. В этом и других отношениях на работу Сэмюэла, вероятно, повлияли предложения Шеннона (Shannon, 1950). В частности, программа Сэмюэла основывалась на применении минимаксной процедуры Шеннона для нахождения лучшего хода в текущей позиции. Обратный проход по дереву поиска из конечных позиций давал оценку позиции при наилучшем ходе в предположении, что машина всегда пытается максимизировать оценку, а противник – минимизировать ее. Сэмюэл называл это «возвратной оценкой» позиции. Достигнув корня дерева поиска – текущей позиции, – минимаксная процедура давала наилучший ход в предположении, что противник будет использовать такой же критерий оценивания, конечно, со своей точки зрения. В некоторых версиях программ Сэмюэла применялись более сложные методы управления поиском, аналогичные тем, что сейчас известны как альфа-бета отсечение (см., например, Pearl, 1984).

Сэмюэл использовал два основных метода обучения, самый простой из которых он называл *зубрежкой*. При этом сохранялось описание каждой позиции, возникавшей на доске в ходе партии, и ее возвратная оценка, вычисленная минимаксной процедурой. В результате, если уже встречавшаяся позиция вновь возникала в качестве конечной позиции дерева поиска, то глубина поиска де-факто увеличивалась, поскольку в ценности этой позиции были кешированы результаты одного или нескольких поисков, выполненных ранее. Поначалу возникла проблема – у программы не было стимулов двигаться по самому прямому пути к победе. Сэмюэл наделил ее «чувством направления», немного уменьшая ценность позиции всякий раз, как она получала очередное значение (называемое уровнем) в процессе минимаксного анализа. «Если теперь программа столкнется с выбором позиций, оценки которых отличаются только номером уровня, то она будет автоматически принимать самое выгодное решение, выбирая низкоуровневый вариант, если выигрывает, и высокоуровневый, если проигрывает» (Samuel, 1959, p. 80). Сэмюэл обнаружил, что эта техника, напоминающая обесценивание, существенна для успешного обучения. Метод зубрежки давал медленное, но устойчивое улучшение игры, наиболее эффективное в дебюте и в эндшпиле. Его программа стала играть «лучше среднего начинающего» после обучения на большом количестве игр против самой себя, различных соперников-людей и на партиях, взятых из книг, в режиме обучения с учителем.

Обучение методом зубрежки и другие аспекты работы Сэмюэла являются сильным аргументом в пользу основной идеи обучения на основе временных различий – ценность состояния должна быть равна ценности вероятных следующих состояний. Ближе всего Сэмюэл подошел к этой идее в своем втором методе обучения, процедуре «обучения посредством обобщения», целью которой была

корректировка параметров функции ценности. Метод Сэмюэла концептуально совпадал с тем, что много позже использовал Тезауро в программе TD-Gammon. Программа играла много игр против другой своей версии и выполняла обновление после каждого хода. Идея обновления Сэмюэла показана на диаграмме обновления на рис. 16.2. Каждый белый кружок представляет позицию, в которую программа делает следующий ход, так называемую позицию *после хода*, а каждый черный кружок – позицию, в которую ходит противник. После хода обеих сторон производится обновление ценности каждой позиции, что приводит ко второй позиции после хода. Обновление производится в направлении минимаксного значения поиска, запущенного из второй позиции после хода. Таким образом, общий эффект такой же, как при возврате на один полный ход в реальной игре и последующем поиске среди возможных событий, как показано на рис. 16.2. В действительности алгоритм Сэмюэла был значительно сложнее по чисто вычислительным причинам, но основную идею мы изложили правильно.

Сэмюэл не включил явного вознаграждения. Вместо этого он фиксировал вес самого важного признака, *преимущества в шашках*, который сравнивал количество шашек у программы и у противника, придавая более высокий вес дамкам и учитывая, что размен шашками выгоднее, когда выигрываешь, чем когда проигрываешь. Таким образом, целью программы Сэмюэла было улучшение преимущества в шашках, от которого в этой игре сильно зависит выигрыш.

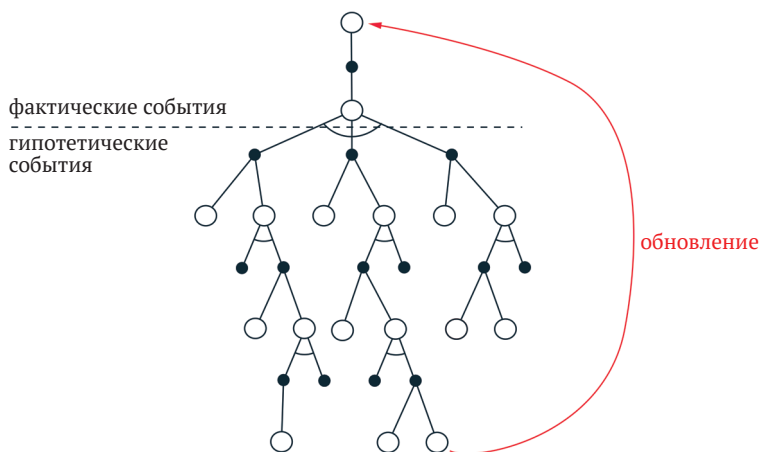


Рис. 16.2 ❖ Диаграмма предшествующих состояний для шашечной программы Сэмюэла

Однако методу обучения Сэмюэла не хватало существенной части истинного алгоритма на основе временных различий. TD-обучение можно рассматривать как способ добиться согласованности функции ценности с самой собой, и это отчетливо видно в методе Сэмюэла. Но, кроме того, необходимо увязать функцию ценности с истинными ценностями состояний. Мы достигали этого с помощью вознаграждений и обесценивания или придавая фиксированную ценность заключительному состоянию. Но в методе Сэмюэла нет вознаграждений, и заключительные позиции игры не обрабатываются как-то особенно. Как отмечал сам

Сэмюэл, его функция ценности могла бы стать согласованной, если просто присвоить всем позициям постоянную ценность. Он надеялся избежать таких решений, придав члену преимущества в шашках большой неизменяемый вес. Но хотя это уменьшало вероятность нахождения бесполезных оценочных функций, полностью исключить их таким образом невозможно. Например, получить постоянную функцию можно, задав изменяемые веса так, что в сумме они сведут на нет влияние неизменяемого.

Так как в процедуре обучения Сэмюэла не было ограничений, разрешающих находить только полезные оценочные функции, оставалась возможность ее ухудшения по мере накопления опыта. И действительно, Сэмюэл наблюдал это в течение продолжительной серии тренировочных партий против самой себя. Чтобы заставить программу снова играть лучше, Сэмюэлу приходилось вмешиваться и сбрасывать в ноль вес с наибольшей абсолютной величиной. По его мнению, такое радикальное вмешательство уводило программу от локального оптимума, но, возможно, на самом деле оно уводило программу от оценочных функций, которые хотя и были согласованными, но никак не влияли на выигрыш или проигрыш.

Несмотря на все эти потенциальные проблемы, шашечная программа Сэмюэла, в которой использовался метод обучения на основе обобщения, достигла уровня «выше среднего». Хорошие игроки-любители характеризовали ее как «искусную, но победимую» (Samuel, 1959). В отличие от версии с зубрежкой, эта версия хорошо играла в миттельшпиле, но оставалась слабой в дебюте и в эндшпиле. В этой программе была также заложена возможность поиска признаков, наиболее полезных для формирования функции ценности. Более поздняя версия (Samuel, 1967) получила улучшенную процедуру поиска, включающую альфа-бета отсечение, широкое применение режима обучения с учителем, названного «обучением по книге», и иерархические таблицы соответствия, названные сигнатурными таблицами (Griffith, 1966), для представления функции ценности с помощью таблицы, а не линейной аппроксимации. Эта версия обучилась играть гораздо лучше программы 1959 года, хотя уровня мастера все же не достигла. Программа игры в шашки Сэмюэла была признана значительным достижением в области искусственного интеллекта и машинного обучения.

16.3. СТРАТЕГИЯ ВЫБОРА СТАВКИ В ПРОГРАММЕ WATSON

IBM Watson¹ – разработанная группой исследователей из компании IBM система для игры в популярной телевикторине *Jeopardy!*². Она снискала славу, завоевав первое место в показательном соревновании с чемпионами-людьми. Хотя главным техническим достижением Watson было умение быстро и правильно отвечать на вопросы, заданные на естественном языке и касавшиеся самых разных областей знания, для победы в *Jeopardy!* немалую важность сыграли изощренные стратегии принятия решений в критические моменты игры. Авторы работы Tesauro, Gondek, Lechner, Fan, and Prager (2012, 2013) адаптировали описанную

¹ Зарегистрированная торговая марка корпорации IBM.

² Зарегистрированная торговая марка Jeopardy Productions Inc. Российский аналог называется «Своя игра».

выше программу Тезауро TD-Gammon к стратегии, используемой Watson для выбора ставки при ответе на вопрос-аукцион (Daily-Double – DD). Авторы сообщают, что эффективность стратегии выбора ставки оказалась намного выше, чем у игроков во время шоу, и что наряду с другими передовыми стратегиями это стало важным вкладом во впечатляющую победу Watson. Здесь мы ограничимся только DD-ставками, поскольку именно эта часть Watson больше всего обязана обучению с подкреплением.

В *Jeopardy!* играют три участника. Перед ними находится табло с 30 полями, каждое из которых скрывает вопрос и имеет денежную ценность. На табло имеется шесть столбцов, каждому из которых соответствует некоторая категория. Участник выбирает поле, ведущий читает соответствующий ему вопрос, и каждый участник может ответить на него, нажав кнопку. Первый, кто нажал кнопку, получает право ответить. Если ответ правильный, его банк увеличивается на сумму, указанную в поле. Если же ответ неверен или если участник не ответил в течение пяти секунд, то его банк уменьшается на эту сумму, а остальные участники получают шанс ответить на тот же вопрос. В одном или двух полях (в зависимости от номера раунда) находятся специальные вопросы-аукционы. Выбравший их участник получает возможность ответить на вопрос и должен решить – еще до того как вопрос прозвучал, – сколько он хочет поставить. Ставка должна быть больше пяти долларов, но не больше суммы в банке участника. Если участник ответит правильно, то его банк увеличивается на сумму ставки, иначе уменьшается на ту же сумму. Игра заканчивается «финальным раундом», в котором каждый участник делает ставку, не оглашая ее, а затем записывает ответ на зачитанный вопрос. Побеждает участник, набравший наибольшую сумму после трех раундов (в каждом раунде открываются все 30 полей). В игре много других деталей, но сказанного достаточно, чтобы оценить важность ставки на вопрос-аукцион. Выигрыш или проигрыш часто зависит от стратегии DD-ставок, выбранной участниками.

Открыв поле с вопросом-аукционом, Watson делает ставку, сравнивая ценности действий $\hat{q}(s, bet)$, которые оценивают вероятности выигрыша в текущем состоянии игры s для каждой суммы ставки. Оставляя в стороне некоторые меры снижения риска, описанные ниже, Watson выбирает ставку с максимальной ценностью действия. Ценности действий вычисляются при каждом принятии решения о ставке с использованием двух типов оценок, обученных еще до начала игры. Оценки первого типа – это оценки ценности послесостояний (раздел 6.8), которые возникли бы при выборе каждой допустимой ставки. Эти оценки были получены от функции ценности состояний $\hat{v}(\cdot, \mathbf{w})$ с параметрами $\mathbf{w}$, которая оценивает вероятность выигрыша Watson из любого состояния игры. Оценки ценностей действий второго типа дают «DD-доверие внутри категории» p_{DD} , оценивающей вероятность, что Watson правильно ответит на еще не открытый вопрос-аукцион.

Тезауро с коллегами применили для обучения функции $\hat{v}(\cdot, \mathbf{w})$ описанный выше подход TD-Gammon к обучению с подкреплением: бесхитростное сочетание нелинейного TD(λ) с многослойной ИНС, веса которой $\mathbf{w}$ обучались методом обратного распространения TD-ошибок на многих имитированных играх. Подаваемые на вход сети состояния были представлены векторами признаков, специально сконструированных для *Jeopardy!*. В число признаков входили текущие банки игроков, оставшееся количество вопросов-аукционов, общая стоимость еще не сыгранных вопросов в долларах и другая информация, относящаяся к тому, сколько осталось

играть. В отличие от программы TD-Gammon, которая обучалась на играх сама с собой, функция $\hat{v}$ для Watson обучалась на миллионах имитированных игр против тщательно спроектированных моделей игроков-людей. Оценки DD-доверия внутри категории были обусловлены количеством правильных ответов r и неправильных ответов w , данных Watson на ранее сыгранные вопросы в этой категории. Зависимости от (r, w) оценивались по фактической верности ответов Watson на многих тысячах категорий, встречавшихся в истории игры.

Зная ранее обученную функцию ценности $\hat{v}$ DD-доверия внутри категории p_{DD} , Watson вычисляла $\hat{q}(s, bet)$ для каждой допустимой ставки по следующей формуле:

$$\hat{q}(s, bet) = p_{DD} \times \hat{v}(S_W + bet, \dots) + (1 - p_{DD}) \times \hat{v}(S_W - bet, \dots), \quad (16.2)$$

где S_W – текущая оценка Watson, а $\hat{v}$ дает оценку ценности состояния игры после ответа Watson на вопрос-аукцион, который может быть правильным или неправильным. Такой способ вычисления ценности действий соответствует сделанному в упражнении 3.19 наблюдению, что ценность действия равна математическому ожиданию ценности следующего состояния при условии данного действия (с тем отличием, что здесь это математическое ожидание следующего послесостояния, потому что полное следующее состояние игры в целом зависит от выбора следующего поля).

Тезауро с коллегами обнаружили, что выбор ставки, максимизирующей ценности действий, влечет за собой «пугающий риск». Это означает, что если Watson ответит на вопрос неправильно, то убыток может свести на нет все шансы на выигрыш. Чтобы уменьшить последствия неправильного ответа, авторы программы подправили формулу (16.2), вычтя небольшую долю стандартного отклонения по оценкам послесостояния Watson «правильно/неправильно». И еще снизили риск, запретив ставки, которые в случае послесостояния «неправильный ответ» могли опустить сумму в банке ниже определенного уровня. Эти меры немного уменьшили математическое ожидание выигрыша Watson, зато значительно снизили риск не только в терминах среднего риска на одну ставку, но и особенно в экстремальных ситуациях, когда без учета риска Watson поставила бы все или почти все, что есть у нее в банке.

Почему при обучении функции ценности $\hat{v}$ не использовалась игра с собой, как при обучении TD-Gammon? В случае *Jeopardy!* этот метод работал бы не очень хорошо, потому что Watson сильно отличается от любого игрока-человека. Игра с собой позволила бы исследовать области пространства состояний, не типичные для людей, а особенно чемпионов среди них. Кроме того, в отличие от нарда, *Jeopardy!* – игра с неполной информацией, поскольку у игроков нет доступа ко всей информации, влияющей на поведение соперников. В частности, участники *Jeopardy!* не знают, насколько соперник уверен, отвечая на вопросы из разных категорий. Игра с собой была бы чем-то вроде игры в покер с противником, имеющим те же карты, что и вы.

Из-за описанных осложнений при разработке стратегии DD-ставок в Watson много внимания было уделено созданию хороших моделей соперников-людей. Эти модели не имели отношения к пониманию естественного языка, а лишь к стохастической природе событий во время игры. Статистику брали из обширного созданного поклонниками архива информации об игре с первого дня ее существования. Архив включает такие сведения, как порядок вопросов, правильные и не-

правильные ответы участников, положение вопросов-аукционов на табло, а также ставки в вопросах-аукционах и в финальных раундах, примерно для 300 000 вопросов. Было построено три модели: модель среднего участника (на основе всех данных), модель чемпиона (на основе статистики 100 лучших игроков) и модель великого чемпиона (на основе статистики 10 лучших игроков). Эти модели использовались не только в качестве соперников на этапе обучения, но и для оценки выгоды, которую давала обученная стратегия DD-ставок. Частота выигрышей Watson в имитациях при использовании эталонной эвристической стратегии DD-ставок составляла 61 %, при использовании обученных ценностей и уровня доверия по умолчанию – 64 %, а при использовании реального уровня доверия внутри категории – 67 %. Авторы сочли это значительным улучшением, учитывая, что DD-ставки были нужны всего от 1.5 до 2 раз в каждой игре.

Поскольку у Watson всего несколько секунд, чтобы сделать ставку, и столько же для выбора поля и принятия решения о нажатии кнопки, время вычислений становится критическим фактором. Реализация $\hat{v}$ посредством ИНС позволяла делать DD-ставки достаточно быстро, не выходя за пределы временных ограничений. Однако после того как благодаря усовершенствованию ПО имитационного моделирования появилась возможность моделировать игры достаточно быстро, ближе к концу игры стало возможно оценивать ценность ставок путем усреднения по многим испытаниям Монте-Карло, в которых последствия каждой ставки определялись доигрыванием до конца (на модели). Применение испытаний Монте-Карло вместо ИНС для выбора финальной ставки в настоящей игре значительно улучшило качество Watson, поскольку ошибки в оценках ценности в конце игры могли сильно повлиять на шансы выигрыша. Если бы все решения принимались посредством испытаний Монте-Карло, выбор ставки, наверное, был бы лучше, но такой возможности просто не было из-за сложности игры и временных ограничений.

Хотя способность быстро и правильно отвечать на вопросы, заданные на естественном языке, делает Watson крупным достижением, вклад в ее впечатляющую победу над чемпионами-людьми внесли и изощренные стратегии принятия решений. Пропитируем работу Tesouro et al. (2012):

...совершенно очевидно, что наши алгоритмы выбора стратегии позволили достичь такого уровня точности и производительности в реальном времени, который превосходит человеческие возможности. Особенно наглядно это проявляется в ставках при ответе на вопросы-аукционы и в финальном раунде – человек не может сравняться в точности оценки величины ставки и уровня доверия с программой Watson, выполняющей сложные вычисления.

16.4. ОПТИМИЗАЦИЯ УПРАВЛЕНИЯ ПАМЯТЬЮ

В большинстве компьютеров в качестве оперативной памяти используются динамические запоминающие устройства с произвольным доступом (ДЗУПВ, англ. DRAM) из-за их низкой стоимости и высокой емкости. В задачу контроллера DRAM входит эффективное использование интерфейса между процессором и системой DRAM, расположенной вне кристалла процессора, с целью обеспечить высокую пропускную способность и низкую задержку передачи данных, т. к. это обяза-

тельное условие высокой скорости выполнения программы. Контроллеру памяти приходится иметь дело с динамически изменяющимся характером запросов чтения-записи, выдерживая при этом многочисленные временные и ресурсные ограничения, налагаемые оборудованием. Это сложнейшая задача планирования, особенно при работе с современными процессорами, оснащенными несколькими ядрами, которые обращаются к одной и той же DRAM-памяти.

В работе İpek, Mutlu, Martínez, and Caruana (2008) (также Martínez and İpek, 2009) спроектирован контроллер памяти на основе обучения с подкреплением и продемонстрировано, что он способен значительно увеличить скорость выполнения программы по сравнению с тем, что было доступно традиционным контроллерам на момент разработки. Побудительным мотивом стали ограничения существующих контроллеров, в которых стратегии планирования не опирались на прошлый опыт и не учитывали долгосрочных последствий принимаемых решений. Проект был осуществлен на имитационной модели, но авторы выполнили детальное проектирование, необходимое для реализации оборудования – включая алгоритм обучения – прямо на кристалле процессора.

Доступ к DRAM включает несколько шагов, которые должны быть выполнены с соблюдением строгих временных ограничений. Система DRAM состоит из нескольких микросхем, каждая из которых содержит несколько прямоугольных массивов ячеек памяти, организованных в виде строк и столбцов. В каждой ячейке хранится один бит, представленный зарядом конденсатора. Поскольку со временем конденсатор разряжается, ячейки необходимо подзаряжать – обновлять – каждые несколько миллисекунд, иначе содержимое памяти будет потеряно. Именно из-за этого свойства ячеек DRAM-память называется «динамической».

С каждым массивом ячеек ассоциирован буфер строки, в который (или из которого) можно скопировать одну строку массива. Команда *активации* «открывает строку», т. е. перемещает содержимое строки с указанным в команде адресом в буфер строки. Когда строка открыта, контроллер может выдать команды *чтения* или *записи* массиву ячеек. Команда чтения помещает одно слово (короткую последовательность соседних битов) из буфера строки на внешнюю шину данных, а команда записи помещает слово с внешней шины данных в буфер строки. Перед тем как открыть другую строку, необходимо выполнить команду *подготовки*, которая копирует (возможно, обновленные) данные из буфера строки обратно в адресованную строку массива ячеек. Затем можно получить доступ к новой строке, еще раз выполнив команду активации. Команды чтения и записи *постолбцовые*, потому что последовательно перемещают биты в столбцы или из столбцов буфера строки; несколько битов можно переместить, не открывая строку повторно. Команды чтения и записи в уже открытую строку можно выполнить быстрее, чем доступ к другой строке, для чего потребовались бы дополнительные команды подготовки и активации; иногда это называют «строковой локальностью». Контроллер памяти поддерживает очередь транзакций памяти, в которой хранятся запросы на доступ от всех процессоров, сообщая обращающимся к системе памяти. Контроллер должен обрабатывать запросы, посылая команды системе памяти и соблюдая многочисленные временные ограничения.

Принятая контроллером стратегия планирования запросов на доступ может оказывать сильное влияние на производительность системы памяти, например на среднюю величину задержки выполнения запросов и максимально достижи-

мую пропускную способность. В случае простейшей стратегии запросы обрабатываются в порядке поступления – все команды, необходимые для выполнения одного запроса, посылаются, перед тем как приступить к обслуживанию другого. Но если система не готова к выполнению какой-то команды или если выполнение команды привело бы к недогруженности ресурсов (например, вследствие временных ограничений, возникших в связи с обслуживанием этой команды), то имеет смысл начать обслуживание следующего запроса, не дожидаясь завершения предыдущего. Эффективность стратегии можно повысить путем переупорядочения запросов, например отдавать предпочтение запросам чтения перед запросами записи или командам чтения-записи в уже открытую строку. Стратегия First-Ready, First-Come-First-Serve (FR-FCFS) (первый готовый, первый пришедший первым обслужен) отдает предпочтение постолбцовым командам (чтения и записи) перед построчными (активации и подготовки), а в случае неоднозначности – самой ранней команде. Доказано, что стратегия FR-FCFS превосходит другие стратегии планирования в терминах средней задержки доступа к памяти при типичных условиях (Rixner, 2004).

На рис. 16.3 представлено схематичное изображение контроллера памяти на основе обучения с подкреплением. Авторы смоделировали процесс доступа к DRAM как МППР, состояниями которого является содержимое очереди транзакций, а действиями – команды системе DRAM: подготовка, активация, чтение, запись и пустая операция (NoOp). Сигнал вознаграждения равен 1, когда действием является чтение или запись, и 0 в противном случае. Переходы состояний считаются стохастическими, потому что следующее состояние системы зависит не только от команды планировщика, но и от таких аспектов поведения системы, которые находятся вне контроля планировщика, например от рабочей нагрузки на процессорные ядра, обращающиеся к системе DRAM.

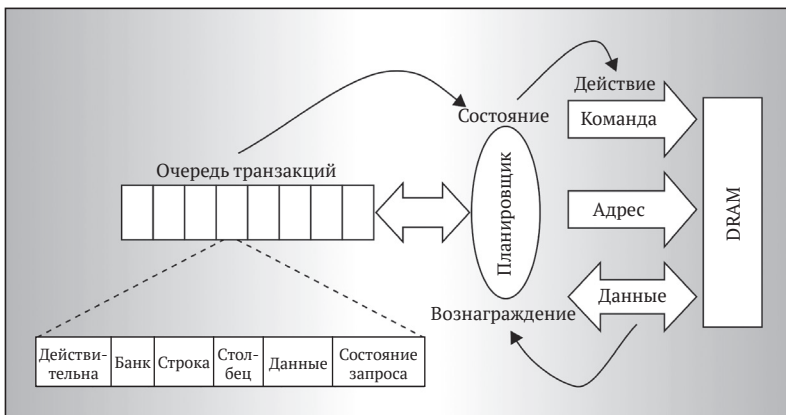


Рис. 16.3 ❖ Схематичное изображение контроллера DRAM на основе обучения с подкреплением. Планировщик является агентом обучения с подкреплением. Окружающая среда представлена признаками очереди транзакций, а действиями являются команды системе DRAM. © 2009 IEEE. Печатается с разрешения авторов статьи J. F. Martinez and E. İpek «Dynamic multicore resource management: A machine learning approach», Micro, IEEE, 29(5), p. 12

Критическим фактором для этого МППР являются ограничения на действия, доступные в каждом состоянии. Напомним (см. главу 3), что множество доступных действий может зависеть от состояния: $A_t \in \mathcal{A}(S_t)$, где A_t – действие на временном шаге t , а $\mathcal{A}(S_t)$ – множество действий, доступных в состоянии S_t . В данном приложении для гарантии целостности системы DRAM должны быть запрещены действия, нарушающие временные или ресурсные ограничения. Хотя это не оговорено явно, Ипек с коллегами добились нужного эффекта, предопределив множества $\mathcal{A}(S_t)$ для всех возможных состояний S_t .

Наличие ограничений объясняет, зачем в МППР нужно действие *NoOp* и почему сигнал вознаграждения равен 0 во всех случаях, кроме отправки команд *чтение* и *запись*. Чтобы максимально задействовать систему памяти, контроллер должен переводить систему в состояния, где можно выбрать действие *чтение* или *запись*: только такие действия приводят к передаче данных по внешней шине, и, следовательно, только они повышают пропускную способность системы. Хотя команды *подготовка* и *активация* не приносят немедленного вознаграждения, агент должен выбирать эти действия, чтобы впоследствии можно было выбрать вознаграждаемые действия *чтение* и *запись*.

Агент планирования применяет алгоритм Sarsa (раздел 6.4) для обучения функции ценности действий. Состояния представлялись шестью целочисленными признаками. Для линейной аппроксимации функции ценности действия применялось плиточное кодирование с хешированием (раздел 9.5.4). Было использовано 32 замощения, в каждом из которых хранилось 256 ценностей действий в виде 16-разрядных чисел с фиксированной точкой. Исследование было ϵ -жадным с $\epsilon = 0.05$.

В состав признаков состояний были включены количество запросов чтения в очереди транзакций, количество запросов записи в очереди транзакций, количество запросов записи в очереди транзакций, ожидающих открытия новой строки, и количество запросов чтения в очереди транзакций, ожидающих открытия новой строки и являющихся самыми старыми из отправленных соответствующими процессорами. (Другие признаки зависели от взаимодействия DRAM с кеш-памятью, эти детали мы здесь опускаем.) Выбор именно этих признаков состояний определялся тем, как авторы понимали факторы, влияющие на производительность DRAM. Например, балансирование скорости обслуживания запросов чтения и записи в зависимости от того, сколько запросов каждого вида находится в очереди транзакций, может предотвратить приостановку взаимодействия системы DRAM с кеш-памятью. На самом деле авторы составили довольно длинный список потенциальных признаков, а затем сократили его до небольшой горстки, воспользовавшись моделированием с отбором признаков на каждом шаге.

Интересный аспект постановки задачи планирования как МППР заключается в том, что признаки, подаваемые на вход плиточному кодированию для нахождения функции ценности действий, отличались от признаков, используемых для задания множеств допустимых действий $\mathcal{A}(S_t)$. Если входные данные для плиточного кодирования основывались на содержимом очереди транзакций, то множества $\mathcal{A}(S_t)$ определялись рядом других признаков, связанных с временными и ресурсными ограничениями, которым должна была удовлетворять аппаратная реализация системы в целом. Таким образом, ограничения на действия гарантировали, что этап исследования в алгоритме обучения не поставит под угрозу

целостность физической системы, коль скоро обучение будет фактически ограничено той «безопасной» областью гораздо большего пространства состояний, которая допускается аппаратной реализацией.

Поскольку целью данной работы было доказать, что самообучающийся контроллер можно реализовать на кристалле, так что обучение будет происходить в онлайн-режиме, когда компьютер работает, деталями аппаратной реализации нельзя было пренебречь. Проект включал два пятиступенчатых конвейера для вычисления и сравнения двух ценностей действий на каждом такте процессора и обновления подходящей ценности действия. Для этого был необходим доступ к данным плиточного кодирования, которые хранились в статической памяти на кристалле. В конфигурации, смоделированной авторами (4-ядерный процессор с частотой 4 ГГц, типичный для тогдашних высокопроизводительных рабочих станций), было 10 тактов процессора на каждый такт DRAM. Учитывая такты, необходимые для заполнения конвейеров, на каждом такте DRAM можно было оценить до 12 действий. Авторы обнаружили, что количество допустимых команд в каждом состоянии редко бывает больше и что потеря производительности из-за эпизодической нехватки времени на рассмотрение всех допустимых команд пренебрежимо мала. Благодаря этой и другим хитроумным деталям проекта оказалось возможно реализовать полный контроллер вместе с алгоритмом обучения на многоядерном кристалле.

Для оценки своего самообучающегося контроллера в процессе имитационного моделирования Ипек с коллегами сравнивали его с тремя другими: 1) вышеупомянутый контроллер типа FR-FCFS с наилучшей средней производительностью; 2) традиционный контроллер, обрабатывающий запросы по порядку; 3) неосуществимый идеальный контроллер, называемый оптимистическим, который способен поддерживать стопроцентную пропускную способность DRAM при достаточном спросе, игнорируя все временные и ресурсные ограничения, но с учетом задержки DRAM (из-за необходимости загружать буфер строки) и ширины полосы пропускания. Было смоделировано девять типов параллельной рабочей нагрузки с интенсивным доступом к памяти, включая научные расчеты и приложения для добычи данных. На рис. 16.4 показана производительность (обратное время выполнения, нормированное на производительность FR-FCFS) каждого контроллера для всех девяти приложений, а также среднегеометрическая производительность по всем приложениям. Самообучающийся контроллер, обозначенный на рисунке RL, для всех приложений показал результаты, превышающие FR-FCFS на 7–33 %. Разумеется, никакой практически реализуемый контроллер не может сравниться по производительности с оптимистическим, но разрыв между ним и самообучающимся контроллером уменьшился на 27 %, что впечатляет.

Поскольку при реализации алгоритма обучения на кристалле ставилась цель в онлайн-режиме адаптировать стратегию планирования к изменяющейся рабочей нагрузке, авторы сравнили онлайн-обучение с предварительно обученной фиксированной стратегией. Они обучили свой контроллер на данных всех девяти эталонных приложений, а затем зафиксировали получившиеся ценности действий в течение всего выполнения приложений на модели. Обнаружилось, что средняя производительность контроллера, обучавшегося в онлайн-режиме, на 8 % выше, чем у контроллера с фиксированной стратегией. Это позволило сделать вывод, что онлайн-обучение – важное свойство предложенного подхода.

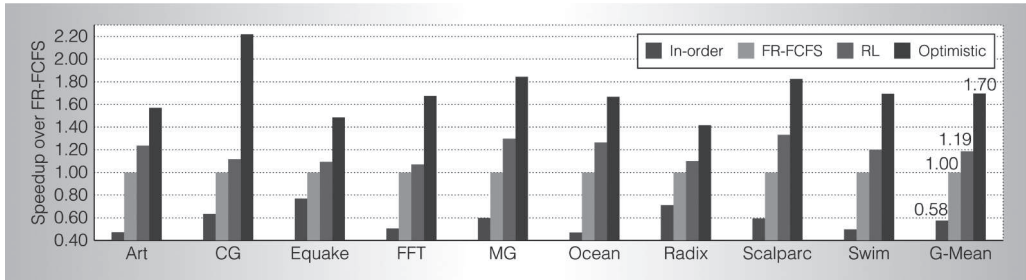


Рис. 16.4 ♦ Производительность четырех контроллеров на наборе из 9 эталонных приложений. Оценивались следующие контроллеры: простейший контроллер «по порядку», FR-FCFS, самообучающийся контроллер RL и неосуществимый оптимистический контроллер, который игнорирует все временные и ресурсные ограничения и дает теоретическую верхнюю границу производительности. В качестве меры производительности взято обратное время выполнения, нормированное на производительность FR-FCFS. Справа показано среднее геометрическое производительностей во всем 9 эталонным приложениям для каждого контроллера. Контроллер RL ближе всего к идеальной производительности. ©2009 IEEE. Печатается с разрешения авторов статьи J. F. Martínez and E. İpek «Dynamic multicore resource management: A machine learning approach», Micro, IEEE, 29(5), p. 13

Самообучающийся контролер так никогда и не был воплощен в «железе» из-за высокой стоимости производства. Тем не менее Ипек с коллегами с помощью результатов моделирования смогли убедительно доказать, что контроллер памяти, обучающийся с подкреплением в онлайн-режиме, потенциально способен улучшить производительность до уровня, который в противном случае потребовал бы более сложных и дорогих систем памяти, и при этом с конструкторов снимается бремя ручного проектирования эффективных стратегий планирования. В работе Mukundan and Martínez (2012) этот проект получил дальнейшее развитие – были изучены самообучающиеся контроллеры с дополнительными действиями, другими критериями производительности и более сложными функциями вознаграждения, выведенными с помощью генетических алгоритмов. Рассматривался критерий производительности, связанный с энергоэффективностью. Результаты этих исследований превзошли предыдущие и намного опережали имевшееся в 2012 году оборудование по всем рассмотренным критериям. Этот подход представляется особенно многообещающим при разработке сложных интерфейсов к DRAM, учитывающих энергопотребление.

16.5. ИГРА В ВИДЕОИГРЫ НА УРОВНЕ ЧЕЛОВЕКА

Одна из самых сложных проблем, возникающих при применении обучения с подкреплением к реальным задачам, – как представлять и хранить функции ценности и стратегии. Если множество состояний не является конечным и достаточно малым, так что можно построить исчерпывающее представление с помощью таблицы соответствия (как во многих наших иллюстративных примерах), то приходится использовать аппроксимацию параметрической функцией. Аппроксима-

ция – все равно, линейная или нелинейная, – опирается на признаки, которые должны быть легко доступны системе обучения и несут информацию, необходимую для квалифицированной работы. Большинство успешных приложений обучения с подкреплением многим обязаны наборам признаков, которые были тщательно сконструированы на основе знаний о конкретной задаче и ее интуитивного понимания.

Группа исследователей из компании Google DeepMind разработала впечатляющую демонстрацию, доказывающую, что глубокая многослойная ИНС способна автоматизировать процесс конструирования признаков (Mnih et al., 2013, 2015). Многослойная ИНС использовалась для аппроксимации функций в обучении с подкреплением еще с 1986 года, когда получил широкую известность алгоритм обратного распространения как метод обучения внутренних представлений (Rumelhart, Hinton, and Williams, 1986; см. раздел 9.7). Сочетание обучения с подкреплением и обратного распространения позволило получить поразительные результаты. Яркими примерами могут служить описанные выше программы TD-Gammon и Watson, созданные Тезауро с коллегами. В этих и других приложениях с успехом использовалась способность многослойных ИНС обучаться релевантной задаче признакам. Однако во всех известных нам примерах самые впечатляющие результаты были достигнуты тогда, когда на вход сети подавались признаки, специально сконструированные для данной задачи. Это наглядно видно на примере программы TD-Gammon. Версия TD-Gammon 0.0, в которой на вход сети подавалось, по сути дела, «голое» представление доски, несущее очень мало информации о специфике нард, обучилась играть примерно на уровне лучших на то время компьютерных программ. Версия TD-Gammon 1.0, в которую были добавлены специализированные для нард признаки, играла значительно лучше всех программ и достойно противостояла игрокам-людям.

Мних с коллегами разработали агента обучения с подкреплением, названного *глубокой Q-сетью* (deep Q-network – DQN), в котором Q-обучение сочеталось с *глубокой сверточной* ИНС, специализированной для обработки пространственных массивов данных, в частности изображений. Мы описывали глубокие сверточные ИНС в разделе 9.7. К моменту выхода работы Мниха с соавторами, посвященной DQN, глубокие ИНС, в т. ч. сверточные, уже принесли впечатляющие результаты во многих приложениях, но еще не получили широкого распространения в обучении с подкреплением.

Мних с коллегами использовали DQN, чтобы показать, как агент обучения с подкреплением может достичь отличных результатов на любом наборе различных задач без необходимости готовить разные специфические наборы признаков. Для демонстрации они обучили DQN играть в 49 разных видеоигр Atari 2600 посредством взаимодействия с игровым эмулятором. DQN обучилась разным стратегиям для каждой из 49 игр (поскольку веса ИНС сбрасывались в случайные значения перед обучением новой игре), но при этом для всех игр использовались одинаковые «сырые» входные данные, архитектура сети и значения параметров (например, размер шага, коэффициент обесценивания, параметры исследования и многие другие, специфичные для реализации). В большинстве случаев DQN научилась играть на том же или более высоком уровне, что человек. Хотя игры были похожи в том смысле, что обучение производилось путем наблюдения за потоком видео, они различались в других аспектах. Действия имели разный эффект, раз-

личалась динамика перехода состояний, и для достижения высоких результатов требовались разные стратегии. Глубокая сверточная ИНС обучилась преобразовывать входные данные в признаки, специализированные для представления ценности действий для игры на высоком уровне.

Atari 2600 – это домашняя игровая консоль, которая в различных вариантах продавалась компанией Atari Inc. с 1977 по 1992 год. Для нее было специально написано или перенесено много аркадных видеоигр, которые теперь считаются классикой, например: Pong, Breakout, Space Invaders и Asteroids. Конечно, игры для Atari 2600 гораздо проще современных, но все равно увлекательны и трудны для игроков-людей. Они давно привлекали внимание как испытательная лаборатория для разработки и оценки методов обучения с подкреплением (Diuk, Cohen, Littman, 2008; Naddaf, 2010; Cobo, Zang, Isbell, and Thomaz, 2011; Bellemare, Veness, and Bowling, 2013). В работе Bellemare, Naddaf, Veness, and Bowling (2012) создана общедоступная среда Arcade Learning Environment (ALE) с целью упростить и привлечь энтузиастов, желающих использовать эти написанные игры для изучения алгоритмов обучения и планирования.

Благодаря исследованиям предшественников и доступности среды ALE Мних с коллегами выбрали коллекцию игр Atari 2600 для демонстрации. На это решение также повлияло ошеломительное качество программы TD-Gammon, сравнившейся по уровню игры в нарды с человеком. DQN похожа на TD-Gammon в части использования многослойной ИНС как метода аппроксимации функций для полугradientной формы TD-алгоритма, в которой градиенты вычислялись посредством обратного распространения. Но вместо $TD(\lambda)$, как в TD-Gammon, в DQN использовалась полугradientная форма Q-обучения. В TD-Gammon оценивались ценности послесостояний, которые было легко получить из правил ходов в нардах. Чтобы воспользоваться таким же алгоритмом для игр Atari, нужно было бы сгенерировать следующие состояния для каждого возможного действия (которые в этом случае не были бы послесостояниями). Это можно было бы сделать, взяв игровой эмулятор и прогнав одношаговые имитации для всех возможных действий (ALE это позволяет). Или можно было для каждой игры построить модель функции перехода состояний, обучить ее и использовать для предсказания следующих состояний (Oh, Guo, Lee, Lewis, and Singh, 2015). Возможно, эти методы и дали бы результаты, сравнимые с DQN, но их было бы сложнее реализовать и для обучения понадобилось бы гораздо больше времени. Еще одной причиной остановиться на Q-обучении было то, что в DQN применялся описанный ниже *метод воспроизведения опыта*, для которого требуется алгоритм с разделенной стратегией. Ну, а коль скоро мы говорим о безмодельном алгоритме с разделенной стратегией, то Q-обучение является естественным выбором.

Прежде чем описывать детали DQN и методику проведения экспериментов, посмотрим, какого уровня мастерства удалось достичь DQN. Мних и его соавторы сравнили игровой счет DQN со счетом лучших систем обучения, описанных в литературе на тот момент, а также со счетом профессионального тестировщика игр и со счетом агента, который выбирал действия случайным образом. В лучшей из известных систем использовалась линейная аппроксимация, а признаки конструировались с привлечением знаний об играх для Atari 2600 (Bellemare, Naddaf, Veness, and Bowling, 2013). DQN обучалась каждой игре, взаимодействуя с эмулятором на протяжении 50 млн кадров, что соответствует 38 дням реального игрового

опыта. В начале обучения каждой игре весам сети DQN присваивались случайные значения. Чтобы оценить уровень мастерства DQN после обучения, игровой счет усреднялся по 30 сеансам каждой игры, которые продолжались по 5 минут и начинались в случайном начальном состоянии. Профессиональный тестировщик играл с тем же эмулятором (с выключенным звуком, чтобы устранить возможное преимущество над DQN, которая не обрабатывала звук). После двух часов тренировки человек сыграл без перерывов примерно 20 эпизодов каждой игры не более 5 минут каждый. DQN обучилась играть лучше самой сильной из предшествующих систем обучения во все игры, кроме шести, а человека превзошла в 22 играх. Мних с соавторами посчитали, что если измеряемое счетом качество системы не ниже 75 % от качества игры человека, то программа достигла или даже превысила уровень человека. Поэтому они заключили, что по уровню мастерства обученная DQN сравнялась или превзошла человека в 29 из 46 игр. Более подробный отчет о результатах см. в статье Mnih et al. (2015).

То, что искусственная самообучающаяся система сумела достичь такого мастерства, впечатляет само по себе, но особенно поразительными эти результаты делает то (и тогда многие считали это прорывом в области искусственного интеллекта), что одна и та же система обучения демонстрировала подобный уровень на очень разных играх без каких-либо зависящих от игры модификаций.

Человек, играющий в любую из 49 игр Atari, видит перед собой кадры, представляющие собой 128-цветные изображения размером 210×160 пикселей, сменяющиеся с частотой 60 Гц. В принципе, точно такие же изображения могли бы подаваться на вход DQN, но, чтобы уменьшить требования к памяти и процессору, авторы подвергли каждый кадр предварительной обработке, преобразовав его в массив значений яркости размером 84×84 . Поскольку для знания полных состояний многих игр Atari недостаточно видеть один кадр, Мних с коллегами «составили» четыре последних кадра в стопку, так что вход сети имел размерность $84 \times 84 \times 4$. Это не решило проблему частичной наблюдаемости для всех игр, но могло сделать многие из них больше похожими на марковские процессы.

Важный момент: все шаги предварительной обработки были в точности одинаковыми для всех 46 игр. Не привлекалось никакой зависящей от игры информации, за исключением общего понимания, что сокращенной размерности должно быть достаточно для обучения хорошим стратегиям и что образование стопки соседних кадров должно в какой-то мере решить проблему частичной наблюдаемости некоторых игр. Поскольку сверх этих минимальных предположений на этапе предварительной обработки кадров не было никакой априорной информации, можно считать входные векторы $84 \times 84 \times 4$ исходными данными для DQN.

Базовая архитектура DQN похожа на глубокую сверточную ИНС, показанную на рис. 9.15 (правда, в отличие от той сети, подвыборка в DQN рассматривается как часть сверточного слоя, а карты признаков состоят из блоков, имеющих только часть возможных рецептивных полей). В DQN три скрытых сверточных слоя, за которыми следует один полносвязный скрытый слой и выходной слой. Все три скрытых сверточных слоя DQN порождают 32 карты признаков 20×20 , 64 карты признаков 9×9 и 64 карты признаков 7×7 . В качестве функции активации каждого блока в каждой карте признаков используется ректифицирующая нелинейность ($\max(0, x)$). Все 3136 ($64 \times 7 \times 7$) блоков третьего сверточного слоя соединены с каждым из 512 блоков полносвязного скрытого слоя, которые, в свою очередь, соеди-

нены с каждым из 18 блоков выходного слоя, по одному для каждого действия, возможного в играх Atari.

В качестве уровней активации выходных блоков DQN использовались оценки оптимальных ценностей действий соответствующих пар состояние–действие для состояния, представленного входом сети. Распределение выходных блоков между действиями варьировалось от одной игры к другой, а поскольку количество допустимых действий изменялось от 4 до 18 в зависимости от игры, не у всех выходных блоков была какая-то функциональная роль в каждой игре. Полезно рассматривать эту сеть как 18 отдельных сетей, по одной для оценивания оптимальной ценности каждого из возможных действий. В действительности у этих сетей общие начальные слои, но выходные блоки обучились использовать признаки, выделяемые этими слоями, по-разному.

Сигнал вознаграждения в DQN показывал, как игровой счет изменялся при переходе от предыдущего временного шага к следующему: +1, если увеличивался, −1, если уменьшался, и 0 в противном случае. Это позволило использовать единый сигнал вознаграждения во всех играх, при этом единственный размер шага хорошо работал для всех игр, хотя диапазон очков изменялся. В DQN применялась ϵ -жадная стратегия, где параметр ϵ линейно убывал на первом миллионе кадров и затем сохранял малое значение до конца сеанса обучения. Значения других параметров, в т. ч. размера шага обучения, коэффициента обесценивания и иных, специфичных для реализации, выбирались путем неформального поиска, исходя из того, какие значения давали наилучшие результаты на небольшой выборке игр. Затем эти значения использовались без изменения во всех играх.

Выбранное DQN действие выполнялось игровым эмулятором, который возвращал вознаграждение и следующий кадр видео. Кадр подвергался предварительной обработке и добавлялся в стопку четырех кадров, которая подавалась на вход сети в следующий раз. Опуская (ненадолго) изменения, внесенные в базовую процедуру Q-обучения Мнихом с коллегами, скажем, что в DQN использовалась следующая полугradientная форма Q-обучения для обновления весов сети:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \left[R_{t+1} + \gamma \max_a \hat{Q}(S_{t+1}, a, \mathbf{w}_t) - \hat{Q}(S_t, A_t, \mathbf{w}_t) \right] \nabla \hat{Q}(S_t, A_t, \mathbf{w}_t), \quad (16.3)$$

где $\mathbf{w}_t$ – вектор весов сети, A_t – действие, выбранное на шаге t , а S_t и S_{t+1} – стопки предобработанных изображений, подаваемые на вход сети в моменты t и $t + 1$.

Градиент в формуле (16.3) вычислялся методом обратного распространения. Снова представим, что для каждого действия имеется отдельная сеть; тогда для обновления на шаге t метод обратного распространения применялся только к сети, соответствующей A_t . Авторы воспользовались техникой, которая, как было показано, улучшает базовый алгоритм обратного распространения в применении к большим сетям. Они применили метод мини-пакетов, при котором веса обновляются только после накопления информации о градиенте по небольшому пакету изображений (в данном случае после обработки 32 изображений). Это давало более гладкие градиенты по сравнению с обычной процедурой обновления весов после каждого действия. Авторы также использовали алгоритм градиентного спуска RMSProp (Tieleman and Hinton, 2012), который ускоряет обучение, корректируя размер шага для каждого веса, исходя из скользящего среднего абсолютных величин недавних градиентов по этому весу.

Мних с коллегами модифицировали базовую процедуру Q-обучения в трех отношениях. Во-первых, они использовали метод *воспроизведения опыта*, впервые изученный в работе Lin (1992). Этот метод заключается в сохранении опыта агента на каждом временном шаге в памяти воспроизведения, к которой производятся обращения при обновлении весов. В DQN это работает так. После того как игровой эмулятор выполнил действие A_t в состоянии, представленном стопкой изображений S_t , и вернул вознаграждение R_{t+1} и стопку изображений S_{t+1} , он добавляет кортеж $(S_t, A_t, R_{t+1}, S_{t+1})$ в память воспроизведения. В этой памяти накапливается опыт многих партий, сыгранных в одну и ту же игру. На каждом временном шаге выполнялось несколько обновлений Q-обучения – мини-пакет – на основе равномерной случайной выборки опыта из памяти воспроизведения. Новым состоянием S_t для следующего обновления становилось не S_{t+1} , как было бы при обычном варианте Q-обучения, а данные, получаемые из какого-то несвязанного опыта, выбранного из памяти воспроизведения. Поскольку Q-обучение – алгоритм с разделенной стратегией, его необязательно применять вдоль связанных траекторий.

Q-обучение с воспроизведением опыта дает несколько преимуществ по сравнению с обычной формой. Способность использовать каждый запомненный опыт для многих обновлений позволила DQN более эффективно обучаться на опыте. Воспроизведение опыта уменьшало дисперсию обновлений, поскольку последовательные обновления были не коррелированы друг с другом, как в случае стандартного Q-обучения. А исключив зависимость последовательных опытов от текущих весов, воспроизведение опыта устранило один источник неустойчивости.

Авторы применили еще одну модификацию для повышения устойчивости Q-обучения. Как и в других методах с бутстрэппингом, цель обновления в алгоритме Q-обучения зависит от текущей оценки функции ценности действий. Когда для представления ценностей действий используется параметрический метод аппроксимации функций, целью является функция от тех самых параметров, которые обновляются. Например, целью в обновлении (16.3) является $\max_a \hat{q}(S_{t+1}, a, \mathbf{w}_t)$. Ее зависимость от $\mathbf{w}_t$ усложняет процесс по сравнению с более простой ситуацией обучения с учителем, при котором цели не зависят от обновляемых параметров. Как обсуждалось в главе 11, это может приводить к колебаниям и (или) расходимости.

Для решения этой проблемы Мних с коллегами применили технику, которая приблизила Q-обучение к более простому случаю обучения с учителем, не отказываясь от бутстрэппинга. После выполнения некоторого числа C обновлений весов $\mathbf{w}$ сети ценности действий текущие веса становились параметрами другой сети, и эти дубликаты весов оставались фиксированными в течение следующих C обновлений $\mathbf{w}$. Выходы этой сети-дубликата для следующих C обновлений $\mathbf{w}$ использовались в качестве целей Q-обучения. Если обозначить $\tilde{q}$ выход сети-дубликата, то правило обновления, заменяющее (16.3), принимает вид:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \left[R_{t+1} + \gamma \max_a \tilde{q}(S_{t+1}, a, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t) \right] \nabla \hat{q}(S_t, A_t, \mathbf{w}_t).$$

Последняя модификация стандартного Q-обучения тоже была предпринята во имя повышения устойчивости. Член ошибки $R_{t+1} + \max_a \tilde{q}(S_{t+1}, a, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)$ усекался так, чтобы он не выходил за пределы отрезка $[-1, 1]$.

Мних с коллегами провели много обучающих прогонов на пяти играх, чтобы понять, как различные особенности дизайна DQN влияют на качество работы сети. Ставились эксперименты с четырьмя комбинациями: когда воспроизведение опыта и дублирующая целевая сеть присутствуют или отсутствуют. Конкретные результаты зависели от игры, но в любом случае каждый из этих аспектов по отдельности значительно улучшал качество, а при совместном использовании качество просто взлетало вверх. Авторы также изучили роль глубокой сверточной ИНС в способности DQN к обучению. Для этого версию DQN, оснащенную такой сетью, сравнили с версией, в которой был всего один линейный слой, в условиях, когда на вход подавались одинаковые стопки предобработанных кадров. Здесь превосходство глубокой сверточной версии над линейной было особенно разительным на всех пяти тестовых играх.

Создание искусственных агентов, блистающих при решении широкого круга трудных задач, всегда было целью искусственного интеллекта. Надеждам на то, что машинное обучение станет средством для достижения этой цели, не суждено было сбыться из-за необходимости скрупулезно конструировать представления признаков для каждой задачи. Сеть DQN компании DeepMind стала крупным достижением, поскольку продемонстрировала, что один агент может обучиться зависящим от задачи признакам и подняться до уровня, сопоставимого с человеком, на различных задачах. Эта демонстрация не привела к созданию агента, который блистал сразу во всех задачах (поскольку обучение производилось для каждой задачи отдельно), но доказала, что глубокое обучение может сократить, а возможно, и вовсе устранить необходимость зависящего от задачи конструирования и настройки. Но, как отмечали Мних с коллегами, DQN не дает полного решения проблемы не зависящего от задачи обучения. Хотя навыки, необходимые для овладения различными играми Atari, заметно различались, все они были приобретены в результате наблюдения видеоизображений, потому-то глубокая сверточная ИНС и стала естественным выбором для этой совокупности задач. Кроме того, на некоторых играх для Atari 2600 сеть DQN по качеству игры существенно уступала человеку. Трудные для DQN игры – особенно «Месть Монтесумы», в которой DQN обучилась играть на уровне игрока, действующего случайным образом, – требовали глубокого планирования, что выходило за рамки запроюктированных возможностей DQN. Кроме того, обучение навыкам управления путем длительной практики, а именно так DQN обучалась играть в игры Atari, – лишь один из многих типов обучения, свойственных человеку. Но, несмотря на все ограничения, DQN раздвинула границы возможного в машинном обучении, убедительно продемонстрировав перспективы, которые сулит сочетание обучения с подкреплением и современных методов глубокого обучения.

16.6. МАСТЕРСТВО ИГРЫ В ГО

Старинная китайская игра го много десятилетий была вызовом для исследователей в области искусственного интеллекта. Методы, с помощью которых другие игровые программы приближались к уровню человека или даже превосходили его, для го не работали. Благодаря усилиям очень активного сообщества программистов го и международным соревнованиям уровень программ игры в го с года-

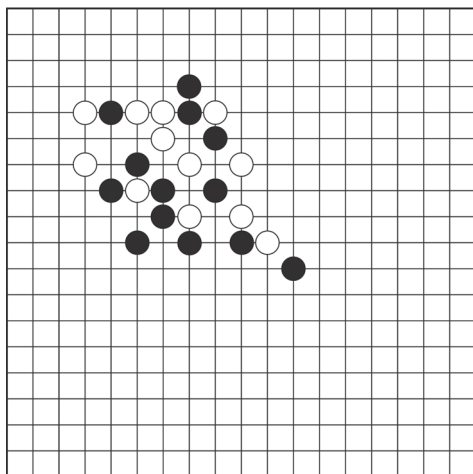
ми существенно возрос. Но до недавнего времени ни одна программа даже приблизиться не могла к уровню мастеров.

Команда из компании DeepMind (Silver et al., 2016) разработала программу AlphaGo, которая преодолела этот барьер, объединив глубокие ИНС (раздел 9.7), обучение с учителем, поиск по дереву методом Монте-Карло (ПДМК, раздел 8.11) и обучение с подкреплением. К моменту публикации работы Силвера с соавторами AlphaGo уже была несомненно сильнее всех остальных существовавших на тот момент программ игры в го и победила чемпиона Европы по го Фан Ху со счетом 5:0. Это была первая победа программы над профессиональным игроком в го, одержанная в полных партиях без форы. Вскоре после этого похожая версия AlphaGo одержала ошеломительную победу над 18-кратным чемпионом мира Ли Седолем, выиграв четыре из пяти партий в матче вызова. Тогда эта новость произвела фурор во всем мире. Специалисты по искусственному интеллекту полагают, что для достижения такого уровня игры понадобится еще много лет, быть может, даже десятилетий.

Здесь мы опишем AlphaGo и последовавшую за ней программу AlphaGo Zero (Silver et al. 2017a). Если AlphaGo опиралась не только на обучение с подкреплением, но и на обучение с учителем по большой базе данных ходов, сделанных мастерами, то в AlphaGo Zero использовалось лишь обучение с подкреплением и никаких данных или инструкций от людей, помимо основных правил игры (отсюда и слово Zero в названии). Сначала более-менее детально опишем AlphaGo, чтобы стало наглядно видно, насколько проста по сравнению с ней AlphaGo Zero, которая при этом играет лучше и с большим основанием может быть названа чистым примером обучения с подкреплением.

Во многих отношениях AlphaGo и AlphaGo Zero – преемники программы Тезауро TD-Gammon (раздел 16.1), которая сама ведет происхождение от шашечной программы Сюмюэла (раздел 16.2). Во всех этих программах участвовало обучение с подкреплением на имитированных играх с собой. При создании AlphaGo и AlphaGo Zero учитывался также опыт, полученный DeepMind при обучении играм Atari программы DQN (раздел 16.5), в которой использовались глубокие ИНС для аппроксимации оптимальных функций ценности.

В го два игрока попеременно ставят черные и белые «камни» в незанятые точки пересечения линий, или «пункты» на доске 19×19 . В результате получают позиции типа показанной на рисунке справа. Цель игры – окружить камнями своего цвета территорию, большую, чем противник. Правила захвата камней просты. Камни игрока захватываются, если они со всех сторон окружены камнями противника, т. е. ни по горизонтали, ни по вертикали не осталось незанятых пунктов. Так, на рис. 16.5 слева показаны три белых камня, рядом с которыми есть незанятый пункт (помечен крестиком X). Если игрок поставит черный камень в пункт X, то все три белых камня будут захвачены



Позиция на доске при игре в го

и сняты с доски (рис. 16.5 в центре). Но если бы белым удалось первыми поставить свой камень в X, то опасность захвата была бы предотвращена (рис. 16.5 справа). Нужны дополнительные правила, препятствующие бесконечным циклам захвата и перезавхвата. Игра заканчивается, когда ни один игрок не желает ставить камень – пасует. Несмотря на простоту правил, игра оказывается очень сложной, и интерес к ней не угасает уже тысячи лет.

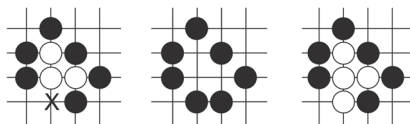


Рис. 16.5 ♠ Правило захвата в го. **Слева:** три белых камня не окружены, потому что пункт X не занят. **В центре:** если поставить в X черный камень, то все три белых будут захвачены и сняты с доски. **Справа:** если сначала поставить в X белый камень, то захват будет предотвращен

Методы, оказавшиеся успешными при программировании других игр, например шахмат, для го не работают. Пространство поиска в го намного больше, чем в шахматах, потому что в любой позиции го больше допустимых ходов, чем в шахматах (≈ 250 против ≈ 35), и количество ходов в одной партии в го больше, чем в шахматах (≈ 150 против ≈ 80). Но не размер пространства поиска – главная причина трудности го. Исчерпывающий поиск невозможен ни в шахматах, ни в го, а игра в го на меньшей доске (например, 9×9) все равно оказалась очень трудной. Специалисты согласны в том, что основной камень преткновения при создании программы, играющей на уровне выше любительского, – трудность определения подходящей функции оценки позиции. Хорошая функция оценки позволяет прекращать поиск, достигнув практически приемлемой глубины, поскольку дает относительно просто вычисляемые предсказания, с большой вероятностью совпадающие с теми, что дал бы поиск на большую глубину. Согласно работе Müller (2002), «простая и при этом разумная функция оценки для го не будет найдена никогда». Крупным шагом вперед стало включение ПДМК в программы игры в го. Все сильнейшие программы, существовавшие на момент разработки AlphaGo, включали ПДМК, но уровня мастера так и не достигли.

Напомним (см. раздел 8.11), что ПДМК – это процедура планирования в момент принятия решений, которая не пытается обучить и сохранить глобальную функцию оценки. Как и разыгрывающий алгоритм (раздел 8.10), она прогоняет много имитаций Монте-Карло целых эпизодов (в данном случае – полных игр в го) для выбора каждого действия (в данном случае – каждого хода: выбрать место для камня или сдать ся). Но, в отличие от простого алгоритма разыгрывания, ПДМК является итеративной процедурой и инкрементно расширяет дерево поиска, корень которого представляет текущее состояние окружающей среды. Как показано на рис. 8.10, на каждой итерации производится обход дерева путем имитации действий в соответствии со статистикой, ассоциированной с ребрами дерева. В базовом варианте, когда имитация доходит до листового узла дерева поиска, ПДМК расширяет дерево, добавляя в него некоторые или все дочерние узлы этого листового узла. Из листового узла или одного из его вновь добавленных потомков

выполняется разыгрывание: имитация, которая обычно проходит весь путь до заключительного состояния, выбирая действия, диктуемые стратегией разыгрывания. По завершении разыгрывания статистика, ассоциированная с ребрами дерева поиска, которые участвовали в проходе, обновляется путем проталкивания вверх по дереву дохода, полученного при разыгрывании. ПДМК продолжает этот процесс, каждый раз начиная из корня дерева поиска в текущем состоянии. Количество итераций определяется временными ограничениями. И в конце выбирается то действие в корневом узле (который по-прежнему представляет текущее состояние окружающей среды), которое диктуется статистикой, накопленной в исходящих из этого узла ребрах. Это действие и выполняет агент. После перехода окружающей среды в следующее состояние ПДМК выполняется снова, но корневой узел теперь представляет новое текущее состояние. Дерево поиска в начале нового выполнения может содержать только новый корневой узел или еще и потомков этого узла, оставшихся от предыдущего выполнения ПДМК. Вся остальная часть дерева отбрасывается.

16.6.1. AlphaGo

Основным новшеством, благодаря которому AlphaGo стала так хорошо играть, был порядок выбора ходов, определяемый новой версией ПДМК. Он зависел от стратегии и функции ценности, найденных с помощью обучения с подкреплением, при котором для аппроксимации функций использовалась глубокая сверточная ИНС. Была и еще одна важная особенность – веса сети в начале обучения с подкреплением выбирались не случайным образом, а были результатом предшествующего обучения с учителем на большом наборе ходов, сделанных мастерами-людьми.

Команда DeepMind назвала примененную в AlphaGo модификацию базового ПДМК «ПДМК с асинхронной стратегией и функцией ценности» (*asynchronous policy and value MCTS*), или АСЦ-ПДМК. Действия выбирались, как в базовом ПДМК, описанном выше, но с некоторыми изменениями в порядке расширения дерева поиска и оценивании ребер действия. В отличие от базовой ПДМК, которая расширяет текущее дерево поиска, используя сохраненные ценности действий для выбора неисследованного ребра из листового узла, процедура АСЦ-ПДМК, реализованная в AlphaGo, расширяла дерево посредством выбора ребра в соответствии с вероятностями, которые вычисляла 13-слойная глубокая сверточная ИНС, названная *ОУ-сеть стратегии* (SL-policy network). Эта сеть была предварительно обучена предсказывать ходы, для чего использовалось обучение с учителем на базе данных, содержащей приблизительно 30 млн ходов, сделанных мастерами.

Затем – и это тоже отличие от базового ПДМК, который оценивает вновь добавленный узел только на основе дохода, полученного в результате начатого из него разыгрывания, – АСЦ-ПДМК оценивала узел двумя способами: по доходу в результате разыгрывания и по функции ценности v_θ , ранее обученной методом обучения с подкреплением. Если s – вновь добавленный узел, то его ценность равна

$$v(s) = (1 - \eta)v_\theta(s) + \eta G, \quad (16.4)$$

где G – доход от разыгрывания, а параметр η определял доли ценностей, получающихся в результате этих двух методов оценивания. В AlphaGo эти ценности

возвращала *сеть ценности* – еще одна 13-слойная сверточная ИНС, обученная, как описано ниже, выводить оценки ценности позиций на доске. Разыгрываниями в АСЦ-ПДМК были имитированные партии, в которых оба игрока использовали стратегию быстрого разыгрывания, возвращаемую простой линейной сетью, также заранее обученной с учителем. В процессе разыгрывания АСЦ-ПДМК запоминала, сколько имитаций прошло по каждому ребру дерева поиска, а по завершении ребро с наибольшим количеством посещений выбиралось в качестве предпочтительного действия, т. е. хода, который AlphaGo и делала.

Сеть ценности имела такую же структуру, как глубокая сверточная ОУ-сеть стратегии, с тем отличием, что у нее был единственный выходной блок, который давал оценки ценности позиций, а не распределения вероятностей допустимых действий, как ОУ-сеть стратегии. В идеале сеть ценности должна была бы выводить ценности оптимальных состояний, и было бы возможно аппроксимировать оптимальную функцию ценности по аналогии с программой TD-Gammon: игра с самой собой с применением нелинейного TD(λ) вкупе с глубокой сверточной ИНС. Но команда DeepMind избрала другой подход, суливший большую выгоду в такой сложной игре, как го. Они разделили процесс обучения сети ценности на два этапа. На первом этапе была создана наилучшая из возможных стратегия, для чего они с подкреплением обучили *сеть ОП-стратегии*. Это была глубокая сверточная сеть с такой же структурой, как у сети ОУ-стратегии. Она была инициализирована конечными весами сети ОУ-стратегии, обученной с учителем, после чего для улучшения ОУ-стратегии был применен алгоритм градиента стратегии. На втором этапе обучения сети ценности команда применила оценивание стратегии методом Монте-Карло по данным, полученным на большом числе имитированных игр с самой собой, в которых ходы выбирались сетью ОП-стратегии.

На рис. 16.6 показаны сети, использованные в AlphaGo, и предпринятые для их обучения шаги, которые команда DeepMind называла «конвейером AlphaGo». Все эти сети были обучены еще до первой настоящей партии, и их веса во время реальной игры не менялись.

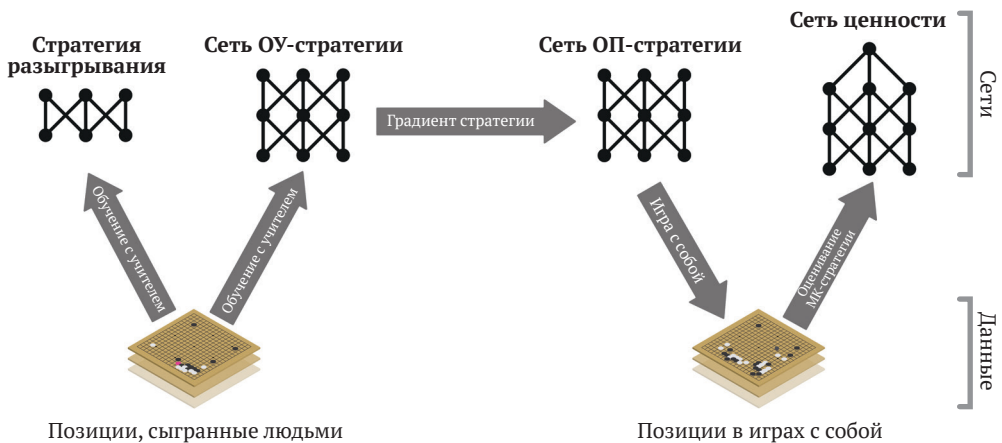


Рис. 16.6 ♠ Конвейер AlphaGo

Приведем некоторые детали устройства ИНС в AlphaGo и их обучения. Сети ОУ- и ОП-стратегии обладали одинаковой структурой и были похожи на описанную в разделе 16.5 глубокую сверточную сеть DQN для игр Atari, только в них было 13 сверточных слоев, а последний слой содержал блоки softmax для каждого пункта на доске для игры в го размером 19×19 . На вход сети подавались стопки изображений $19 \times 19 \times 48$, в которых каждый пункт доски был представлен значениями 48 бинарных или целочисленных признаков. Например, для каждого пункта существовал один признак, показывающий, занят он камнем AlphaGo, камнем противника или вообще не занят, – это было «сырое» представление позиции на доске. Другие признаки основывались на правилах го, например: количество незанятых соседних пунктов, количество камней противника, которые будут захвачены, если в данный пункт поместить камень, количество передач хода с момента помещения камня в данный пункт и прочие признаки, которые команда сочла важными.

На обучение сети ОУ-стратегии ушло примерно три недели, при этом использовалась распределенная реализация стохастического градиентного подъема на 50 процессорах. Сеть достигла верности 57 %, в то время как наилучшая верность, достигнутая другими группами на момент публикации, составляла 44.4 %. Для обучения с подкреплением сети ОП-стратегии применялся алгоритм градиента стратегии на имитированных играх между текущей сетью ОУ-стратегии и противниками, которые пользовались случайно выбранными стратегиями, полученными на более ранних стадиях алгоритма обучения. Игра против случайно выбранного набора соперников предотвращала чрезмерно близкую подгонку к текущей стратегии. Сигнал вознаграждения был равен +1, если текущая стратегия выигрывала, –1, если проигрывала, и 0 в противном случае. В этих партиях две стратегии сталкивались друг с другом напрямую, без участия ПДМК. Имитация многих партий одновременно на 50 процессорах позволила обучить сеть ОП-стратегии на миллионе партий за один день. Во время тестирования конечной ОП-стратегии выяснилось, что она выигрывает более 80 % партий против ОУ-стратегии и 85 % партий против программы, в которой использовался ПДМК с имитацией 100 000 партий на каждом ходе.

Структура сети ценности была похожа на сети ОУ- и ОП-стратегии с тем отличием, что у нее был всего один выходной блок. Эта сеть принимала те же входы данных, что сети ОУ- и ОП-стратегии с дополнительным бинарным признаком, обозначавшим текущий цвет камней. Авторы использовали оценивание стратегии методом Монте-Карло, чтобы обучить эту сеть на данных, полученных в результате игры с самой собой с применением ОП-стратегии. Чтобы избежать переобучения и неустойчивости из-за сильной корреляции между позициями, встречающимися при игре с самой собой, команда DeepMind построила набор данных из 30 млн позиций, случайно выбранных из единственной партии с собой. Затем было выполнено обучение с помощью 50 млн мини-пакетов, каждый из которых содержал 32 позиции, случайно выбранных из этого набора данных. Обучение заняло неделю на 50 GPU.

Стратегия разыгрывания была заранее обучена с учителем с помощью простой линейной сети на корпусе из 8 млн ходов, сделанных людьми. Сеть стратегии разыгрывания должна была выводить действия быстро, сохраняя при этом разумную верность. В принципе, сети ОУ- и ОП-стратегии тоже можно было бы

использовать для разыгрывания, но прямое распространение в этих глубоких сетях занимало слишком много времени для применения в имитациях разыгрывания – а таких имитаций нужно было выполнить много при каждом решении о выборе хода в реальной партии. По этой причине сеть стратегии разыгрывания пришлось сделать более простой, чем остальные сети стратегии, и входные признаки для нее тоже вычислялись быстрее. Сеть стратегии разыгрывания позволяла выполнять приблизительно 1000 полных имитаций партий в секунду в каждом потоке обработки в программе AlphaGo.

Может возникнуть вопрос, почему для выбора действий на этапе расширения АСЦ-ПДМК использовалась ОУ-стратегия, а не превосходящая ее ОП-стратегия. Время вычислений для обеих стратегий было примерно одинаковым, т. к. одинаковой была архитектура сети. Но команда обнаружила, что AlphaGo лучше играла против человека, когда в АСЦ-ПДМК использовалась именно ОУ-стратегия. Они отнесли это на счет того, что ОП-стратегия настраивалась для ответа на оптимальные ходы, а не на более широкое множество ходов, характерное для игры человека. Интересно, что в случае функции ценности, используемой в АСЦ-ПДМК, ситуация оказалась прямо противоположной. Обнаружилось, что если в АСЦ-ПДМК использовалась функция ценности, выведенная из ОП-стратегии, то программа работала лучше, чем при использовании функции, выведенной из ОУ-стратегии.

Для достижения впечатляющего мастерства пришлось привлечь комбинацию нескольких методов. Команда DeepMind подвергла оценке разные версии AlphaGo, чтобы оценить вклад отдельных компонентов. Параметр η в формуле (16.4) определял долю каждого из двух способов оценки состояния игры: с помощью сети ценности и разыгрывания. При $\eta = 0$ AlphaGo использовала только сеть ценности без разыгрывания, а при $\eta = 1$ – только разыгрывание. Обнаружилось, что AlphaGo с одной лишь сетью ценности играла лучше, чем с одним лишь разыгрыванием, да и вообще лучше любой из программ, существовавших в то время. Наилучший результат был достигнут при $\eta = 0.5$, это доказывало, что сочетание сети ценности с разыгрыванием – важный фактор успеха AlphaGo. Эти методы оценивания дополняют друг друга: сеть ценности оценивает высококачественную ОУ-стратегию, которая, однако, слишком медленна для использования в реальных партиях, а стратегия разыгрывания более слабая, но зато и гораздо более быстрая, к тому же способная увеличить точность оценок сети ценности для особых состояний, встречающихся в игре.

В общем, поразительный успех AlphaGo стал причиной нового всплеска веры в обещания искусственного интеллекта, особенно в системы, сочетающие обучение с подкреплением с глубокими ИНС для решения проблем в других трудных и интересных областях.

16.6.2. AlphaGo Zero

Опираясь на опыт, полученный при работе над AlphaGo, команда DeepMind разработала программу AlphaGo Zero (Silver et al. 2017a). В отличие от AlphaGo, в этой программе вообще не использовались данные или инструкции, созданные с участием людей, сверх базовых правил игры (отсюда и слово Zero в названии). Она была обучена исключительно методами обучения с подкреплением на играх с собой, а в качестве входных данных подавались только описания расположения

камней на доске. В AlphaGo Zero была реализована форма итерации по стратегиям (раздел 4.3), в которой оценивание стратегии чередовалось с ее улучшением. На рис. 16.7 приведена общая схема алгоритма AlphaGo Zero. Существенное различие между AlphaGo Zero и AlphaGo заключалось в том, что в AlphaGo Zero ПДМК использовался для выбора ходов в процессе обучения с подкреплением на играх сама с собой, тогда как в AlphaGo – в реальной игре после, а не во время обучения. Есть и другие различия, помимо отказа от данных и признаков, созданных с участием человека: в AlphaGo Zero используется только одна глубокая сверточная ИНС и более простой вариант ПДМК.

ПДМК в AlphaGo Zero проще тем, что в нем нет разыгрывания полных партий, а потому не нужна стратегия разыгрывания. На каждой итерации ПДМК в AlphaGo Zero прогоняется имитация, заканчивающаяся в листовом узле текущего дерева поиска, а не в заключительной позиции имитации полной партии. Но, как и в AlphaGo, каждая итерация ПДМК в AlphaGo Zero направляется выходом глубокой сверточной сети, обозначенной f_θ на рис. 16.7, где θ – вектор весов сети. Входом в сеть, архитектуру которой мы опишем ниже, являются простые представления позиций на доске, а выход состоит из двух частей: скалярное значение v – оценка вероятности того, что текущий игрок выиграет, начав в текущей позиции, и вектор $\mathbf{p}$ вероятностей ходов, по одному элементу для каждого возможного размещения камня в текущей позиции плюс пас.

Однако в игре с собой AlphaGo Zero при выборе действий ориентировалась не только на вероятности $\mathbf{p}$, но и на скалярное значение на выходе сети – для управления каждым выполнением ПДМК, в результате которого новые вероятности ходов, показанные на рис. 16.7, возвращались в качестве стратегий π_i . В этих стратегиях учтены результаты многих имитаций, прогоняемых ПДМК при каждом выполнении. В итоге стратегия, которой следует AlphaGo Zero, является улучшением стратегии, определяемой только выходами сети $\mathbf{p}$. В работе Silver et al. (2017a) утверждается, что «тем самым ПДМК можно рассматривать как мощный оператор улучшения стратегии».

Опишем ИНС AlphaGo Zero и порядок ее обучения более подробно. На вход сети подавалась стопка изображений $19 \times 19 \times 17$, состоящая из 17 плоскостей бинарных признаков. Первые 8 плоскостей содержали простые представления расположения камней текущего игрока в текущей и семи предыдущих позициях: значение признака равно 1, если в соответствующем пункте находится камень игрока, и 0 в противном случае. В следующих 8 плоскостях точно так же кодировалось положений камней противника. Последняя плоскость признаков содержала постоянное значение, указывающее цвет камней текущего игрока: 1 – черный, 0 – белый. Поскольку в го запрещено повторение позиций, а одному игроку дается некоторое количество «компенсационных пунктов» за то, что он ходит не первым, то текущая позиция на доске не является марковским состоянием го. Именно поэтому нужны признаки, описывающие прошлые позиции на доске и цвет камней.

Сеть является «двухголовой», т. е. после нескольких начальных слоев сеть расщепляется на две отдельные «головы», состоящие из дополнительных слоев, которые по отдельности подаются на вход двух наборов выходных блоков. В данном случае одна голова «питала» 362 выходных блока, порождающих $19^2 + 1$ вероятностей ходов $\mathbf{p}$, по одному для каждого возможного размещения камня плюс пас, а другая голова «питала» только один выходной блок, порождающий скаляр v –

оценку вероятности того, что текущий игрок выиграет в текущей позиции. До расщепления сеть состояла из 41 сверточного слоя, за каждым из которых следовала пакетная нормировка, с прямыми связями, добавленными для реализации остаточного обучения с помощью пар слоев (см. раздел 9.7). В общем, вероятности и ценности ходов вычислялись 43 и 44 слоями соответственно.

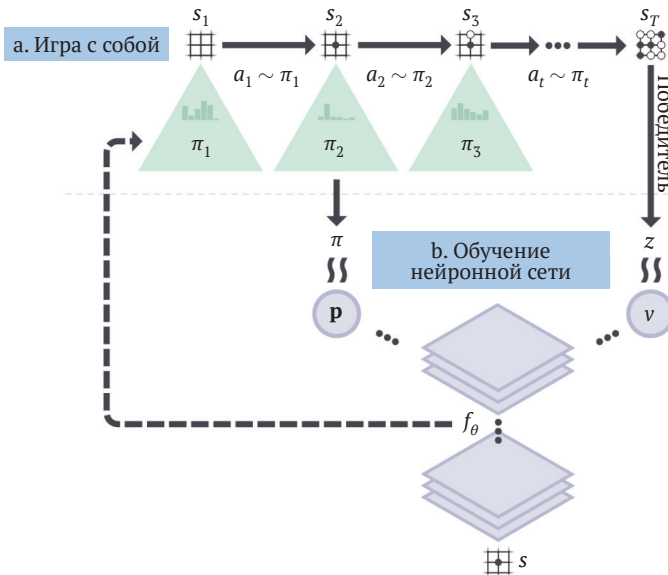


Рис. 16.7 ♠ Обучение с подкреплением AlphaGo Zero на играх с собой. а) Программа сыграла много партий с собой, одна из которых показана здесь в виде последовательности позиций на доске $s_i, i = 1, 2, \dots, T$, с ходами $a_i, i = 1, 2, \dots, T$, и победителем z . Каждый ход a_i определялся вероятностями действий π_i , возвращенными ПДМК, который выполнялся из корневого узла s_i под руководством глубокой сверточной сети (обозначена на рисунке f_θ) с последними вычисленными весами θ . Здесь это показано только для одной позиции s , но для всех s_i на вход сети подается простое представление позиции (и нескольких прошлых позиций, хотя здесь это не показано), а выходом является вектор $\mathbf{p}$ вероятностей ходов, который направляет прямой поиск ПДМК, и скалярное значение v , которое оценивает вероятность, что текущий игрок выиграет в позиции s_i . б) Обучение глубокой сверточной сети. Обучающими примерами были случайно выбранные шаги из недавно сыгранных с собой партий. Веса θ обновлялись с целью приблизить вектор стратегии $\mathbf{p}$ к вероятностям π , возвращенным ПДМК, и включить победителей z в оценку вероятности выигрыша v . Печатается по черновику работы Silver et al. (2017a) с разрешения авторов и компании DeepMind

Начав со случайных весов, сеть обучалась методом стохастического градиентного спуска (с импульсом, регуляризацией и убывающим размером шага) на пакетах примеров, равномерно выбираемых из всех шагов последних 500 000 игр с собой с текущей наилучшей стратегией. Чтобы поощрить исследование всех возможных шагов, к выходу сети $\mathbf{p}$ прибавлялся шум. В процессе обучения были контрольные точки – через каждую 1000 шагов, согласно работе Silver et al. (2017a), – в которых стратегия на выходе ИНС с последними вычисленными весами оценивалась по-

средством имитации 400 партий (для выбора каждого хода использовался ПДМК с 1600 итераций) против текущей наилучшей стратегии. Если новая стратегия одерживала победу (с некоторым запасом, чтобы снизить влияние шума на результат), то она становилась наилучшей стратегией для последующих игр с собой. Веса сети обновлялись с целью приблизить выходную стратегию $\mathbf{p}$ к стратегии, возвращенной ПДМК, а выходную ценность v – к вероятности, что текущая наилучшая стратегия выиграет в позиции, представленной входом сети.

Команда DeepMind обучила AlphaGo Zero на 4.9 млн игр с собой, что заняло 3 дня. Каждый ход в каждой игре выбирался путем прогона 1600 итераций ПДМК, что занимало приблизительно 0.4 секунды на ход. Веса сети обновлялись в результате обработки 700 000 пакетов, каждый из которых состоял из 2048 позиций. Затем проводились турниры с участием обученной AlphaGo Zero и той версии AlphaGo, которая победила Фан Ху со счетом 5:0, и версии, победившей Ли Седоля со счетом 4:1. Для оценивания сравнительного качества программ использовался рейтинг Эло. Предполагается, что разность между двумя рейтингами Эло предсказывает исход в партии между игроками. Рейтинги Эло программы AlphaGo Zero, версии AlphaGo, обыгравшей Фан Ху, и версии, обыгравшей Ли Седоля, были равны соответственно 4308, 3144 и 3739. При таких разностях рейтингов вероятность победы AlphaGo Zero во встречах с другими программами была близка к единице. В матче из 100 партий между AlphaGo Zero, обученной, как описано выше, и версией, обыгравшей Ли Седоля, при тех же условиях, что в том матче, AlphaGo Zero победила AlphaGo во всех 100 партиях.

Команда DeepMind также сравнила AlphaGo Zero с программой, в которой использовалась ИНС с такой же архитектурой, но обученная с учителем предсказывать ходы человека на наборе данных, содержащем приблизительно 30 млн позиций из 160 000 партий. Обнаружилось, что программа, обученная с учителем, поначалу играла лучше AlphaGo Zero и лучше предсказывала ходы мастеров, но стала показывать худшие результаты после обучения AlphaGo Zero в течение одного дня. Это навело на мысль, что AlphaGo Zero нашла стратегию игры, отличную от человеческой. И действительно, AlphaGo Zero отыскала и стала предпочитать новые вариации классических последовательностей ходов.

Окончательные тесты алгоритма AlphaGo Zero были проведены на версии, имевшей более глубокую ИНС и обученной более чем на 29 млн игр с собой, что заняло 40 дней. Как и раньше, веса были инициализированы случайным образом. Эта версия достигла рейтинга Эло 5185. Команда организовала встречу этой версии AlphaGo Zero с программой AlphaGo Master, сильнейшей на то время, которая совпадала с AlphaGo Zero, но, подобно AlphaGo, пользовалась данными и признаками, подготовленными с участием человека. Рейтинг Эло программы AlphaGo Master's составлял 4858, в онлайн-играх она обыгрывала сильнейших профессиональных игроков со счетом 60:0. В матче из 100 партий AlphaGo Zero с большей сетью и более продолжительным временем обучения обыграла AlphaGo Master со счетом 89:11, тем самым убедительно продемонстрировав эффективность алгоритма AlphaGo Zero при решении задач.

Программа AlphaGo Zero не оставила сомнений в том, что превзойти уровень человека можно с помощью чистого обучения с подкреплением, дополненного простым вариантом ПДМК и глубокой ИНС, имеющей минимальные знания о предметной области и не опирающейся на данные или руководство со сторо-

ны человека. Мы наверняка еще увидим, как системы, вдохновленные примером AlphaGo и AlphaGo Zero, применяются к трудным задачам в других областях.

Недавно в работе Silver et al. (2017b) была описана еще лучшая программа AlphaZero, которая даже не располагала информацией о правилах го. В AlphaZero применяется общий алгоритм обучения с подкреплением, который улучшает считавшиеся до сих пор лучшими программы для игры в го, шахматы и сёги.

16.7. ПЕРСОНАЛИЗИРОВАННЫЕ ВЕБ-СЛУЖБЫ

Персонализация таких веб-служб, как доставка новостей или рекламных объявлений, – один из подходов, призванных повысить удовлетворенность пользователей веб-сайтом или увеличить отдачу от маркетинговой кампании. Стратегия может рекомендовать, какой контент будет наилучшим для данного пользователя в зависимости от профиля его интересов и предпочтений, построенного на основе истории действий в сети. Это естественная область для применения машинного обучения и, в частности, обучения с подкреплением. Система обучения с подкреплением может улучшить рекомендательную стратегию, внося коррективы в ответ на отзывы пользователей. Один из способов получить отзыв – организовать исследование уровня удовлетворенности сайтом, но для получения обратной связи в режиме реального времени обычно анализируют поток кликов как индикаторов интереса пользователя к ссылке.

В маркетинге уже давно используется метод *A/B-тестирования* – простой тип обучения с подкреплением, позволяющий решить, какой из двух вариантов сайта, А или В, предпочитают пользователи. Будучи неассоциативным, как задача о двухруком бандите, этот подход не позволяет персонализировать доставку контента. Для персонализации службы необходимо добавить контекст, содержащий признаки, которые описывают отдельных пользователей и доставляемый контент. Это было формализовано в виде задачи о контекстуальном бандите (или задачи об ассоциативном обучении с подкреплением, см. раздел 2.9), в которой целевая функция – максимизировать общее количество кликов пользователем. В работе Li, Chu, Langford, and Schapire (2010) алгоритм контекстуального бандита применен к задаче о персонализации веб-страницы Yahoo! Front Page Today (одна из самых посещаемых страниц интернета на момент исследования) с целью выбрать заглавную новость. Целевой функцией была максимизация коэффициента кликабельности (click-through rate – CTR), равного отношению числа перешедших по ссылке к общему числу видевших ее. Предложенный авторами алгоритм контекстуального бандита был лучше стандартного неассоциативного алгоритма бандита на 12.5 %.

В работе Theocharous, Thomas, and Ghavamzadeh (2015) приведены аргументы в пользу того, что можно добиться лучших результатов, сформулировав задачу о персонализированных рекомендациях как задачу о марковском процессе принятия решений (МППР) с целевой функцией максимизации общего числа кликов пользователей при повторных посещениях сайта. Стратегии, вытекающие из формулировки в терминах контекстуального бандита, являются жадными в том смысле, что не принимают в расчет отдаленных последствий действий. По существу, в этих стратегиях каждое посещение сайта рассматривается так, будто

его совершил новый пользователь, равномерно выбираемый из генеральной совокупности посетителей сайта. Не учитывая тот факт, что многие пользователи несколько раз заходят на одни и те же сайты, жадные стратегии не могут воспользоваться возможностями, открывающимися в результате длительного взаимодействия с отдельными пользователями.

В качестве примера того, как в маркетинге можно воспользоваться длительным взаимодействием с пользователем, Теочарус с коллегами сравнил жадную стратегию с долговременной стратегией отображения объявлений, рекламирующих некоторый товар, например автомобиль. В рекламе, отображаемой жадной стратегией, может содержаться предложение о скидке, если пользователь купит автомобиль немедленно. Пользователь либо принимает предложение, либо уходит с сайта, и если когда-нибудь вернется, то, вероятно, увидит то же самое предложение. С другой стороны, долговременная стратегия может сначала затянуть пользователя в «воронку продаж», а только потом представить конечное идеальное предложение. Можно начать с описания выгодных финансовых условий, затем расхвалить прекрасный отдел обслуживания, а при следующем визите предложить окончательную скидку. При такой стратегии пользователь, возможно, совершит больше кликов при повторных визитах на сайт, и если стратегия хорошо продумана, то объем продаж увеличится.

Во время работы в компании Adobe Systems Incorporated Теочарус с коллегами поставили ряд экспериментов, чтобы узнать, действительно ли стратегии, спроектированные для максимизации числа кликов в длительной перспективе, лучше краткосрочных жадных стратегий. Набор инструментов Adobe Marketing Cloud, которым пользуются многие компании для проведения цифровых маркетинговых кампаний, предоставляет инфраструктуру для автоматизации ориентированных на пользователя рекламных кампаний и кампаний по сбору средств. На самом деле внедрение новых стратегий с помощью этих инструментов рискованно, потому что результаты могут оказаться плачевными. Поэтому исследователям необходимо оценить, что получилось бы при внедрении новой стратегии, но сделать это на основе данных, собранных в результате применения других стратегий. Таким образом, ключевым аспектом этой работы было оценивание с разделенной стратегией. Кроме того, команда хотела получить высокую степень достоверности, чтобы уменьшить риск внедрения новой стратегии. Хотя главной задачей исследования было достоверное оценивание с разделенной стратегией (см. также Thomas, 2015; Thomas, Theocharous, and Ghavamzadeh, 2015), здесь мы остановимся только на алгоритмах и их результатах.

Авторы сравнили результаты двух алгоритмов обучения и рекомендации. Первый алгоритм, названный ими *жадной оптимизацией*, стремился максимизировать только вероятность немедленного клика. Как и в стандартной постановке задачи о контекстуальном бандите, этот алгоритм не учитывал отдаленных последствий рекомендаций. Целью другого алгоритма – обучения с подкреплением – основанного на МППР, было увеличение количества кликов при многократном посещении сайта пользователем. Этот алгоритм авторы называли *оптимизацией пожизненной ценности* (life-time value – LTV). Перед обоими алгоритмами стояли трудные проблемы, потому что сигнал вознаграждения в этой предметной области очень разрежен, т. к. пользователи обычно не кликают по рекламным объявлениям, а это значит, что клик – случайное событие и дисперсия дохода очень высока.

Для обучения и тестирования алгоритмов использовались наборы данных из банковской отрасли. Каждый набор содержал много полных траекторий взаимодействия пользователя с сайтом банка, в которых пользователю демонстрировалось одно из нескольких возможных предложений. Если пользователь кликал по нему, то начислялось вознаграждение 1, иначе 0. Один набор данных содержал приблизительно 200 000 взаимодействий за месяц проведения банком кампании, в которых случайным образом демонстрировалось одно из семи предложений. Другой набор данных относился к кампании другого банка и содержал 4 000 000 взаимодействий с демонстрацией 12 возможных предложений. В состав данных о каждом взаимодействии включались такие признаки пользователя, как время с момента последнего посещения сайта, количество посещений, время последнего клика, географическое местоположение, что-то типа набора интересов и демографическая информация.

Жадная оптимизация основывалась на вычислении оценки вероятности клика как функции от признаков пользователя. Для этого использовалось обучение с учителем по одному из наборов данных с помощью алгоритма случайного леса (random forest – RF) (Breiman, 2001). RF-алгоритмы широко применяются в крупномасштабных приложениях, поскольку являются эффективными средствами предсказания, не склонны к переобучению и относительно нечувствительны к выбросам и шуму. Затем Теочарус с коллегами использовали обученное отображение для определения ε -жадной стратегии, с вероятностью $1 - \varepsilon$ выбиравшей предложение, для которого RF-алгоритм предсказывал наибольшую вероятность клика, и с вероятностью ε – одно из других предложений (случайным образом с равномерным распределением).

Для LTV-оптимизации использовался пакетный алгоритм обучения с подкреплением, названный *подобранной Q-итерацией* (fitted Q iteration – FQI). Это вариант алгоритма *подобранной итерации по ценности* (Gordon, 1999), адаптированный к Q-обучению. Пакетный режим означает, что для обучения с самого начала доступен весь набор данных – в противоположность онлайновому режиму, в котором данные поступают последовательно в процессе работы алгоритма обучения (именно такие алгоритмы мы в основном рассматривали в этой книге). Пакетные алгоритмы обучения с подкреплением иногда необходимы, если онлайн-обучение практически нереализуемо. При этом можно использовать любой пакетный алгоритм регрессии, основанный на обучении с учителем, в т. ч. алгоритмы, которые хорошо масштабируются на пространства высокой размерности. Сходимость FQI зависит от свойств алгоритма аппроксимации функций (Gordon, 1999). В применении к LTV-оптимизации Теочарус с коллегами использовали тот же RF-алгоритм, что и для жадной оптимизации. Поскольку в этом случае сходимость FQI не монотонна, авторы запоминали наилучшую FQI-стратегию посредством оценивания с разделенной стратегией на контрольном наборе данных. Окончательной стратегией для тестирования LTV-подхода была ε -жадная стратегия, основанная на лучшей стратегии, найденной FQI, когда в качестве начальной функции ценности действий бралось отображение, вычисленное RF-алгоритмом в подходе на базе жадной оптимизации.

Для измерения эффективности стратегий, порожденных обоими подходами – жадным и LTV, Теочарус с коллегами использовали метрику CTR и метрику, которую называли LTV. Эти метрики различаются тем, что LTV учитывает только уникальных посетителей:

$$CTR = \frac{\text{Всего кликов}}{\text{Всего посещений}};$$

$$LTV = \frac{\text{Всего кликов}}{\text{Всего посетителей}}.$$

На рис. 16.8 показано, в чем различие между этими метриками. Каждый кружок представляет одно посещение сайта пользователем; черные кружки соответствуют посещениям, закончившимся кликом. В каждой строке представлены посещения одного пользователя. При вычислении CTR все посетители считаются различными, поэтому для данной последовательности $CTR = 0.35$, тогда как $LTV = 1.5$. Поскольку LTV превышает CTR тем больше, чем чаще одни и те же пользователи заходят на сайт, ее можно считать показателем успешности стратегии с точки зрения вовлечения пользователей в продолжительные взаимодействия с сайтом.

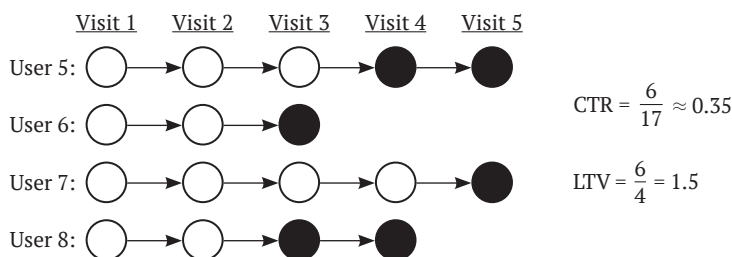


Рис. 16.8 ❖ Коэффициент кликабельности (CTR) и пожизненная ценность (LTV). Каждый кружок представляет одно посещение пользователем; черные кружки обозначают посещения, закончившиеся кликом. Заимствовано из работы Theocharous et al. (2015)

Тестирование обеих стратегий производилось с помощью обладающего высокой достоверностью метода оценивания с разделенной стратегией на тестовом наборе данных, содержащем реальные взаимодействия с сайтом банка при случайной стратегии. Как и ожидалось, жадная оптимизация оказалась лучшей с точки зрения метрики CTR, а LTV-оптимизация – лучшей с точки зрения метрики LTV. Дополнительно (эти детали мы опускаем) метод оценивания с разделенной стратегией давал вероятностные гарантии, что LTV-оптимизация с высокой вероятностью выберет стратегии, более качественные, чем внедренные в данный метод. Воодушевленная этими гарантиями, Adobe в 2016 году объявила, что новый LTV-алгоритм станет стандартным компонентом Adobe Marketing Cloud, чтобы торговые предприятия могли выдавать последовательности предложений, следуя стратегии, которая с большой вероятностью принесет более высокий доход, чем стратегия, не принимающая во внимание отдаленных последствий.

16.8. ПАРЕНИЕ В ВОСХОДЯЩИХ ПОТОКАХ ВОЗДУХА

Птицы и планеры пользуются восходящими потоками воздуха, чтобы набрать высоту, а затем продолжать полет, расходуя мало или вообще не расходуя энергии. Такое парение в восходящих потоках воздуха – это сложный навык, требующий

реакции на небольшие изменения окружающей среды с целью продолжать набор высоты, пока восходящий поток не иссяк. В работе Reddy, Celani, Sejnowski, and Vergassola (2016) обучение с подкреплением используется для исследования стратегий парения, эффективных в условиях сильной атмосферной турбулентности, обычно сопровождающей восходящие потоки воздуха. Их основной целью было понять, какие ориентиры воспринимают птицы и как они пользуются ими для достижения впечатляющих результатов, однако полученные результаты полезны также для технологий производства автономных планеров. Ранее обучение с подкреплением применялось для решения задачи об эффективной навигации вблизи столба горячих газов над местом пожара (Woodbury, Dunn, and Valasek, 2014), но не более трудной задачи парения внутри самого турбулентного восходящего потока.

Редди с коллегами построили модель парения в виде непрерывного МППР с обесцениванием. Агент взаимодействовал с детальной моделью планера, летящего в турбулентном потоке воздуха. Было приложено много усилий, чтобы модель генерировала реалистичные условия парения в восходящем потоке воздуха, в частности исследовались различные подходы к моделированию атмосферы. Для экспериментов по обучению был смоделирован воздушный поток в трехмерном кубе с ребром 1 км, одна грань которого находилась на уровне земли. Для этого использовалась сложная система дифференциальных уравнений в частных производных, включающая скорость, температуру и давление. Внесение небольших возмущений в процессе численного моделирования заставляло модели порождать аналоги восходящих потоков и сопутствующую им турбулентность (рис. 16.9 слева). Полет планера моделировался с помощью уравнений аэродинамики, включающих скорость, подъемную силу, лобовое сопротивление воздуха и другие факторы, описывающие полет летательного аппарата с неизменяемой геометрией крыла с неработающим двигателем. Для маневрирования планер должен был изменять угол атаки (угол между плоскостью крыла планера и направлением потока воздуха) и угол крена (рис. 16.9 справа).

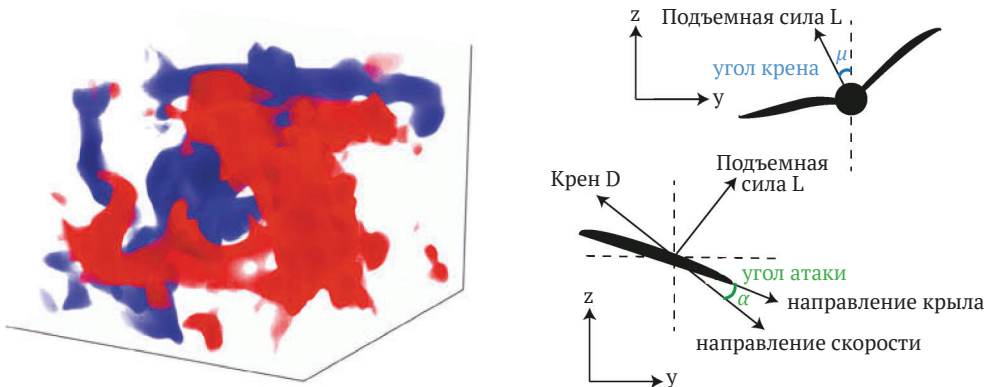


Рис. 16.9 ❖ Модель парения в восходящих потоках воздуха. **Слева:** мгновенный снимок поля вертикальной составляющей скорости в моделируемом воздушном кубе: красным (синим) цветом обозначена область сильного нисходящего (восходящего) потока. **Справа:** параметры полета с неработающим двигателем. Показаны угол крена μ и угол атаки α . Взято с разрешения авторов из статьи Reddy, Celani, Sejnowski, and Vergassola «Learning to Soar in Turbulent Environments», PNAS vol. 113(22), p. E4879, 2016

Для описания интерфейса между агентом и окружающей средой необходимо было определить действия агента, информацию о состоянии, которую агент получает от среды, и сигнал вознаграждения. В результате экспериментов было принято решение, что достаточно трех действий отдельно для угла атаки и угла крена: увеличить или уменьшить текущий угол крена или угол атаки на 5° , на 2.5° или оставить угол без изменения. Всего получалось 32 возможных действия. Допустимый диапазон изменения угла крена составлял от -15° до $+15^\circ$.

Целью исследования было установить, какой минимальный набор сенсорных ориентиров необходим для эффективного парения – и для того чтобы пролить свет на поведение птиц, и для того чтобы минимизировать сложность датчиков, требующихся для автоматизированного управления планером. Авторы испытывали различные множества сигналов, подаваемых на вход агента обучения с подкреплением. Начали они с агрегирования состояний (раздел 9.3) в четырехмерном пространстве состояний со следующими измерениями: локальная вертикальная скорость ветра, локальное вертикальное ускорение ветра, крутящий момент, зависящий от разности между вертикальными составляющими скорости ветра на концах левого и правого крыльев, и локальная температура. Для каждого измерения было определено три интервала дискретизации: большое положительное, большое отрицательное и малое. Описанные ниже результаты показывали, что только два из этих измерений существенны для эффективного парения.

Основная цель парения в восходящих потоках воздуха – набрать как можно большую высоту в каждом пойманном восходящем потоке. Авторы попробовали прямолинейный сигнал вознаграждения – вознаграждать агента в конце каждого эпизода в зависимости от набранной в этом эпизоде высоты. При этом если планер коснулся земли, то начислялось большое отрицательное вознаграждение, а в противном случае нулевое. Выяснилось, что при таком сигнале вознаграждения обучение не приносит успеха в эпизодах с реалистичной длительностью, и следы приемлемости не помогают. Экспериментируя с другими сигналами, авторы обнаружили, что наилучшие результаты получаются, когда сигнал вознаграждения на каждом временном шаге представляет собой линейную комбинацию вертикальной скорости и вертикального ускорения ветра, наблюдавшихся на предыдущем шаге.

Для обучения использовался одношаговый алгоритм Sarsa, а действия выбирались из распределения softmax, основанного на нормированных ценностях действий. Точнее, вероятности действий вычислялись по формуле (13.2) с такими предпочтениями действий:

$$h(s, a, \theta) = \frac{\hat{q}(s, a, \theta) - \min_b \hat{q}(s, b, \theta)}{\tau(\max_b \hat{q}(s, b, \theta) - \min_b \hat{q}(s, b, \theta))},$$

где θ – вектор параметров, содержащий по одному элементу для каждого действия и агрегированной группы состояний, а функция $\hat{q}(s, a, \theta)$ просто возвращает элемент, соответствующий s, a , обычным для методов агрегирования состояний способом. В приведенной выше формуле предпочтения действий формируются путем нормирования приближенных ценностей действий на интервал $[0, 1]$ и последующего деления на τ – положительный «температурный параметр»¹. При увеличении

¹ В работе Reddy et al. это описывается несколько иначе, но наш вариант эквивалентен.

τ вероятность выбора действия в меньшей степени зависит от его предпочтения, а когда τ стремится к нулю, вероятность выбора самого предпочтительного действия стремится к единице, т. е. стратегия приближается к ϵ -жадной. Температурный параметр τ был инициализирован значением 2.0 и постепенно уменьшался до 0.2 в процессе обучения. Предпочтения действий вычислялись по текущим оценкам ценностей действий. Действию с максимальной оценкой ценности назначается предпочтение $1/\tau$, действию с минимальной оценкой – предпочтение 0, а предпочтения остальных действий пропорционально масштабировались между этими крайними значениями. Размер шага и коэффициент обесценивания были равны соответственно 0.1 и 0.98.

В каждом эпизоде обучения агент управлял имитированным полетом в независимо сгенерированном периоде имитированных турбулентных потоков воздуха. Каждый эпизод продолжался 2.5 минуты с шагом 1 секунда. Обучение сходилось после нескольких сотен эпизодов. На рис. 16.10 слева показана выборочная траектория до обучения, когда агент выбирает действия случайным образом. Начав полет в верхней точке куба, планер движется в направлении стрелки и быстро теряет высоту. На рис. 16.10 справа показана траектория после обучения. Планер начинает полет в той же точке (теперь она находится в нижней части куба) и набирает высоту, двигаясь по спирали внутри восходящего потока воздуха. Хотя, как выяснили авторы, результаты сильно зависят от конкретного имитированного периода, количество касаний земли планером устойчиво уменьшалось по мере продолжения обучения.

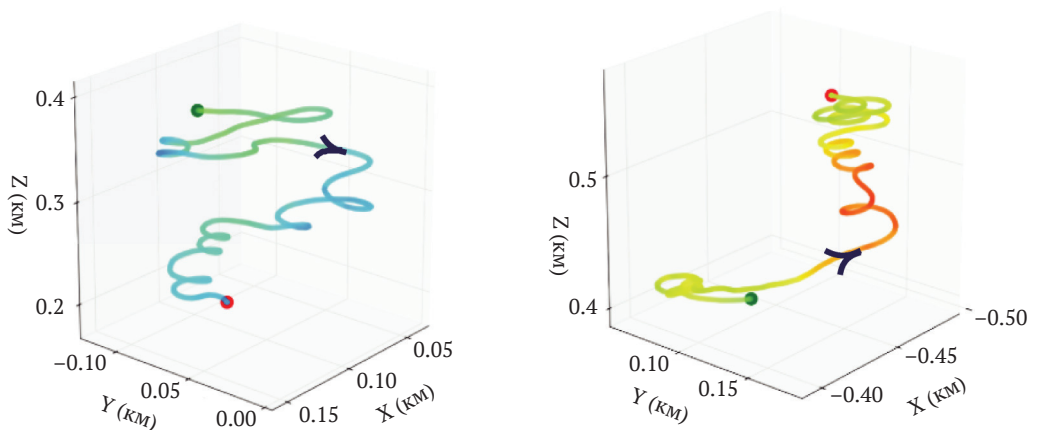


Рис. 16.10 ❖ Выборочные траектории планирования в восходящих потоках воздуха. Стрелки показывают направление полета из одной и той же начальной точки (обратите внимание, что масштаб высоты изменен). **Слева:** до обучения: агент выбирает действия случайным образом, и планер снижается. **Справа:** после обучения: планер набирает высоту, двигаясь по спиральной траектории. Взято с разрешения авторов из статьи Reddy, Celani, Sejnowski, and Vergassola «Learning to Soar in Turbulent Environments», PNAS vol. 113(22), p. E4879, 2016

После экспериментов с различными наборами признаков, доступных агенту обучения, оказалось, что наилучшие результаты получаются в случае комбинации вертикального ускорения ветра и крутящего момента. Авторы пришли к вы-

воду, что коль скоро эти признаки дают информацию о градиенте вертикальной скорости ветра в двух направлениях, они позволяют контроллеру выбирать одно из двух действий: повернуть, изменив угол крена, или продолжать движение тем же курсом, оставив угол крена без изменения. Это позволяет планеру оставаться внутри восходящего потока воздуха. Вертикальная скорость ветра указывает на силу потока, но не помогает оставаться внутри него. Выяснилось, что температура тоже мало на что влияет. Также оказалось, что управление углом атаки не помогает оставаться внутри конкретного воздушного потока, но полезно при перемещении от одного потока к другому при дальних полетах, например планировании на расстояние или миграции птиц.

Из-за того, что парение при разных уровнях турбулентности требует различных стратегий, обучение производилось в условиях, варьирующихся от слабой до сильной турбулентности. При сильной турбулентности быстрота изменения скоростей ветра и планера оставляла контроллеру мало времени для реакции. Это уменьшало возможности управления по сравнению с маневрированием в условиях слабых флуктуаций. Редди с коллегами проанализировали стратегии, получающиеся в результате обучения алгоритмом Sarsa при различных условиях. Общими для стратегий, обученных во всех режимах, были следующие особенности: обнаружив отрицательное ускорение ветра, резко накренившись на крыло с большей подъемной силой; обнаружив большое положительное ускорение и отсутствие крутящего момента, не делать ничего. Однако различие в уровне турбулентности влекло за собой различия в стратегиях. Стратегии, обученные при сильной турбулентности, были более консервативны, т. е. предпочитали небольшое изменение угла крена, тогда как при слабой турбулентности наилучшим действием считался как можно более крутой поворот за счет резкого изменения угла крена. Систематическое изучение углов крена, предпочитаемых стратегиями, обученными при разных условиях, привело авторов к выводу, что контроллер может корректировать стратегию в зависимости от режима турбулентности, если будет определять момент, когда вертикальное ускорение ветра пересекает некоторый порог.

Редди с коллегами также провели эксперименты по исследованию того, как параметр обесценивания γ влияет на качество обученных стратегий. Обнаружилось, что набранная в эпизоде высота увеличивается с ростом γ , достигая максимума при $\gamma = 0.99$, и это наводит на мысль, что для эффективного парения в восходящих потоках воздуха нужно принимать во внимание отдаленные последствия управляющих решений.

Это вычислительное исследование парения в восходящих потоках воздуха показывает, что благодаря обучению с подкреплением можно добиться прогресса, ставя перед собой различные цели. Стратегии обучения, имеющие доступ к различным наборам внешних ориентиров и управляющих действий, вносят вклад как в чисто инженерную задачу – конструирование автономных планеров, так и в научную задачу – лучше понять навыки парения у птиц. В обоих случаях гипотезы, выдвинутые на основе экспериментов по обучению, можно проверить на практике: конструируя настоящие планеры или сравнивая предсказания с наблюдаемым поведением птиц, парящих в восходящих потоках.

Глава 17

Передовые рубежи

В этой, последней главе мы коснемся некоторых тем, которые выходят за рамки книги, но кажутся нам особенно важными для будущего обучения с подкреплением. Многие из них выводят за пределы надежных знаний, а некоторые – и за пределы МППР вообще.

17.1. ОБЩИЕ ФУНКЦИИ ЦЕННОСТИ И ВСПОМОГАТЕЛЬНЫЕ ЗАДАЧИ

На протяжении книги мы выработали весьма общую нотацию для функций ценности. При обучении с разделенной стратегией мы допускали, что функция ценности обусловлена произвольной целевой стратегией. Затем в разделе 12.8 мы обобщили обесценивание на *функцию завершения* $\gamma: \mathcal{S} \mapsto [0, 1]$, так чтобы на каждом шаге определения дохода (12.17) можно было применять разные коэффициенты обесценивания. Это позволило нам выражать предсказания того, какое вознаграждение мы получим на произвольном горизонте, зависящем от состояния. Следующий и, наверное, последний шаг – обобщить вознаграждения, так чтобы была возможность предсказывать произвольные сигналы. Вместо предсказания суммы будущих вознаграждений мы могли бы предсказывать сумму будущих значений слуховых или цветовых ощущений или внутренних полученных в результате сложной обработки сигналов, например другого предсказания. Не важно, какие сигналы складываются подобным образом в предсказании чего-то вроде функции ценности, результат мы называем *кумулянт* этого предсказания. Это формализуется в понятии *кумулянтного сигнала* $C_t \in \mathbb{R}$. Тогда можно записать *общую функцию ценности* (ОФЦ) в виде:

$$v_{\pi, \gamma, C}(s) = \mathbb{E} \left[\sum_{k=t}^{\infty} \left(\prod_{i=t+1}^k \gamma(S_i) \right) C_{k+1} \middle| S_t = s, A_{t:\infty} \sim \pi \right]. \quad (17.1)$$

Как и традиционные функции ценности (v_{π} или q_{*}), это идеальная функция, которую мы хотим аппроксимировать параметрической формой; ее можно было бы и дальше обозначать $\hat{v}(s, \mathbf{w})$, хотя, конечно, $\mathbf{w}$ будет своим для каждого предсказания, т. е. для каждого выбора π , γ и C_t . Поскольку ОФЦ может не иметь никакой связи с вознаграждением, наверное, не стоило бы называть ее функцией *ценности*. Можно было бы назвать ее просто предсказанием или, если мы хотим различать

понятия, *прогнозом* (Ring, готовится к публикации). Но как бы ни называть, она имеет вид функции ценности и потому может быть обучена, как обычно, с применением разработанных в этой книге методов обучения приближенных функций ценности. Наряду с обученными предсказаниями мы могли обучать также стратегии максимизации этих предсказаний – с помощью обобщенной итерации по стратегиям (раздел 4.6) или методами исполнитель–критик. Таким образом, агент мог бы обучиться предсказанию и управлению большим количеством сигналов, а не только долгосрочным вознаграждением.

Зачем могло бы понадобиться предсказание и управление другими сигналами, помимо долгосрочного вознаграждения? Существуют *вспомогательные* задачи, дополняющие главную задачу максимизации вознаграждения. Один из ответов на поставленный выше вопрос заключается в том, что умение предсказывать и управлять широким спектром сигналов может привести к весьма полезному способу моделирования окружающей среды. В главе 8 мы видели, что хорошая модель иногда позволяет агенту получать вознаграждение эффективнее. Для доведения этого ответа до логического конца понадобится еще несколько понятий, поэтому мы отложим его до следующего раздела. А сначала рассмотрим, как еще наличие нескольких разнородных предсказаний может оказаться полезным агенту обучения с подкреплением.

Вспомогательные задачи могут оказать содействие главной тем, что нуждаются в тех же представлениях (или части из них), которые нужны в главной задаче. Некоторые вспомогательные задачи могут быть проще, т. е. характеризуются меньшей задержкой и имеют более понятную связь между действиями и исходами. Если в простых вспомогательных задачах можно на ранней стадии найти хорошие признаки, то эти признаки могут значительно ускорить обучение в главной задаче. Это вовсе необязательно так, но во многих случаях выглядит правдоподобно. Например, если обучиться предсказывать и управлять датчиками на коротких промежутках времени, скажем порядка секунд, то, возможно, появится идея неких объектов, которые впоследствии помогут предсказывать и управлять долгосрочным вознаграждением.

Можно представить себе искусственную нейронную сеть (ИНС), в которой последний слой расщеплен на несколько частей, или *голов*, работающих над разными задачами. Одна голова может порождать приближенную функцию ценности для главной задачи (кумулянтom тогда является вознаграждение), а другие – решения различных вспомогательных задач. Все головы могли бы распространять ошибки методом стохастического градиентного спуска одному и тому же телу – общей предшествующей части сети, – которое тогда попыталось бы сформировать в своем предпоследнем слое представления, поддерживающие все головы. Исследователи экспериментировали с такими вспомогательными задачами, как предсказание изменения пикселей, вознаграждения на следующем шаге и распределения дохода. Во многих случаях этот подход позволил значительно ускорить обучение в главной задаче (Jaderberg et al., 2017). Аналогично вычисление нескольких предсказаний неоднократно предлагалось как способ направить конструирование оценок состояний (см. раздел 17.3).

Еще один простой способ использовать обучение вспомогательных задач для улучшения общих результатов лучше всего пояснить, проведя аналогию с классическим обусловливанием в психологии (раздел 14.2). Одна из интерпретаций

классического обусловливания состоит в том, что эволюция вмонтировала рефлекторную (не требующую обучения) ассоциацию с определенным действием, если предсказан определенный сигнал. Например, у человека и многих животных, похоже, имеется рефлекс моргания, когда предсказание попадания постороннего предмета в глаз превышает определенный порог. Предсказание поддается обучению, но ассоциация предсказания с закрыванием глаза встроена, поэтому животное защищено от неожиданного попадания чего-то в глаз. Аналогично встроеной может быть ассоциация страха с учащенным сердцебиением или с оцепенением. Проектировщики агентов могут сделать нечто подобное, организовав внутреннюю (не требующую обучения) связь предсказаний некоторых событий с предопределенными действиями. Например, беспилотный автомобиль, обучающийся предсказывать, приведет ли движение вперед к столкновению, можно снабдить встроеным рефлексом останавливаться или отворачивать, когда предсказание превышает некоторый порог. Или возьмем робот-пылесос, который обучился предсказывать возможность разрядки батарей, прежде чем он сможет добраться до зарядного устройства. Он мог бы рефлекторно возвращаться к зарядному устройству, когда предсказание становится ненулевым. Правильное предсказание должно было бы зависеть от размера дома, от того, в какой комнате находится робот, и от возраста батареи – все это конструктор робота знать не может. Конструктору было бы трудно создать надежный алгоритм, который решал бы, когда возвращаться к зарядному устройству, в терминах показаний датчиков, но это легко сделать в терминах обученного предсказания. Мы предвидим много подобных ситуаций, когда обученные предсказания можно с пользой объединить со встроеными алгоритмами для управления поведением.

И пожалуй, самая важная роль вспомогательных задач – они позволяют отказаться от принятого в этой книге предположения о том, что представление состояния фиксировано и передается агенту. Чтобы разобраться в этом, нам придется отступить на несколько шагов назад – только так можно оценить значительность этого предположения и последствия отказа от него. Мы сделаем это в разделе 17.3.

17.2. АБСТРАГИРОВАНИЕ ВРЕМЕНИ ПОСРЕДСТВОМ ОПЦИЙ

Формализм МППР привлекателен еще и тем, что его можно с пользой применять к задачам с разной временной шкалой, например для формализации задач о принятии решения о том, какие мускулы сократить для захвата предмета, на какой рейс купить билет, чтобы прибыть в далекий город в удобное время, каким делом заняться, чтобы получать от жизни удовлетворение. Эти задачи сильно разнятся с точки зрения временной шкалы, но каждую можно сформулировать как МППР и решать с помощью процессов планирования или обучения, описанных в этой книге. Все они подразумевают взаимодействие с миром, последовательное принятие решений, а цель можно описать в терминах накопления вознаграждения со временем. А раз так, то их можно поставить как МППР.

Но хотя все эти задачи формулируются как МППР, вы вряд ли рискнете предположить, что это *один и тот же* МППР. Ведь такой разный масштаб времени, такие разные понятия о выборе и действиях! Например, не стоит, наверное, планировать полет через весь континент на том же уровне, что сокращения мускулов. Од-

нако для других задач – захвата, метания стрел в дартс или удара по бейсбольному мячу – низкоуровневое сокращение мускулов вполне может оказаться подходящим уровнем. Люди естественно делают все это, не переключая уровни. Можно ли расширить каркас МППР, так чтобы он охватывал все уровни одновременно?

Быть может. Одна из популярных идей состоит в том, чтобы формализовать МППР на детальном уровне, с небольшим временным шагом, разрешив в то же время планирование на более высоких уровнях с применением расширенного образа действий, соответствующего многим временным шагам базового уровня. Для этого мы должны определить понятия *образа действий*, которое охватывало бы много временных шагов и включало понятие завершения. Общий способ формализовать эти две идеи – ввести в рассмотрение стратегию π и зависящую от состояния функцию завершения γ , как в ОФЦ. Мы определим обобщенное понятие действия, которое назовем *опцией*, как пару, состоящую из стратегии и функции завершения. Выполнить опцию $\omega = (\pi_\omega, \gamma_\omega)$ в момент t означает выбрать действие A_t из $\pi_\omega(\cdot | S_t)$ и завершить процесс в момент $t + 1$ с вероятностью $1 - \gamma_\omega(S_{t+1})$. Если опция не завершилась в момент $t + 1$, то выбирается действие A_{t+1} из $\pi_\omega(\cdot | S_{t+1})$, и опция завершается в момент $t + 2$ с вероятностью $1 - \gamma_\omega(S_{t+2})$, и так далее, пока процесс наконец не завершится. Удобно рассматривать низкоуровневые действия как частные случаи опций – каждое действие a соответствует опции $(\pi_\omega, \gamma_\omega)$, стратегия которой выбирает действие a ($\pi_\omega(s) = a$ для всех $s \in \mathcal{S}$), а функция завершения равна нулю ($\gamma_\omega(s) = 0$ для всех $s \in \mathcal{S}^+$). По существу, опции расширяют пространство состояний. Агент может выбрать либо низкоуровневое действие (опцию), завершающееся после одного временного шага, либо продленную опцию, которая может выполняться в течение многих шагов до завершения.

Опции проектируются, так чтобы они были взаимозаменяемы с низкоуровневыми действиями. Например, понятие функции ценности действий q_π естественно обобщается на функцию ценности опций, которая принимает на входе состояние и опцию, а возвращает математическое ожидание дохода, который будет получен, если начать из этого состояния, выполнить опцию до завершения и далее следовать стратегии π . Можно также обобщить понятие стратегии на иерархическую стратегию, которая осуществляет выбор из множества опций, а не действий, и выбранная опция выполняется до завершения. В русле этих идей многие описанные в книге алгоритмы можно обобщить, так что они будут обучать приближенные функции ценности опций и иерархические стратегии. В простейшем случае процесс обучения «перепрыгивает» от инициализации опции к ее завершению, и обновление производится только при завершении опции. Но можно производить обновления и на каждом временном шаге, применяя «внутриопционные» алгоритмы обучения, для чего, вообще говоря, требуется обучение с разделенной стратегией.

Быть может, самое важное обобщение, ставшее возможным благодаря идее опций, – модель окружающей среды, разработанная в главах 3, 4 и 8. Традиционная модель действий включает вероятности перехода состояний и ожидаемое немедленное вознаграждение в результате выполнения действия в каждом состоянии. Как обобщить традиционную модель действий на *модели опций*? Для опций модель снова состоит из двух частей: одна соответствует переходу состояний в результате выполнения опции, а другая – ожидаемому накопленному вознаграждению вдоль пути. Часть модели опций, связанная с вознаграждением,

по аналогии с ожидаемым вознаграждением для пар состояние–действие (3.5) описывается равенством

$$r(s, \omega) \doteq \mathbb{E}[R_1 + \gamma R_2 + \gamma^2 R_3 + \dots + \gamma^{\tau-1} R_\tau | S_0 = s, A_{0:\tau-1} \sim \pi_\omega, \tau \sim \gamma_\omega] \quad (17.2)$$

для всех опций ω и всех состояний $s \in \mathcal{S}$, где τ – случайный временной шаг, на котором опция завершается согласно функции γ_ω . Отметим роль общего параметра обесценивания γ в этом равенстве – обесценивание производится с коэффициентом γ , а завершение опции – согласно функции γ_ω . Часть модели опций, связанная с переходом состояний, немного сложнее. Эта часть характеризует вероятность каждого возможного результирующего состояния (как в (3.4)), но теперь состояние может наступить после разного числа временных шагов, на каждом из которых должно применяться разное обесценивание. Для каждого состояния s , в котором опция ω может начинаться, и каждого состояния s' , в котором ω может завершаться, модель ω определяет величину

$$p(s'|s, \omega) \doteq \sum_{k=1}^{\infty} \gamma^k \Pr\{S_k = s', \tau = k | S_0 = s, A_{0:k-1} \sim \pi_\omega, \tau \sim \gamma_\omega\}. \quad (17.3)$$

Отметим, что из-за множителя γ^k величина $p(s'|s, \omega)$ уже не является вероятностью перехода, и сумма этих величин по всем значениям s' не равна 1. (Тем не менее мы продолжим использовать нотацию «|» в выражении для p .)

Приведенное определение части модели опций, относящейся к переходам, позволяет нам выписать уравнения Беллмана и сформулировать алгоритмы динамического программирования, применимые ко всем опциям, в т. ч. и низкоуровневым действиям, рассматриваемым как частный случай. Например, общее уравнение Беллмана для ценностей состояний иерархической стратегии π имеет вид:

$$h(s, a, \theta) = \frac{\hat{q}(s, a, \theta) - \min_b \hat{q}(s, b, \theta)}{\tau(\max_b \hat{q}(s, b, \theta) - \min_b \hat{q}(s, b, \theta))}, \quad (17.4)$$

где $\Omega(s)$ обозначает множество опций, доступных в состоянии s . Если $\Omega(s)$ включает только низкоуровневые действия, то это уравнение сводится к варианту обычного уравнения Беллмана (3.14) с тем отличием, что γ включено в новое определение p (17.3) и потому не встречается. В соответствующие алгоритмы планирования γ тоже не входит. Например, алгоритм итерации по ценности для опций, аналогичный (4.10), имеет вид:

$$v_{k+1}(s) \doteq \max_{\omega \in \Omega(s)} \left[r(s, \omega) + \sum_{s'} p(s'|s, \omega) v_k(s') \right] \quad \text{для всех } s \in \mathcal{S}. \quad (17.5)$$

Если $\Omega(s)$ включает все низкоуровневые действия, доступные в каждом состоянии s , то этот алгоритм сходится к традиционной функции v_* , по которой можно вычислить оптимальную стратегию. Но особенно полезно планировать с опциями, когда в каждом состоянии рассматривается только подмножество возможных опций (из $\Omega(s)$). Тогда итерация по ценности сходится к наилучшей иерархической стратегии на ограниченном множестве опций. Хотя эта стратегия может быть неоптимальной, сходимость может оказаться гораздо быстрее, поскольку рассматривается меньше опций и каждая опция может перепрыгивать через много временных шагов.

Для планирования с опциями нужно либо иметь модели опций, либо обучить их. Естественный способ обучения модели опций – описать ее как совокупность ОФЦ (определенных в предыдущем разделе), а затем обучить эти ОФЦ с помощью представленных в книге методов. Нетрудно видеть, как это можно сделать для части модели опций, связанной с вознаграждением. Нужно просто взять кумулянт одной ОФЦ в качестве вознаграждения ($C_t = R_t$), стратегией сделать стратегию опции ($\pi = \pi_\omega$), а функцией завершения – коэффициент обесценивания, умноженный на функцию завершения опции ($\gamma(s) = \gamma \cdot \gamma_\omega(s)$). Тогда истинная ОФЦ равна части модели опций, относящейся к вознаграждению, $v_{\pi, \gamma, C}(s) = r(s, \omega)$, а для ее аппроксимации можно использовать методы обучения, описанные в книге. Часть модели опций, относящаяся к переходу состояний, немного сложнее. Требуется назначить по одной ОФЦ каждому состоянию, в котором может завершиться опция. Мы не хотим, чтобы эти ОФЦ накапливали что-то до того, как опция завершится, да и в этом случае – лишь тогда, когда завершение имеет место в подходящем состоянии. Этого можно добиться, следующим образом выбрав кумулянт ОФЦ, предсказывающей переход в состояние s' : $C_t = (1 - \gamma_\omega(S_t)) \cdot 1_{S_t=s'}$. Стратегия и функция завершения ОФЦ выбираются такими же, как в части модели опций, относящейся к вознаграждению. Тогда истинная ОФЦ равна части s' модели переходов состояний опции, $v_{\pi, \gamma, C}(s) = p(s' | s, \omega)$, и для ее обучения опять-таки можно воспользоваться описанными в книге методами. Хотя каждый из этих шагов выглядит естественно, собрать их воедино (включая аппроксимацию функций и другие существенные компоненты) далеко не тривиально, и эта проблема пока не решена.

Упражнение 17.1. В этом разделе описаны опции для случая с обесцениванием, но обесценивание вряд ли применимо к управлению, когда используется аппроксимация функций (раздел 10.4). Как выглядит естественное уравнение Беллмана для иерархической стратегии, аналогичное (17.4), но в постановке со средним вознаграждением (раздел 10.3)? Каковы две части модели опций, аналогичные (17.2) и (17.3), для постановки со средним вознаграждением? □

17.3. НАБЛЮДЕНИЯ И СОСТОЯНИЕ

В этой книге мы записывали обученные приближенные функции ценности (и стратегии в главе 13) как функции от состояния окружающей среды. Это существенное ограничение методов, представленных в части I, где обученная функция ценности реализовывалась в виде таблицы, так чтобы можно было точно аппроксимировать любую функцию ценности; этот случай эквивалентен предположению о том, что агент может полностью наблюдать все состояние окружающей среды. Но во многих интересных случаях и, конечно, в жизни любого существа, наделенного естественным интеллектом, органы чувств дают лишь частичную информацию о состоянии мира. Какие-то объекты могут быть загорожены другими, находиться позади агента или за много километров от него. В таких случаях потенциально важные аспекты состояния окружающей среды непосредственно не наблюдаемы, и предположение о том, что обученная функция ценности реализована в виде таблицы над пространством состояний окружающей среды, было бы чрезмерно сильным, нереалистичным и ограничительным.

Система параметрической аппроксимации функции, разработанная нами в части II, гораздо менее ограничительна, и можно даже считать, что вообще не налагает ограничений. В части II мы сохранили предположение о том, что обученные функции ценности (и стратегии) являются функциями от состояния окружающей среды, но разрешили налагать на них произвольные ограничения в виде параметризации. Вызывает некоторое удивление, что аппроксимация функций включает важные аспекты частичной наблюдаемости; этот факт осознан далеко не всеми. Например, если имеется ненаблюдаемая переменная состояния, то параметризацию можно выбрать так, чтобы приближенная ценность не зависела от этой переменной. Эффект получается такой же, как если бы эта переменная состояния была ненаблюдаемой. Поэтому все результаты, полученные для параметрического случая, можно в неизменном виде применять к частичной наблюдаемости. В этом смысле случай параметрической аппроксимации функций включает случай частичной наблюдаемости.

Тем не менее существует много вопросов, которые невозможно изучить без более явного рассмотрения частичной наблюдаемости. Не имея возможности раскрыть здесь эту тему во всей полноте, мы все-таки можем вкратце описать требуемые изменения. Всего шагов четыре.

Во-первых, следует изменить постановку задачи. Нам доступны не состояния окружающей среды, а только *наблюдения* – сигналы, которые зависят от состояния, но, как датчики робота, несут лишь частичную информацию о нем. Для удобства мы можем без ограничения общности предположить, что вознаграждение является прямой известной функцией от наблюдения (быть может, наблюдение – это вектор, а вознаграждение – один из его элементов). Тогда вознаграждение с окружающей средой не описывается явными состояниями или вознаграждениями, а является просто чередующейся последовательностью действий $A_t \in \mathcal{A}$ и наблюдений $O_t \in \mathcal{O}$:

$$A_0, O_1, A_1, O_2, A_2, O_3, A_3, O_4, \dots,$$

продолжающейся бесконечно (см. уравнение (3.1)) или распадающейся на эпизоды, завершающиеся специальным заключительным наблюдением.

Во-вторых, мы можем восстановить идею состояния, используемую в этой книге, по последовательности наблюдений и действий. Будем употреблять слово *история* и обозначать H_t начальную часть траектории, ведущей к наблюдению: $H_t \doteq A_0, O_1, \dots, A_{t-1}, O_t$. История – это максимум того, что мы можем знать о прошлом, не выходя за пределы потока данных (поскольку история и есть весь прошлый поток данных). Конечно, история растет вместе с t и может стать большой и громоздкой. Идея состояния – взять некую компактную выжимку из истории, которая для предсказания будущего была бы так же полезна, как история целиком. Поясним, что это означает точно. Чтобы успешно играть роль выжимки, состояние должно быть функцией от истории, $S_t = f(H_t)$, а чтобы быть таким же полезным для предсказания будущего, как история целиком, оно должно обладать *марковским свойством*. Формально это свойство функции f . Говорят, что функция f обладает марковским свойством, если для любых двух историй h и h' , которые f отображает в одно и то же состояние ($f(h) = f(h')$), вероятности следующего наблюдения одинаковы:

$$f(h) = f(h') \Rightarrow \Pr\{O_{t+1} = o | H_t = h, A_t = a\} = \Pr\{O_{t+1} = o | H_t = h', A_t = a\} \quad (17.6)$$

для всех $o \in \mathcal{O}$ и $a \in \mathcal{A}$. Если f марковская, то $S_t = f(H_t)$ является состоянием в том смысле, который мы придавали этому термину в книге. Поэтому далее будем называть его *марковским состоянием*, чтобы отличить от состояний, являющихся выжимкой из истории, но не обладающих марковским свойством (мы рассмотрим их ниже).

Марковское состояние – хорошая основа для предсказания следующего наблюдения (17.6), но, что еще важнее, это также хорошая основа для предсказания или управления чем угодно. Например, пусть *тест* – любая конкретная последовательность чередующихся действий и наблюдений, которые могут иметь место в будущем. Так, трехшаговый тест обозначается $\tau = a_1 o_1 a_2 o_2 a_3 o_3$. Вероятность теста при условии истории h определяется следующим образом:

$$p(\tau|h) \doteq \Pr\{O_{t+1} = o_1, O_{t+2} = o_2, O_{t+3} = o_3 | H_t = h, A_t = a_1, A_{t+1} = a_2, A_{t+2} = a_3\}. \quad (17.7)$$

Если функция f марковская и h и h' – любые две истории, отображаемые f в одно и то же состояние, то для любого теста τ произвольной длины его вероятности при условии этих двух историй должны быть одинаковы:

$$f(h) = f(h') \Rightarrow p(\tau|h) = p(\tau|h'). \quad (17.8)$$

Иными словами, марковское состояние резюмирует всю содержащуюся в истории информацию, необходимую для определения вероятности любого теста. На самом деле оно резюмирует все необходимое для любого предсказания, в т. ч. любой ОФЦ, и для оптимального поведения (если f марковская, то всегда существует детерминированная функция π такая, что выбор $A_t \doteq \pi(f(H_t))$ оптимален).

Третий шаг обобщения обучения с подкреплением на частичную наблюдаемость – разрешение некоторых вычислительных проблем. В частности, мы хотим, чтобы состояние было компактной выжимкой из истории. Например, тождественная функция удовлетворяет всем условиям марковского свойства, но пользы от нее мало, потому что соответствующее состояние $S_t = H_t$ со временем растет и становится громоздким. Но есть и более фундаментальная причина – оно никогда не повторяется, т. е. агент никогда не встретит одно и то же состояние дважды (в непрерывной задаче) и, стало быть, не сможет воспользоваться методом табличного обучения. Мы хотим, чтобы состояния были не только марковскими, но и компактными. Похожая проблема возникает в связи с получением и обновлением состояния. Нам не нужна функция f , принимающая на входе истории целиком. Из вычислительных соображений мы предпочли бы получить такой же эффект с помощью рекуррентного обновления, которое вычисляет S_{t+1} по S_t и следующему инкременту данных, A_t и O_{t+1} :

$$S_{t+1} = u(S_t, A_t, O_{t+1}) \quad \text{для всех } t \geq 0 \quad (17.9)$$

при заданном первом состоянии S_0 . Функция u называется *функцией обновления состояния*. Например, если бы f была тождественной функцией ($S_t = H_t$), то u просто расширяла бы S_t , добавляя в конец A_t и O_{t+1} . При заданной f всегда можно построить соответствующую u , но с точки зрения вычислений это может быть неудобно, и, как в примере с тождественной функцией, получающееся состояние

может оказаться некомпактным. Функция обновления состояния – важнейшая часть любой архитектуры агента, предназначенного для работы в условиях частичной наблюдаемости. Она должна быть вычислительно эффективна, поскольку ни выбрать действие, ни сделать предсказание нельзя, пока состояние недоступно. На рис. 17.1 приведена схема такой архитектуры агента.

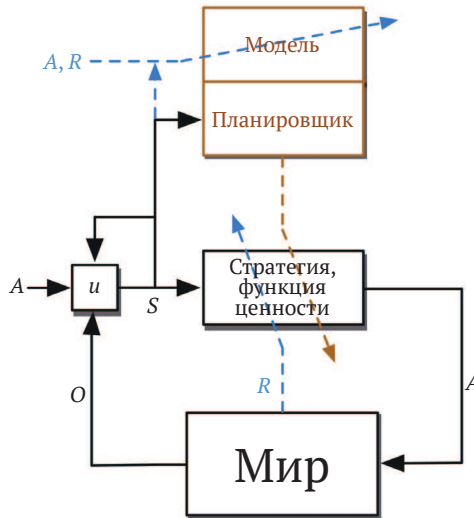


Рис. 17.1 ❖ Концептуальная архитектура агента, включающая модель, планировщик и функцию обновления состояния. В данном случае мир получает действия A и порождает наблюдения O . Наблюдения и копия действия используются функцией обновления состояния u для порождения нового состояния. Новое состояние является входом для стратегии и функции ценности, которые порождают новое действие, а также вход для планировщика (и для u). Потоки информации, наиболее существенные для обучения, показаны штриховыми линиями, пересекающими по диагонали прямоугольники, которые они изменяют. Вознаграждение R непосредственно изменяет стратегию и функцию ценности. Действие, вознаграждение и состояние изменяют модель, которая работает в тесной связке с планировщиком с целью изменить стратегию и функцию ценности. Заметим, что операции планировщика можно отсоединить от взаимодействия агента и окружающей среды, тогда как другие процессы должны работать синхронно с этим взаимодействием, чтобы не отставать от поступления новых данных. Также заметим, что модель и планировщик имеют дело не напрямую с наблюдениями, а только с состояниями, порожденными функцией u , которые могут играть роль целей при обучении модели

Пример получения марковских состояний с помощью функции обновления состояний дает популярный байесовский подход, известный под названием *частично наблюдаемый МППР*, или ЧНМППР (Partially Observable MDP – POMDP). В этом случае предполагается, что в окружающей среде имеется корректно определенное *скрытое состояние* X_t , которое и объясняет все порождаемые наблюдения, но агенту недоступно (и не следует путать его с состоянием S_t , которое агент использует для предсказаний и принятия решений). Естественное для ЧНМППР марковское состояние S_t – это *распределение* скрытых состояний при условии истории, оно называется *доверительным состоянием*. Для определенности рассмотрим обыч-

ный случай, когда существует конечное число скрытых состояний $X_t \in \{1, 2, \dots, d\}$. Тогда доверительным состоянием будет вектор $S_t \doteq \mathbf{s}_t \in \mathbb{R}^d$ с элементами

$$\mathbf{s}_t[i] \doteq \Pr\{X_t = i | H_t\} \text{ для всех возможных скрытых состояний } i \in \{1, 2, \dots, d\}.$$

Размер доверительного состояния (количество элементов вектора) остается постоянным, как бы ни росло t . Его также можно инкрементно обновлять с помощью байесовского правила в предположении, что имеется полная информация о внутреннем устройстве окружающей среды. Точнее, i -й элемент функции обновления доверительного состояния имеет вид:

$$u(\mathbf{s}, a, o)[i] = \frac{\sum_{x=1}^d \mathbf{s}[x] p(i, o | x, a)}{\sum_{x=1}^d \sum_{x'=1}^d \mathbf{s}[x] p(x', o | x, a)} \quad (17.10)$$

для всех $a \in \mathcal{A}$, $o \in \mathcal{O}$ и доверительных состояний $\mathbf{s} \in \mathbb{R}^d$ с элементами $\mathbf{s}[x]$, где функция четырех аргументов p в данном случае не такая, как обычно в МППР (см. главу 3), а аналогичная функция для ЧНМППР в терминах *скрытого* состояния: $p(x', o | x, a) \doteq \Pr\{X_t = x', O_t = o | X_{t-1} = x, A_{t-1} = a\}$. Этот подход популярен в теоретических работах и имеет много важных приложений, но принятые в нем предположения и вычислительная сложность плохо масштабируются, поэтому мы не рекомендуем его как подход к искусственному интеллекту.

Еще один пример марковских состояний дают *представления прогностических состояний* (ППС) (Predictive State Representation – PSR). ППС решают проблему слабости подхода ЧНМППР, связанную с тем, что семантика состояния агента в нем S_t крепко привязана к состоянию окружающей среды X_t , которое не наблюдаемо в принципе, поэтому что-то узнать о нем сложно. В ППС и родственных подходах семантика состояния агента вместо этого привязана к предсказаниям будущих наблюдений и действий, которые вполне поддаются наблюдению. В ППС марковское состояние определяется как d -мерный вектор вероятностей d специально подобранных «ядерных» тестов в смысле определения (17.7) выше. Затем вектор обновляется функцией обновления состояний u , которая аналогична байесовскому правилу, но с семантикой, привязанной к наблюдаемым данным, что теоретически упрощает ее обучение. Этот подход имеет ряд обобщений, например: конечные тесты, композиционные тесты, мощные «спектральные» методы, а также тесты с обратной связью и с абстрагированием времени, обучаемые TD-методами. Некоторые из лучших теоретических достижений получены для систем, известных под названием *наблюдаемые операторные модели* (Observable Operator Model – OOM) и *последовательные системы* (Sequential Systems) (Thon, 2017).

Четвертый и последний шаг в нашем кратком описании частичной наблюдаемости в обучении с подкреплением – как восстановить аппроксимацию. Во введении к части II мы говорили, что любой серьезный подход к искусственному интеллекту должен включать аппроксимацию. Это справедливо для состояний в той же мере, как для функций ценности. Мы должны ввести понятие приближенного состояния и научиться работать с ним. Приближенное состояние будет играть в наших алгоритмах ту же роль, что и прежде, поэтому мы продолжим использовать нотацию S_t для состояния, используемого агентом, хотя оно может и не быть марковским.

Наверное, самый простой пример приближенного состояния – последнее наблюдение, $S_t \doteq O_t$. Конечно, при таком подходе ни о какой обработке скрытой информации не может быть и речи. Было бы лучше использовать последние k наблюдений и действий, $S_t \doteq O_t, A_{t-1}, O_{t-1}, \dots, A_{t-k}$, при некотором $k \geq 1$. Этого можно добиться с помощью функции обновления состояний, которая помещает в историю новые данные, вытесняя самые старые. Подход на основе истории k -го порядка по-прежнему очень прост, но может заметно расширить возможности агента по сравнению с использованием только лишь последнего наблюдения в качестве состояния.

Что будет, если марковское свойство (17.6) выполняется лишь приближенно? К сожалению, предсказания на длительную перспективу могут ухудшиться очень сильно, если предсказания на один шаг, определяющие марковское свойство, хоть немного неточны. И долгосрочные тесты, и ОФЦ, и функции обновления состояний могут аппроксимироваться плохо. Целевые функции аппроксимации на короткий и на длительный срок попросту различны, и в настоящее время неизвестно никаких полезных теоретических гарантий.

И тем не менее есть причины полагать, что общая идея, изложенная в этом разделе, применима к приближенному случаю. Общая идея заключается в том, что состояние, хорошее для некоторых предсказаний, хорошо и для других (в частности, марковского состояния, достаточного для предсказания на один шаг, достаточно и для всех остальных). Если отступить на шаг от этого конкретного результата для марковского случая, то эта общая идея похожа на то, что мы обсуждали в разделе 17.1, посвященном многоголовому обучению и вспомогательным задачам. Мы говорили, что представления, хорошие для вспомогательных задач, часто хороши и для главной. Объединив то и другое, мы придем к общему подходу к частичной наблюдаемости и обучению представлений, в котором делается несколько предсказаний, используемых, чтобы придать нужное направление процессу конструирования признаков состояний. Гарантии, которые дает идеальное, но непрактичное марковское свойство, заменяются эвристикой, согласно которой что хорошо для некоторых предсказаний, может оказаться хорошо и для других. Этот подход неплохо масштабируется при добавлении вычислительных ресурсов. Имея мощный компьютер, можно поэкспериментировать с большим числом предсказаний, быть может, отдавая предпочтение тем, которые больше всего похожи на интересующие нас в конечном итоге, или тем, которые проще всего надежно обучить, или отобранным еще по какому-то критерию. Здесь важно не останавливаться на выборе предсказаний вручную. Это должен делать агент. Возможно, для этого потребуется общий язык предсказаний, чтобы агент мог систематически исследовать большое пространство возможных предсказаний, отыскивая среди них наиболее полезные.

В частности, оба подхода, ЧНМППР и ППС, могут работать с приближенными состояниями. Семантика состояния полезна при построении функции обновления состояний, как в этих двух подходах и в истории k -порядка. Семантика не обязательно должна быть корректной, чтобы сохранять полезную информацию в состоянии. Некоторые подходы к пополнению состояний, например сети эхосостояний (Jaeger, 2002), позволяют хранить почти произвольную информацию об истории и тем не менее работают хорошо. Возможностей много, и мы ожидаем появления новых работ и идей в этой области. Обучение функции обновления со-

стояний для приближенного состояния – существенная часть проблемы обучения представлений, возникающей в обучении с подкреплением.

17.4. ПРОЕКТИРОВАНИЕ СИГНАЛОВ ВОЗНАГРАЖДЕНИЯ

Важное преимущество обучения с подкреплением над обучением с учителем заключается в том, что первое не нуждается в подробных инструкциях: генерация сигнала вознаграждения не зависит от знаний о том, какими должны быть правильные действия агента. Но успех приложения обучения с подкреплением сильно зависит от того, насколько хорошо сигнал вознаграждения отражает цели проектировщика и оценивает прогресс в достижении цели. Поэтому проектирование сигнала вознаграждения – критически важная часть любого приложения обучения с подкреплением.

Под проектированием сигнала вознаграждения мы понимаем проектирование той части окружающей агента среды, которая отвечает за вычисление скалярного вознаграждения R_t и отправку его агенту в каждый момент t . Обсуждая терминологию в конце главы 14, мы сказали, что R_t больше похож на сигнал, генерируемый в мозге животного, чем на объект или событие в окружающей животное среде. Те части нашего мозга, которые генерируют эти сигналы, эволюционировали миллионы лет и хорошо приспособлены к опасностям, с которыми сталкивались наши предки в борьбе за передачу своих генов будущим поколениям. Поэтому не стоит думать, что проектирование хорошего сигнала вознаграждения – всегда простое дело!

Одна из трудностей – спроектировать сигнал вознаграждения, так чтобы по мере обучения агента его поведение становилось ближе, а в идеале совпадало с тем, к которому стремится проектировщик приложения. Это может быть легко, если цель проектировщика проста и легко идентифицируется, например найти решение корректно поставленной задачи или заработать как можно больше очков в четко определенной игре. В таких случаях агент обычно вознаграждается в соответствии с успехом в решении задачи или улучшением своего счета. Но в некоторых задачах цели трудно выразить в виде сигналов вознаграждения. Особенно это относится к задачам, в которых агент должен искусно выполнить сложное задание или набор заданий, как, например, домашний робот-помощник. Кроме того, агенты обучения с подкреплением могут найти неожиданные способы получить от окружающей среды вознаграждение – некоторые из них могут быть нежелательны или даже опасны. Это старая проблема, стоящая перед любым методом, основанным на оптимизации, в т. ч. обучением с подкреплением. Мы еще вернемся к ней в разделе 17.6 – последнем в этой книге.

Даже когда цель проста и легко идентифицируется, часто возникает проблема разреженного вознаграждения. Начисление ненулевого вознаграждения достаточно часто, чтобы агент хотя бы один раз достиг цели, не говоря уже о том, чтобы научиться делать это эффективно при разных начальных условиях, – задача не из легких. Пар состояние–действие, для которых генерация вознаграждения, безусловно, оправдана, может быть мало, и они могут находиться далеко друг от друга. А вознаграждения, знаменующие приближение к цели, могут встречаться нечасто, поскольку прогресс трудно или даже невозможно обнаружить. Агент может

бесцельно блуждать в течение длительного времени (Мински в статье 1961 года называл это «проблемой плато»).

На практике проектирование сигнала вознаграждения часто сводится к неформальному поиску методом проб и ошибок сигнала, дающего приемлемые результаты. Если агент не обучается, обучается слишком медленно или не тому, чему нужно, то проектировщик поправляет сигнал вознаграждения и пробует еще раз. Для этого проектировщик судит о качестве агента по критерию, который пытается перевести в сигнал вознаграждения, так чтобы его цели совпали с целями агента. А если обучение слишком медленное, то проектировщик может попробовать придумать неразрезанный сигнал вознаграждения, который эффективно руководит обучением на всем протяжении взаимодействия агента с окружающей средой.

Возникает искушение решить проблему разреженного вознаграждения, начисляя агенту вознаграждение за достижение подцелей, которые, по мнению проектировщика, являются важными промежуточными остановками на пути к основной цели. Но дополнение сигнала вознаграждения другими сигналами, пусть даже с самыми лучшими намерениями, может привести к тому, что агент станет вести себя совсем не так, как задумано, и может вовсе не достигнуть цели. Лучше оставить сигнал вознаграждения в покое, а дополнить аппроксимацию функции ценности начальной гипотезой о том, какой она должна быть, или начальными гипотезами о том, какими должны быть некоторые ее части. Например, предположим, что мы хотим предложить $v_0: \mathcal{S} \rightarrow \mathbb{R}$ в качестве начальной гипотезы об истинной оптимальной функции ценности v_* и что используется линейная аппроксимация с признаками $\mathbf{x}: \mathcal{S} \rightarrow \mathbb{R}^d$. Тогда начальное приближение к функции ценности следовало бы определить так:

$$\hat{v}(s, \mathbf{w}) \doteq \mathbf{w}^T \mathbf{x}(s) + v_0(s), \quad (17.11)$$

а веса обновлять как обычно. Если начальный вектор весов равен $\mathbf{0}$, то начальная функция ценности будет равна v_0 , но качество асимптотического решения будет, как обычно, определяться вектором признаков. Такую инициализацию можно провести для произвольных нелинейных аппроксиматоров и произвольных форм v_0 , хотя не гарантируется, что это обязательно ускорит обучение.

Особенно эффективный подход к проблеме разреженного вознаграждения предлагает техника *формирования*, введенная психологом Б. Ф. Скиннером и описанная в разделе 14.3. Ее эффективность опирается на тот факт, что разреженное вознаграждение – проблема не просто разреженного сигнала, но и стратегии агента, не позволяющей ему часто посещать вознаграждающие состояния. Формирование включает изменение сигнала вознаграждения в процессе обучения, начав с сигнала, не являющегося разреженным при заданном начальном поведении агента и постепенно модифицируя его в сторону сигнала, подходящего для исходной задачи, интересующей исследователя. Каждая модификация производится так, чтобы агент часто получал вознаграждение при текущем поведении. Агент сталкивается с последовательностью задач обучения с подкреплением возрастающей трудности, но то, чему он обучился на предыдущем этапе, делает следующую задачу относительно простой, потому что теперь агент получает вознаграждение чаще, чем если бы у него не было предшествующего опыта решения более легких задач. Такое формирование – важная техника при дрессировке животных, но доказало свою эффективность и в вычислительном обучении с подкреплением.

А что, если исследователь совсем не знает, каким должно быть вознаграждение, но существует другой агент, быть может, человек, который уже поднатерел в решении задачи и за поведением которого можно наблюдать? В таком случае можно воспользоваться методами, известными под разными названиями: «имитационное обучение», «обучение путем демонстрации» или «обучение на стажировке». Идея в том, чтобы воспользоваться знаниями опытного агента, но сохранить возможность в конечном итоге превзойти его. Обучение на поведении эксперта можно проводить непосредственно с учителем или выделить сигнал вознаграждения с помощью так называемого «инверсного обучения с подкреплением», а затем применить какой-нибудь алгоритм обучения с подкреплением с этим сигналом, чтобы обучить стратегию. Задача инверсного обучения с подкреплением, исследованная в работе Ng and Russell (2000), заключается в том, чтобы попытаться выделить сигнал вознаграждения, наблюдая только поведение эксперта. Сделать это точно невозможно, потому что стратегия может быть оптимальной относительно многих разных сигналов (например, всех сигналов, которые приносят одно и то же вознаграждение для всех состояний и действий), но можно найти правдоподобных кандидатов на роль сигнала вознаграждения. К сожалению, для этого необходимы сильные предположения, в т. ч. знания о динамике окружающей среды и о векторах признаков, относительно которых сигнал вознаграждения линеен. Кроме того, метод требует, чтобы задача была полностью решена (например, методами динамического программирования) несколько раз. Но, несмотря на все эти трудности, в работе Abbeel and Ng (2004) утверждается, что инверсное обучение с подкреплением иногда может оказаться эффективнее обучения с учителем, если нужно извлечь пользу из поведения эксперта.

Еще один подход к отысканию хорошего сигнала вознаграждения – автоматизировать поиск сигнала методом проб и ошибок, о чем мы уже упоминали выше. С точки зрения приложения, сигнал вознаграждения – параметр алгоритма обучения. Как и для всех остальных параметров, поиск хорошего сигнала можно автоматизировать, определив пространство возможных кандидатов и применив алгоритм оптимизации. Алгоритм оптимизации оценивает каждый потенциальный сигнал вознаграждения, исполняя систему обучения с подкреплением с этим сигналом на протяжении нескольких шагов, а затем оценивая результат с помощью «высокоуровневой» целевой функции, которая призвана закодировать истинную цель проектировщика, игнорируя ограничения агента. Сигналы вознаграждения можно даже улучшить с помощью онлайн-градиентного спуска, в которых вычисляется градиент этой высокоуровневой целевой функции (Sorg, Lewis, and Singh, 2010). Если проводить аналогию с природой, то алгоритм оптимизации высокоуровневой целевой функции можно уподобить эволюции, а саму эту функцию – эволюционной приспособленности животного, определяемой количеством его потомков, доживших до репродуктивного возраста.

Вычислительные эксперименты с такой двухуровневой оптимизацией – один уровень аналогичен эволюции, а другой заключается в обучении отдельных агентов с подкреплением – подтвердили, что одной лишь интуиции не всегда достаточно, чтобы придумать хороший сигнал вознаграждения (Singh, Lewis, and Barto, 2009). Качество агента вознаграждения с подкреплением, оцениваемое с помощью высокоуровневой целевой функции, может быть очень чувствительно к деталям сигнала вознаграждения агента, причем зависимость может быть неочевидной и определяться ограничениями агента и окружающей средой, в кото-

рой агент действует и обучается. Эти эксперименты продемонстрировали также, что цель агента не обязательно должна совпадать с целью его проектировщика.

Поначалу это кажется противоречащим интуиции, но может оказаться, что агент не способен достичь целей, поставленных проектировщиком, каким бы ни был сигнал вознаграждения. Агенту приходится обучаться при различных ограничениях, например: ограниченные вычислительные ресурсы, ограниченный доступ к информации об окружающей среде или ограниченное время обучения. Если такие ограничения имеют место, то иногда обучение достижению цели, отличной от цели проектировщика, помогает подойти к цели проектировщика ближе, чем при лобовом преследовании этой цели (Sorg, Singh, and Lewis, 2010; Sorg, 2011). Такого рода примеры легко найти в природе. Поскольку мы не можем непосредственно оценить питательную ценность большинства продуктов, эволюция – проектировщик нашего сигнала вознаграждения – снабдила нас сигналом вознаграждения, побуждающим искать определенные вкусы. Это, конечно, не безошибочный критерий (он даже может оказаться вредным для здоровья, если окружающая среда чем-то отличается от знакомой нашим предкам), но он компенсирует многие наши ограничения: ограниченность способностей к восприятию, ограниченное время обучения и риски, сопутствующие попыткам найти здоровую диету на личном опыте. Аналогично, поскольку животное не может наблюдать собственную эволюционную приспособленность, эта целевая функция не годится на роль сигнала вознаграждения при обучении. Вместо нее эволюция предлагает сигналы вознаграждения, чувствительные к наблюдаемым предсказателям эволюционной приспособленности.

Наконец, напомним, что агент обучения с подкреплением необязательно является полным организмом или роботом; он может быть частью более сложной функционирующей системы. Это означает, что на сигналы вознаграждения могут влиять факторы внутри объемлющего функционирующего агента, например мотивационные состояния, воспоминания, идеи или даже галлюцинации. Сигналы вознаграждения могут также зависеть от свойств самого процесса обучения, например способа измерения его прогресса. Если сделать сигналы вознаграждения чувствительными к информации о таких внутренних факторах, то можно будет обучить агента управлять «когнитивной архитектурой», частью которой он является, а также приобретать знания и навыки, которым трудно было бы обучиться на сигнале вознаграждения, зависящем только от внешних событий. Такого рода возможности привели к идее «внутренне мотивированного обучения с подкреплением», которую мы кратко обсудим в конце следующего раздела.

17.5. ОСТАЮЩИЕСЯ ВОПРОСЫ

В этой книге мы в общих чертах познакомились с подходом к искусственному интеллекту на основе обучения с подкреплением. Грубо говоря, этот подход опирается на совместную работу безмодельных и основанных на модели методов, как в архитектуре Дуна из главы 8, в сочетании с аппроксимацией функций, разработанной в части II. Мы рассматривали в основном онлайн-овые и инкрементные алгоритмы, которые считали фундаментальными даже для основанных на модели методов, а также способы применения этих алгоритмов к обучению с разделенной стратегией. Полное обоснование последнего было представлено только в этой

главе. То есть всю дорогу мы представляли обучение с разделенной стратегией как заманчивый способ разрешить дилемму исследования и использования, но лишь в этой главе обсудили обучение многим разным вспомогательным задачам одновременно с помощью ОФЦ и иерархического получения знаний о мире в терминах абстрагированных от времени моделей опций – то и другое подразумевает обучение с разделенной стратегией. Еще остается много работы, мы не раз говорили об этом в книге, и подтверждением тому служат направления дополнительных исследований, упомянутые в этой главе. Но допустим, что все описанное в данной книге и в этой главе уже сделано. Что потом? Конечно, мы не можем наверняка знать, что потребуется, но можем высказать некоторые догадки. В этом разделе мы перечислим шесть проблем, которые, как нам кажется, нужно будет разрешить будущим исследователям.

Во-первых, нам по-прежнему нужны мощные методы параметрической аппроксимации функций, которые хорошо работают в инкрементной и онлайн-овой постановке. Методы, основанные на глубоком обучении и ИНС, – важный шаг в этом направлении, но все-таки они хорошо работают только при пакетном обучении на больших наборах данных, при обучении на большом количестве офлайн-овых игр с собой или при обучении на чередующемся опыте нескольких агентов, решающих одну и ту же задачу. Эти и другие постановки – попытки обойти фундаментальное ограничение современных методов глубокого обучения и найти способ быстро обучаться в инкрементных, онлайн-овых условиях, наиболее естественных для алгоритмов обучения с подкреплением, описанных в этой главе. Иногда эту проблему называют «катастрофической интерференцией» или «коррелированными данными». Стоит обучиться чему-то новому, как это новое замещает, а не дополняет выученное ранее, а все, чему удалось научиться до этого, теряется. Чтобы сохранить и повторно воспроизвести старые данные и не дать пропасть плодам прошлого обучения, часто применяются такие приемы, как «буферы воспроизведения». Нужно честно признать, что современные методы глубокого обучения плохо приспособлены к онлайн-овому обучению. Мы не думаем, что это ограничение непреодолимо, но алгоритмы, которые решают данную проблему, не утрачивая преимуществ глубокого обучения, еще только предстоит изобрести. Большинство современных исследований по глубокому обучению направлены на то, чтобы обойти это ограничение, а не снять его.

Во-вторых (и это, пожалуй, тесно связано с первой проблемой), нам по-прежнему нужны методы обучения признакам, чтобы результаты последующего обучения хорошо обобщались. Это частный случай общей проблемы, которую называют «обучением представлений», «конструктивной индукцией» и «метаобучением», – как воспользоваться опытом не только для того, чтобы обучить желаемую функцию, но и индуктивные смещения, с тем чтобы результаты будущего обучения обобщались лучше, а значит, само обучение занимало меньше времени? Это старая проблема, восходящая еще к истокам искусственного интеллекта и распознавания образов, т. е. к 1950–1960-м годам¹. Такая давность должна за-

¹ Кто-то, наверное, будет утверждать, что глубокое обучение решает эту проблему, например что описанная в разделе 16.5 сеть DQN как раз и является примером решения, но нас это не убеждает. Пока еще слишком мало свидетельств в пользу того, что глубокое обучение само по себе решает проблему обучения представлений эффективно и в общем виде.

ставить задуматься. Быть может, решения и не существует. Но столь же вероятно, что время для отыскания решения и демонстрации его эффективности еще не настало. В наши дни машинное обучение производится в куда больших масштабах, чем в прошлом, и потенциальные преимущества хорошего метода обучения представлений стали намного очевиднее. С 2013 года ежегодно проводится Международная конференция по обучению представлений, посвященная этому и смежным вопросам. Кроме того, не так часто можно встретить исследование обучения представлений в контексте обучения с подкреплением. Обучение с подкреплением открывает новые подходы к этой старой проблеме, например вспомогательные задачи, обсуждавшиеся в разделе 17.1. В обучении с подкреплением проблему обучения представлений можно рассматривать как проблему обучения функции обновления состояний (раздел 17.3).

В-третьих, нам по-прежнему нужны методы планирования при наличии обученных моделей окружающей среды. Методы планирования доказали свою высочайшую эффективность в таких приложениях, как AlphaGo Zero и шахматные программы, в которых модель среды известна из правил игры или может быть представлена по-другому проектировщиком. Но случаи полного основанного на модели обучения с подкреплением, когда модель окружающей среды сначала обучается на данных, а затем используется для планирования, редки. Один из примеров такого рода дает архитектура Dyna, описанная в главе 8, но и у нас, и в большинстве последующих работ в ней используется табличная модель без аппроксимации функций, что сильно ограничивает ее применимость. Немного есть работ, включающих обученные линейные модели, а еще меньше таких, где исследованы также абстрагирующие время модели опций, описанные в разделе 17.2.

Необходима дополнительная работа, прежде чем планирование с использованием обученных моделей станет эффективным. Например, обучение модели должно быть избирательным, поскольку рамки модели оказывают большое влияние на эффективность планирования. Если модель сфокусирована на главных последствиях самых важных опций, то планирование может быть эффективным и быстрым, а если она включает детали второстепенных последствий опций, которые вряд ли когда-нибудь будут выбраны, то планирование может оказаться почти бесполезным. Модели окружающей среды следует конструировать с оглядкой и на состояния, и на динамику, ставя целью оптимизацию процесса планирования. Следует вести постоянный мониторинг того, как разные части модели повышают или снижают эффективность планирования. В отрасли еще не уделялось достаточно внимания этому комплексу проблем, и никто не занимался разработкой методов обучения моделей, которые принимали бы во внимание вытекающие из этих проблем следствия.

Четвертая проблема, которую нужно разрешить в будущих исследованиях, – автоматизация выбора заданий, над которыми агент работает и которые использует для развития своей компетенции. В машинном обучении принято, что человек ставит задачи, в которых агент должен достичь мастерства. Поскольку эти задачи известны заранее и остаются фиксированными, их можно встроить в код алгоритма обучения. Но, заглядывая вперед, мы хотели бы, чтобы агент сам решал, в каких задачах он должен попытаться преуспеть. Это могут быть подзадачи уже известной конкретной общей задачи. Или это может быть построение структур-

ных элементов, которые позволят эффективнее обучаться разнообразным задачам, с которыми агенту, возможно, предстоит столкнуться в будущем, но которые пока неизвестны.

Это могут быть вспомогательные задачи, или ОФЦ, рассмотренные в разделе 17.1, или задачи, решаемые с помощью опций (раздел 17.2). Например, при формировании ОФЦ какими должны быть кумулянт, стратегия и функция завершения? В настоящее время их можно задавать только вручную, но куда большей эффективности и общности можно было бы достичь, если бы выбирать задачи можно было автоматически, особенно когда они выводятся из того, что агент конструировал ранее как результат обучения представлений или опыта решения предыдущих задач. Если проектирование ОФЦ автоматизировать, то сами проектные варианты нужно будет представлять явно. Они будут находиться не в голове проектировщика и не будут встраиваться в код, а вместо этого должны будут стать частью самого механизма, чтобы их можно было автоматически задавать и изменять, производить мониторинг, фильтрацию и поиск. Тогда из задач можно будет строить иерархии, как обстоит дело с признаками в ИНС. Задачи – это вопросы, а содержимое ИНС – ответы на эти вопросы. Мы ожидаем, что понадобится полная иерархия вопросов, соответствующая иерархии ответов, которые дают современные методы глубокого обучения.

Пятая проблема, на которой мы хотели бы заострить внимание, – взаимодействие между поведением и обучением посредством какого-нибудь вычислительного аналога любознательности. В этой главе мы грезили о такой постановке задачи, при которой много задач обучаются одновременно с применением методов с разделенной стратегией на одном и том же потоке опыта. Предпринимаемые действия, конечно, будут оказывать влияние на этот поток опыта, что, в свою очередь, будет определять, насколько продвинулось обучение и какие задачи обучены. Если вознаграждение недоступно или поведение слабо влияет на него, то агент вправе выбирать действия, которые в каком-то смысле максимизируют обучение на задачах, т. е. использовать некоторую метрику прогресса обучения как внутреннее или «присущее» вознаграждение, реализующее вычислительную форму любознательности. Помимо измерения прогресса обучения, присущее вознаграждение может, в числе прочего, сигнализировать о получении неожиданных, новых или еще чем-то интересных входных данных или оценивать способность агента вызывать изменения в окружающей среде. Сгенерированные таким способом сигналы присущего вознаграждения могут использоваться агентом для постановки задач самому себе путем определения обсуждавшихся выше вспомогательных задач, ОФЦ или опций, так чтобы приобретенные навыки могли внести вклад в способность агента к решению будущих задач. Результатом является вычислительный аналог чего-то вроде игры. Уже выполнено много предварительных работ по такому использованию сигналов присущего вознаграждения, и в этой области остается немало интересных тем для будущих исследований.

Последняя проблема, заслуживающая внимания будущих исследователей, – разработка методов, которые позволили бы безопасно встраивать агентов обучения с подкреплением в физическую окружающую среду. Это одна из самых насущных областей будущих исследований, и мы обсудим ее в следующем разделе.

7.6. ЭКСПЕРИМЕНТАЛЬНОЕ ПОДТВЕРЖДЕНИЕ ГИПОТЕЗЫ ОБ ОШИБКЕ ПРЕДСКАЗАНИЯ ВОЗНАГРАЖДЕНИЯ

Когда в середине 1990-х годов мы работали над первым изданием этой книги, искусственный интеллект как раз демонстрировал значительный прогресс и начинал оказывать влияние на общество, хотя исследователей по-прежнему воодушевляли в основном *обещания*, которые он сулил. Частью этого мироощущения стало машинное обучение, но оно еще не превратилось в неотъемлемый атрибут ИИ. Сегодня это обещание воплотилось в приложения, которые изменяют жизни миллионов людей, а машинное обучение стало самостоятельной технологией – одной из ключевых. Когда мы пишем это, второе, издание, многие из самых поразительных достижений искусственного интеллекта включают обучение с подкреплением, особенно «глубокое обучение с подкреплением», т. е. обучение с подкреплением, в котором для аппроксимации функции применяются искусственные нейронные сети. Скоро мы увидим целую волну приложений ИИ к реальному миру, и многие из них будут включать обучение с подкреплением, глубокое и не только, которое окажет на нашу жизнь такое влияние, которое сейчас даже трудно предвидеть.

Но изобилие успешных реальных приложений не означает, что настала эра истинного искусственного интеллекта. Несмотря на заметный прогресс во многих областях, пропасть между искусственным интеллектом и интеллектом человека, да даже и животных, остается огромной. В некоторых предметных областях, даже таких трудных, как игра го, машина может превзойти человека, но еще далеко до разработки систем, которые, как мы сами, будут полноценными интерактивными агентами, способными адаптироваться в широких пределах, решать различные задачи, проявлять эмоции, творческие способности и быстро обучаться на опыте. Обучение с подкреплением, в центре которого лежит обучение путем взаимодействия с динамической окружающей средой, по мере своего развития станет одним из главных компонентов агентов, обладающих такими способностями.

Благодаря связям с психологией и нейронауками (главы 14 и 15) обучение с подкреплением имеет прямое отношение еще к одной давней цели искусственного интеллекта: пролить свет на фундаментальные вопросы, касающиеся разума и его появления в результате мозговой деятельности. Теория обучения с подкреплением уже вносит вклад в наше понимание мозгового вознаграждения, мотивации и процессов принятия решений, и имеются веские причины полагать, что благодаря связям с вычислительной психиатрией она поспособствует разработке методов лечения психических расстройств, в т. ч. наркомании и наркозависимости.

Еще одна работа, которую обучение с подкреплением может взять на себя в будущем, – помощь человеку в принятии решений. Стратегии, выведенные с помощью обучения с подкреплением в имитированной окружающей среде, могут что-то подсказать людям, принимающим решения, в таких областях, как образование, здравоохранение, транспорт, энергетика и выделение ресурсов в государственном секторе. Особенный интерес представляет ключевая черта обучения с подкреплением – учет долгосрочных последствий принимаемых решений. Это

наглядно видно в таких играх, как нарды и го, в которых были продемонстрированы наиболее впечатляющие успехи обучения с подкреплением, но справедливо и для других ответственных решений, влияющих на жизнь людей и всей планеты. В обучении с подкреплением прослеживается связь с методами содействия в принятии решений, которые были в прошлом разработаны специалистами во многих дисциплинах. При наличии развитых методов аппроксимации функций и больших вычислительных ресурсов обучение с подкреплением имеет все шансы справиться с некоторыми трудностями вертикального масштабирования традиционных методов поддержки принятия решений на более крупные и сложные задачи.

Быстрый прогресс в области искусственного интеллекта породил опасения, что ИИ может представлять серьезную угрозу обществу и даже самому человечеству. Известный ученый и пионер ИИ Герберт Саймон (Herbert Simon) предвидел те предостережения, которые мы слышим сегодня, когда проводил презентацию на Earthware Symposium в университете Карнеги-Меллона в 2000 году (Simon, 2000). Он говорил об извечном конфликте между обещаниями и опасностями любого нового знания, напоминал древнегреческий миф о Прометее – герое современной науки, укравшем у богов огонь для блага людей, – и о Пандоре, ящик которой открывался легким, невинным движением, но выпускал в мир неисчислимые беды. Принимая неизбежность этого конфликта, Саймон призывал нас осознать, что мы сами – творцы своего будущего, а не просто зрители, и что наши решения могут склонить чашу весов в пользу Прометея. Безусловно, это относится и к обучению с подкреплением, которое может принести благо обществу, но может породить и нежелательные последствия, если пользоваться им беспечно. Таким образом, безопасность приложений искусственного интеллекта с привлечением обучения с подкреплением – тема, заслуживающая пристального внимания.

Агент обучения с подкреплением может обучаться, взаимодействуя или с реальным миром, или с имитацией какой-то его части, или с обоими источниками опыта сразу. Конструкторы имитационной модели предоставляют безопасную окружающую среду, в которой агент может заниматься исследованием и обучением, не рискуя нанести вред себе или окружению. В большинстве современных приложений обучение стратегии производится на имитированном опыте, а не путем прямого взаимодействия с реальным миром. Помимо предотвращения нежелательных последствий для мира, для обучения на имитированном опыте мы можем создать практически неограниченное количество данных, как правило с меньшими затратами, чем было бы нужно для получения реального опыта. А поскольку время в имитационной модели обычно течет гораздо быстрее реального, то обучение часто удается завершить быстрее.

Тем не менее, чтобы использовать потенциал обучения с подкреплением до конца, агентов необходимо погружать в поток реального опыта, чтобы они могли действовать, исследовать и обучаться в нашем, а не в выдуманном мире. В конце концов, алгоритмы обучения с подкреплением – по крайней мере, рассмотренные в этой книге – предназначены для обучения в онлайн-овом режиме, они повторяют многие аспекты поведения животных, позволяющие выживать в нестабильной и враждебной среде. Погружение агентов обучения с подкреплением в реальный мир может произвести революцию в исполнении обещаний искусственного интеллекта – усилить и расширить человеческие способности.

Основная причина, почему мы хотим, чтобы агент обучения с подкреплением действовал и обучался в реальном мире, состоит в том, что зачастую трудно, а то и невозможно имитировать реальный опыт настолько достоверно, что результирующие стратегии – не важно, выработаны они с помощью обучения с подкреплением или другими методами, – хорошо – и безопасно – работали при управлении реальными действиями. Это особенно относится к средам, динамика которых зависит от поведения людей, например в образовании, здравоохранении, на транспорте и в сфере государственного управления – тех областях, где повышение качества принятия решений, безусловно, могло бы принести пользу. Однако именно в связи с агентами, погруженными в реальный мир, к предостережениям о потенциальных опасностях искусственного интеллекта следует прислушаться особенно внимательно.

Некоторые из этих предостережений особенно актуальны в применении к обучению с подкреплением. Поскольку обучение с подкреплением основано на оптимизации, оно унаследовало все плюсы и минусы, свойственные любому методу оптимизации. Минус – проблема конструирования такой целевой функции или сигнала вознаграждения, чтобы оптимизация давала желаемые результаты, избегая нежелательных. В разделе 17.4 мы говорили, что агенты обучения с подкреплением могут находить неожиданные способы получить вознаграждение от окружающей среды, и какие-то из них могут быть нежелательны или даже опасны. Когда мы лишь косвенно описываем, чему система должна обучиться, как в случае проектирования сигнала вознаграждения для системы обучения с подкреплением, мы не можем знать, насколько точно агент выполнил наше пожелание, пока обучение не завершится. Конечно, эта проблема не нова и свойственна не только обучению с подкреплением; она имеет долгую историю и в литературе, и в технике. Например, в поэме Гёте «Ученик чародея» (Goethe, 1878) ученик воспользовался магией, что зачаровать метлу, заставив ее носить вместо него воду, но результатом стал неожиданный потоп, поскольку ученик плохо владел магией. Если говорить о технике, то Норберт Винер, основатель кибернетики, предупреждал об этой проблеме больше века назад, ссылаясь на сказку об обезьяньей лапе (Wiener, 1964): «...она дает то, о чем вы просите, а не то, о чем должны были бы попросить или что имели в виду» (стр. 59). В современном контексте эта проблема подробно обсуждалась в работе Nick Bostrom (2014). Любой имеющий опыт в области обучения с подкреплением, скорее всего, наблюдал, как созданные им системы находят неожиданные способы получить большое вознаграждение. Иногда неожиданное поведение – благо: оно решает задачу новым изящным способом. Но бывает, что открытия агента нарушают ограничения, о которых конструктор системы никогда бы не подумал. Тщательно продумывать сигналы вознаграждения абсолютно необходимо, если агент должен будет действовать в реальном мире, когда у человека не будет ни возможностей наложить запрет на его действия, ни средств остановить его.

Несмотря на возможность непредвиденных негативных последствий, оптимизация вот уже сотни лет применяется инженерами, архитекторами и другими людьми, проекты которых изменили мир к лучшему. Многочему, что есть хорошего в нашей окружающей среде, мы обязаны методам оптимизации. Было разработано немало подходов, сводящих к минимуму риск оптимизации, например наложение жестких и мягких ограничений, разрешение только надежных, учитывающих риск стратегий и оптимизация с несколькими целевыми функциями. Некоторые

из этих подходов адаптированы к обучению с подкреплением, но для разрешения всех сомнений необходимы дополнительные исследования. Гарантирование того, что цель агента обучения с подкреплением созвучна нашей собственной, – проблема, которая еще ждет своего решения.

Еще одна проблема, возникающая, когда агенты обучения с подкреплением действуют и обучаются в реальном мире, касается не того, чему они должны обучиться в итоге, а их поведения в процессе обучения. Как убедиться, что у агента достаточно опыта, чтобы обучиться качественной стратегии и при этом не нанести вред окружающей среде, другим агентам и самому себе (точнее, свести вероятность ущерба к приемлемо низкому уровню)? Эта проблема тоже не нова и не уникальна для обучения с подкреплением. Управление риском и сглаживание риска во встроенном обучении с подкреплением аналогично проблеме, с которой инженеры сталкиваются с того самого момента, как начали использовать автоматическое управление в ситуациях, когда поведение устройства управления может повлечь неприемлемые, возможно катастрофические, последствия, как при управлении самолетом или тонкими химическими процессами. Для практического применения управления необходимо скрупулезное моделирование системы и всесторонние испытания. Существует даже тщательно разработанная теория, направленная на обеспечение сходимости и устойчивости адаптивных контроллеров, предназначенных для работы в условиях, когда динамика управляемой системы известна не полностью. Теоретические гарантии никогда не являются незыблемыми, поскольку зависят от истинности исходных предположений, но без этой теории в сочетании с практическими методами управления и сглаживания рисков автоматическое управление – адаптивное и любое другое – не приносило бы такой пользы в части повышения качества, эффективности и рентабельности процессов, которую мы наблюдаем и на которую привыкли полагаться. Одно из самых насущных направлений будущих исследований по обучению с подкреплением – адаптировать и развить методы, разработанные в технике автоматического управления, с целью сделать безопасным полное встраивание агентов обучения с подкреплением в физическую окружающую среду.

В заключение вернемся к призыву Саймона осознать, что мы – творцы своего будущего, а не просто зрители. Своими личными решениями и воздействием, оказываемым на управление нашим обществом, мы можем способствовать тому, чтобы польза от новых технологий превышала вред, который они способны причинить. В случае обучения с подкреплением для этого открываются обширные возможности – оно может повысить качество жизни на нашей планете, сделать ее более справедливой и устойчивой, но может и принести новые беды. Угроза, которая уже нависла над нами, – исчезновение рабочих мест, вызванное пришествием искусственного интеллекта. И все же есть основательные причины полагать, что польза от внедрения искусственного интеллекта может перевесить его разрушительные последствия. Что же касается безопасности, то риски, присущие обучению с подкреплением, не так уж сильно отличаются от тех, с которыми мы научились успешно справляться в похожих приложениях оптимизации и методов автоматического управления. По мере выхода обучения с подкреплением в реальный мир разработчики обязаны будут придерживаться наработанных практик в схожих технологиях, не забывая развивать их, чтобы последнее слово всегда оставалось за Прометеем.

БИБЛИОГРАФИЧЕСКИЕ И ИСТОРИЧЕСКИЕ ЗАМЕЧАНИЯ

- 17.1** Общие функции ценности впервые были описаны Саттоном с сотрудниками (Sutton, 1995a; Sutton et al., 2011; Modayil, White and Sutton, 2013). В работе Ring (готовится к печати) поставлен обширный мысленный эксперимент с ОФЦ («прогнозы»), который оказал заметное влияние, хотя еще и не опубликован.

Первые демонстрации многоголового обучения в обучении с подкреплением приведены в работе Jaderberg et al. (2017). В работе Bellemare, Dabney and Munos (2017) показано, что предсказание дополнительных аспектов распределения вознаграждения может значительно ускорить обучение с целью оптимизации его математического ожидания – это пример вспомогательных задач. Многие другие ученые продолжили это направление исследований.

Насколько нам известно, общая теория классического обусловливания как обученных предсказаний в сочетании со встроенными рефлексивными реакциями на эти предсказания еще не получила внятного освещения в литературе по психологии. В работе Modayil and Sutton (2014) она описывается как подход к конструированию роботов и других агентов и называется «павловским управлением», чтобы подчеркнуть ее происхождение от классического обусловливания.

- 17.2** Формализация абстрагированного от времени порядка действий в виде опций впервые описана в работе Sutton, Precup, and Singh (1999) на основе предшествующих работ Parr (1998) и Sutton (1995a) и классических работ по полумарковским процессам принятия решений (см., например, Puterman, 1994). В докторской диссертации Precup (2000) идеи опций получили полное развитие. У этих ранних работ было важное ограничение – в них не рассматривался случай разделенной стратегии с аппроксимацией функций. В общем случае для внутриопционного обучения необходимо обучение с разделенной стратегией, но в то время надежного способа реализовать это для аппроксимации функций не было. Сейчас в нашем распоряжении много устойчивых методов обучения с разделенной стратегией, но на момент публикации этой книги их сочетание с идеями опций исследовано еще недостаточно. В работах Barto and Mahadevan (2003) и Hengst (2012) можно найти обзор формализма опций и другие подходы к абстрагированию от времени.

Применение ОФЦ для реализации моделей опций ранее не описывалось. В нашем изложении использован прием из работы Modayil, White and Sutton (2014) для предсказания сигналов в момент завершения стратегий.

Из других работ, в которых изучались модели опций с аппроксимацией функций, упомянем Sorg and Singh (2010) и Bacon, Harb, and Precup (2017).

Обобщение опций и моделей опций на постановку со средним вознаграждением еще не рассматривалось в литературе.

- 17.3** Хорошее изложение подхода на основе ЧНМППР имеется в работе Monahan (1982). ППС и тесты введены в работе Littman, Sutton and Singh (2002). ООМ

появились в работах Jaeger (1997, 1998, 2000). Последовательные системы, которые унифицируют ППС, ООМ и многие другие работы, впервые описаны в докторской диссертации Michael Thon (2017; Thon and Jaeger, 2015). Обобщение на сети временных связей есть в работах Tanner (2006), Sutton and Tanner, 2005, а затем распространено на опции (Sutton, Rafols, and Koop, 2006).

Теория обучения с подкреплением с немарковским представлением состояний в явном виде разработана в работах Singh, Jaakkola, and Jordan (1994; Jaakkola, Singh, and Jordan, 1995). Ранние подходы к частичной наблюдаемости в контексте обучения с подкреплением имеются в работах Chrisman (1992), McCallum (1993, 1995), Parr and Russell (1995), Littman, Cassandra, and Kaelbling (1995) и Lin and Mitchell (1992).

- 17.4 Первые попытки включить консультирование и преподавание в обучение с подкреплением предприняты в работах Lin (1992), Maclin and Shavlik (1994), Clouse (1996) и Clouse and Utgoff (1992). Формирование Скиннера не следует путать с техникой «формирования на основе потенциала», предложенной в работе Ng, Harada, and Russell (1999). В работе Wiewiora (2003) показано, что эта техника эквивалентна более простой идее построения хорошего начального приближения к функции ценности, как в (17.11).

- 17.5 Мы рекомендуем книгу Goodfellow, Bengio, and Courville (2016)¹, где обсуждаются методы современного глубокого обучения. Проблема катастрофической интерференции в ИНС исследована в работах McCloskey and Cohen (1989), Ratcliff (1990) и French (1999). Идея буферов воспроизведения выдвинута в работе (1992) и активно использовалась для глубокого обучения системы для игр Atari (раздел 16.5, Mnih et al., 2013, 2015).

Мински (Minsky, 1961) одним из первых поднял вопрос об обучении представлений.

Среди немногих работ, где рассматривается планирование с обученными приближенными моделями, отметим Kuvayev and Sutton (1996), Sutton, Szepesvári, Geramifard and Bowling (2008), Nouri and Littman (2009) и Hester and Stone (2012).

В области искусственного интеллекта хорошо известно о необходимости проявлять избирательность при конструировании модели с целью избежать замедления планирования. Из классических работ упомянем Minton (1990) и Tambe, Newell, and Rosenbloom (1990). В работе Hauskrecht, Meuleau, Kaelbling, Dean, and Boutilier (1998) этот эффект продемонстрирован в МППР с детерминированными параметрами.

В работе Schmidhuber (1991a, b) высказано предположение, что нечто, подобное любознательности, могло бы принести результат, если бы сигналы вознаграждения были функцией скорости улучшения, имеющей у агента модели окружающей среды. Функция полномочий, предложенная в работе Klyubin, Polani, and Nehaniv (2005), является теоретико-информационной мерой способности агента управлять своей средой, которая может функ-

¹ Бенджио Йошуа, Гудфеллоу Ян, Курвилль Аарон. Глубокое обучение. М.: ДМК Пресс, 2017.

ционировать как присущий сигнал вознаграждения. Сборник Baldassarre and Mirolli (2013) включает различные статьи на тему присущего вознаграждения и мотивации с биологической и вычислительной точек зрения, в т. ч. о «внутренне мотивированном обучении с подкреплением» (мы пользуемся термином, введенным в работе Singh, Barto, and Chentenez (2004)). См. также Oudeyer and Kaplan (2007), Oudeyer, Kaplan, and Hafner (2007), and Barto (2013).

Предметный указатель

К страницам, номера которых выделены *курсивом*, рекомендуется обращаться в первую очередь. На страницах, номера которых выделены полужирным шрифтом, приведен псевдокод алгоритмов.

A

AlphaGo, AlphaGo Zero, AlphaZero, 504
Atari, видеоигры, 498

B

BOXES, 45, 101, 282

D

Dyna-архитектура, 198, 201

E

Emphatic-TD методы, 368
с разделенной стратегией, 330
Expected Sarsa, 167

K

k-рукые бандиты, 51

N

n-шаговые методы, 176
 $Q(\sigma)$, **192**
Sarsa, **181, 292**
дифференциальный, **301**
с разделенной стратегией, **185**
TD, **179**
обновления по дереву, **190**
усеченные λ -доходные, 346

Q

Q-обучение, 49, 165
двойное, 171
Q-планирование, **197**
 $Q(\lambda)$ Уоткинса, 364
 $Q(\sigma)$, 190, **192**

R

R-обучение, 303
REINFORCE, 378, **380**
с базой, **382**

S

Sarsa, 162, **163, 289**
Expected, 167, 175
двойной, 171
n-шаговый, 183
n-шаговый с разделенной стратегией, 185
n-шаговый, 181, **292**
дифференциальный, **301**
с разделенной стратегией, **185**
дифференциальный одношаговый, **296**
сравнение с Q-обучением, 166, 167
Sarsa(λ), 354, **356**
истинно онлайнный, **358**
softmax, 374, 381, 389, 457, 508
в задаче о бандитах, 64, 72

T

TD-ошибка, 154
в случае аппроксимации, 318
дифференциальная, 296
n-шаговая, 301
TD-Gammon, 481
TD(λ), 342, **343**
истинно онлайнный, 350, **351**
усеченный, 346, 348

W

Watson (программа для игры Jeopardy), 489

A

Абстрагирование времени, 523
Агрегирование состояний, 245
Алгоритм бандита (ghjcnjq), **58**
Алгоритм всех действий, 379
Алгоритм наименьших средних квадратов, 328, 353
Алгоритм с остаточным градиентом, 320, 325
наивный, 319

Алгоритм TD наименьших квадратов (LSTD), 272
 Аппроксимация с помощью ядерных функций, 276
 Аппроксимация стратегии, 374
 Аппроксимация функции ценности, 239
 Аппроксимация функций, 236
 Априорная информация, 38, 60, 281, 376
 Аренда автомобилей, пример, 110, 173, 252
 Асинхронное динамическое программирование, 115, 119
 Ассоциативное обучение с подкреплением, 72, 479
 Ассоциативный поиск, 68

Б

База, 381, 391
 Безопасность, 496, 542
 Беллман Ричард, 40, 101, 120
 Беллмана оператор, 315
 Беллмана уравнение, 40
 для v_* , 87
 для дифференциальных функций ценности, 295
 для оптимальных функций ценности v_* и q_* , 92
 для опций, 525
 Беллмановская ошибка, 316, 318, 320
 вектор ошибок, 316
 обучаемость, 322
 Бинарные признаки, 257, 265, 291, 356
 Биореактор, пример, 78
 Блуждание на краю обрыва, пример, 166, 167
 Блэкджек, пример, 124, 131, 138
 Бутстрэппинг, 119, 231, 360, 383
 n -шаговый, 176, 301
 в психологии, 398, 402, 407, 408
 и аппроксимация функций, 250, 312
 и динамическое программирование, 119
 и методы Монте-Карло, 126
 и устойчивость, 310
 оценка, 157, 294, 312, 341

В

Воспроизведение опыта, 502
 Вспомогательные задачи, 522, 531, 538
 Вторичное подкрепление, 47, 398, 408, 424
 Выборка по значимости, 135, 187, 305
 взвешенная и обыкновенная, 137
 и бесконечная дисперсия, 139
 инкрементная реализация, 142
 и следы приемлемости, 361
 коэффициент, 136, 184, 305

 приведенная, 147
 с учетом обесценивания, 146
 Выборочное и полное обновление, 154, 208

Г

Гедонистические нейроны, 460
 Генетические алгоритмы, 46
 Гипотеза об ошибке предсказания вознаграждения, 437, 443
 Гиттинса индекс, 70
 Глубокое обучение, 267, 503, 536
 Глубокое обучение с подкреплением, 281
 Глубокое остаточное обучение, 271
 Градиент, 243
 Градиентные TD-методы, 327, 367
 Грубое кодирование, 257, 282

Д

Двойное обучение, 169, 175
 Детеныш газели, пример, 30
 Диаграмма предшествующих состояний, 88, 174
 для Expected Sarsa, 168
 для n -шагового Expected Sarsa, 182
 для n -шагового $Q(\sigma)$, 191
 для n -шагового Sarsa, 182
 для n -шагового TD, 177
 для Q -обучения, 168
 для $Q(\lambda)$, 365
 для Sarsa, 163
 для Sarsa(λ), 355
 для TD(0), 154
 для TD(λ), 340
 для Tree Backup(λ), 366
 для динамического программирования, 88, 90, 93, 211
 для методов Монте-Карло, 126
 для составного обновления, 339
 для усеченного TD(λ), 347
 для шашечной программы Сэмюэла, 488
 Дилемма исследования-использования, 28, 135, 536
 Динамическое программирование, 40, 102, 213, 309
 и аппроксимация функций, 286
 и искусственный интеллект, 120
 и опции, 525
 Динамическое программирование в реальном времени (ДПРВ), 216
 Доступ с очередностью, пример, 297
 Дофамин, 432, 437, 473
 и наркозависимость, 468
 Доход, 82

λ-доход, 338
 усеченный, 346
n-шаговый, 178
 для методов с аппроксимацией функций, 252
 для методов с переменным управлением, 186
 для обновления по дереву, 189
 для ценностей действий, 181
 для Expected Sarsa, 183
 для $Q(\sigma)$, 191
 дифференциальный, 295, 301, 386
 плоский частичный, 146
 с зависимым от состояния завершением, 360

З

Заблокированный лабиринт, пример, 203
 Завтрак, пример, 30, 49
 Задача игрока, пример, 113
 Задача о машине на горе, пример, 290, 357
 Задача о перемещении стержня, пример, 209
 Задача о бандитах, 51
 Закон эффекта, 42, 72, 395, 411, 477

И

Игра в гольф, пример, 89, 96
 Идентификация системы, 418
 Избирательная адаптация с бутстрэппингом, 284
 Инструментальное обусловливание, 410.
 См. также Закон эффекта
 и мотивация, 414
 проблемные ящики Торндайка, 411
 Интерфейс между агентом и окружающей средой, 74, 529
 Искусственные нейронные сети (ИНС), 267, 283, 453, 490, 498, 536
 Искусственный интеллект, 26, 535, 542
 Исполнитель–критик, 48, 373, 383, 391, 465
 нейронный, 452
 одношаговый (эпизодический), 384
 со следами приемлемости (непрерывный), 386
 со следами приемлемости (эпизодический), 385
 Исследовательские старты, 128, 130, 217
 Итерация по стратегиям, 40, 109, 110
 Итерация по ценности, 112, 113

К

Катастрофическая интерференция, 536
 Кибернетика, 541

Классическое обусловливание, 48, 395
 TD-модель, 401
 блокировка, 425
 и обусловливание высшего порядка, 397
 задержка и следовое обусловливание, 396
 модель Рескорлы–Вагнера, 399
 Когнитивные карты, 416
 Коллективное обучение с подкреплением, 462
 Кольцевые гонки, упражнение, 145
 Конструирование признаков, 252
 Контекстуальные бандиты, 68
 Контпример Бэрда, 308, 330, 332, 335
 Контпример Цициклиса и ван Роя, 311
 Коэффициент ветвления, 212, 482
 Крестики-нолики, 34, 44, 172
 Критик, 45, 283, 398, 477
 Кумулянт, 521

Л

Латентное обучение, 233, 417, 420
 Линейная аппроксимация функций, 246, 314
 Любознательность, 538

М

Марковский процесс вознаграждения (МПВ), 158
 Марковский процесс принятия решений, 27, 40, 74
 Марковское свойство, 76, 149, 527
 Метод выборочного среднего, 53
 Метод проб и ошибок, 32, 42, 462
 Методы градиента стратегии, 373
 REINFORCE, 380, 382
 исполнитель–критик, 384, 386
 Методы Монте-Карло, 122
 алгоритм первого посещения для предсказания, 124
 алгоритм первого посещения для управления, 133
 алгоритм управления с разделенной стратегией, 143, 144
 градиентный метод для оценивания *vp*, 244
 Монте-Карло с исследовательскими стартами, 131
 первого и всех посещений, 123
 предсказание с разделенной стратегией, 135, 143
 Методы с единой стратегией, 132
n-шаговые, 179, 181
 Sarsa, 162, 163
 TD(0), 152, 153

исполнитель–критик, **384, 386**
 Монте-Карло, 132, **133, 380, 382**
 приближенные управления, **289, 292, 296, 301**
 со следами приемлемости, **343, 351, 356, 358**

Методы с разделенной стратегией, 304

Emphatic-TD(λ), 368
 Expected Sarsa, 167
 GQ(λ), 367
 GTD(λ), 367
 HTD(λ), 367
 n -шаговые, 184
 n -шаговый Q(σ), **192**
 n -шаговый Sarsa, **185**
 n -шаговый обновления по дереву, **190**
 Q(λ), 364
 Q-обучение, **165**
 Tree-Backup(λ), 365
 и методы с единой стратегией, 132, 135
 и следы приемлемости, 361
 Монте-Карло, 135
 уменьшение дисперсии, 332
 Методы ценности действий, 373
 в задачах о бандитах, 53
 МК с постоянным α , 153
 Многорукые бандиты, 51
 Модели распределения, 195, 226
 Модель окружающей среды, 32, 195
 Мотивация, 414
 Мыльный пузырь, пример, 127

Н

Нарды, 222, 481
 Наркотическая зависимость, 468
 Нейродинамическое программирование, 41
 Нейронауки, 29, 49, 432
 Нейроэкономика, 472, 480
 Непрерывное действие, 102, 289, 388
 Непрерывное состояние, 102, 266, 282
 Непрерывные задачи, 82, 84, 100, 157, 294, 344
 Нестационарность, 56, 59, 68, 302

О

Обесценивание, 82, 241, 288, 331, 377, 487
 в задаче о балансировании стержня, 84
 возражения, 299
 зависящее от состояния, 359
 Обновления по дереву алгоритм
 n -шаговый, 188, **190**

Tree-Backup(λ), 365
 Обобщенная итерация по стратегиям (ОИС), 116, 123, 129, 173, 230
 Обозначения, 20
 Обратное представление следов приемлемости, 338, 344
 Обратное распространение, 48, 269, 284, 465, 484, 498
 Обучение без учителя, 27, 270
 Обучение на основе временных различий, 35, 152
 n -шаговое, 176, **179, 251**
 TD(0), **153, 244**
 TD(1), 344
 TD(λ), 342, **343**
 истинно онлайнный, 350, **351**
 λ -доходные методы
 онлайнные, 348
 офлайнные, 341
 усеченные, 346
 история, 47
 оптимальность, 159
 преимущества, 157
 Обучение представлений, 536
 Обучение с подкреплением, 26
 Обучение с учителем, 27, 44, 239
 Общие функции ценности (ОФЦ), 521, 538
 Окружающая среда, 74
 Оперантное обусловливание.
 См. Инструментальное обусловливание
 Оптимальное управление, 40, 48
 Оптимизация управления памятью, 492
 Оптимистические начальные значения, 60, 233
 Оптимумы локальные и глобальные, 241
 Опции, 524
 модели опций, 524
 Оси методов обучения с подкреплением, 230
 Основанные на модели и безмодельные методы, 32, 195
 в нейронауках, 466
 в обучении животных, 417
 Отложенное вознаграждение, 27, 74, 294
 Отложенное подкрепление, 415
 Оценивание стратегии, 103.
 См. также Предсказание
 итеративное, 105
 Оценка максимального правдоподобия, 161
 Оценочная обратная связь, 51, 74

П

Павловское обусловливание.
 См. Классическое обусловливание

Павловское управление, 395, 425, 543
 Параметр затухания следа (λ), 337, 343
 зависящий от состояния, 359
 Парение в восходящих потоках
 воздуха, 516
 Переменное управление, 186, 192, 330
 и следы приемлемости, 361
 Персонализация таких веб-служб, 513
 Планирование, 31, 32, 37, 173, 195, 417
 в психологии, 417
 с обученными моделями, 198, 537
 с опциями, 523, 525
 Пластичность, зависящая от момента
 времени спайка (SDTP), 459
 Плиточное кодирование, 260, 266, 282, 291,
 292, 495
 Поведенческая стратегия, 135, 143
 Поглощающее состояние, 85
 Подкорковые ядра, 442
 Поездка домой, пример, 155
 Пожарной цепочки алгоритм, 46, 48, 174
 Поиск по дереву методом Монте-Карло
 (ПДМК), 226
 Полиномиальный базис, 253
 Полное обновление, 104, 210, 231
 Полугradientные методы, 244, 305
 Послесостояния, 172, 175, 221, 222, 232, 484,
 490
 Постановка со средним
 вознаграждением, 294, 306, 526
 Предпочтения действий, 374, 381, 389, 518
 в задачах о бандитах, 64, 69
 Предсказание, 103
 и аппроксимацией, 238
 и управление, 394
 методами Монте-Карло, 123
 с разделенной стратегией, 135
 TD-методами, 152
 Приближенное динамическое
 программирование, 41
 Привычное и целеустремленное
 поведение, 418
 Пример управления доступом
 с очередностью, 303
 Приоритетный проход, 206, **208**
 Программа игры в шашки Сэмюэла, 286, 486
 Проклятие размерности, 29, 40, 264, 276
 Проксимальные TD-методы, 336
 Проходы, 104, 197
 Прямое и косвенное обучение
 с подкреплением, 198, 233
 Псевдозавершение, 331, 360
 Психология, 29, 40, 46, 393

Р

Радиально-базисные функции (РБФ), 265
 Размер шага, 35, 58, 153, 158, 159
 Разыгрывающие алгоритмы, 224
 Распределение поощрения, 44, 46, 74, 344,
 459
 в психологии, 398, 415
 структурное, 442, 463, 466
 Распределение с единой стратегией, 214,
 240, 250, 305, 310, 331
 Робот-уборщик, пример, 79

С

Самообучающиеся автоматы, 45
 Свойство уменьшения ошибки, 180, 339
 Сеточный мир, примеры, 88, 94, 105, 183
 блуждание на краю обрыва, 166
 ветренный, 164
 заблокированный лабиринт, Дупа, 203
 короткий путь в лабиринте, Дупа, 204
 лабиринт, Дупа, 201
 Сигнал вознаграждения, 26, 31, 75, 80, 414,
 435, 455
 и сигнал подкрепления, 428, 435
 присущий, 538
 проектирование, 532, 541
 разреженный, 532
 Сигнал подкрепления, 435
 Синаптическая пластичность, 434
 двух- и трехфакторная, 458
 хеббовская, 458
 Системы классификации, 46, 48
 Слабые и сильные методы, 29
 Следы приемлемости, 337, 402, 416, 456
 голландские, 352
 замещающие, 352, 358
 контингентные/неконтингентные, 457, 470
 накапливающиеся, 352, 358, 362
 с разделенной стратегией, 361
 Случайное блуждание, 127
 с 19 состояниями, 180, 341
 результаты TD(λ), 345, 350
 с 1000 состояний, 245, 260
 базис Фурье и полиномиальный
 базис, 257
 с пятью состояниями, 158
 Смертельная триада, 312
 Смещение максимизации, 169
 Составное обновление, 339, 371
 Составной стимул, 397, 399, 425, 438

Состояние, 33, 76
 доверительное, 529
 и наблюдения, 526
 марковское свойство, 527
 история k -го порядка, 531
 наблюдаемые операторные модели (ООМ), 530
 представления прогностических состояний, 530
 скрытое, 529
 функция обновления состояния, 528
 частично наблюдаемые МППР, 40, 529
 Спроецированная беллмановская ошибка, 334
 вектор, 315, 317
 Среднеквадратическая беллмановская ошибка, 316
 ошибка дохода, 324
 ошибка ценности, 240
 спроецированная беллмановская ошибка, 317
 TD-ошибка, 318
 Среднеквадратическая ошибка (СКО), 159
 Стохастическая аппроксимация, условия сходимости, 59
 Стохастический градиентный спуск (СГС), 242
 по беллмановской ошибке, 318
 Стохастически эквивалентная оценка, 161
 Стратегия, 31, 68, 86
 иерархическая, 524
 мягкая и ϵ -мягкая, 133, 143

Т

Табличные методы решения, 50
 Теорема о градиенте стратегии, 376
 доказательство для непрерывного случая, 387
 доказательство для эпизодического случая, 377
 Теория игр, 46
 Теория управления, 29
 Томпсона выборка, 70, 73
 Траекторная выборка, 213

У

Удовольствие и неудовольствие, 32, 43, 473
 Улучшение стратегии, 107
 теорема, 107, 134
 Управление и предсказание, 394
 Управление поиском, 199
 Усреднители, 311

Ф

Формирование, 413, 533
 Функции завершения, 359, 521
 Функция обновления состояния, 528
 Функция ценности, 32, 86
 действий, 86, 92, 94, 101, 162, 165
 действий приближенная: $\hat{q}(s, a, w)$, 288
 дифференциальная, 288
 и эволюционные методы, 36
 приближенная: $\hat{v}(s, w)$, 238
 при заданной стратегии: v_π и q_π , 86
 при оптимальной стратегии: v_* и q_* , 91
 Фурье базис, 254

Ц

Целевая стратегия, 135, 143
 Цель обновления, 58, 178, 239
 Ценность, 32, 52, 74

Ч

Частично наблюдаемый МППР (ЧНМППР), 529

Ш

Шахматы, 30, 47, 81, 223, 513

Э

Эволюционные методы, 33, 36, 46
 Эволюция, 33, 428, 534
 Эвристический поиск, 221, 232
 в TD-Gammon, 485
 в шашечной программе Сэмюэла, 487
 как последовательность обновлений, 223
 Эпизоды, эпизодические задачи, 37, 82, 122
 Эргодичность, 295

Книги издательства «ДМК ПРЕСС»
можно купить оптом и в розницу
в книготорговой компании «Галактика»
(представляет интересы издательств
«ДМК ПРЕСС», «СОЛОН ПРЕСС», «КТК Галактика»).

Адрес: г. Москва, пр. Андропова, 38;
тел.: **(499) 782-38-89**, электронная почта: **books@aliens-kniga.ru**.
При оформлении заказа следует указать адрес (полностью),
по которому должны быть высланы книги;
фамилию, имя и отчество получателя.

Желательно также указать свой телефон и электронный адрес.
Эти книги вы можете заказать и в интернет-магазине: **www.a-planeta.ru**.

Ричард С. Саттон, Эндрю Дж. Барто

Обучение с подкреплением **Введение**

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com
Перевод *Слинкин А. А.*
Корректор *Синяева Г. И.*
Верстка *Чаннова А. А.*
Дизайн обложки *Мовчан А. Г.*

Формат 70 × 100 1/16.
Гарнитура «PT Serif». Печать офсетная.
Усл. печ. л. 44,85. Тираж 500 экз.

Веб-сайт издательства: **www.dmkpress.com**