Простые рекуррентные нейронные сети

Сеть Элмана

Рекуррентную нейронную сеть Элмана так же называют Simple Recurrent Neural Network (Simple RNN, SRN).

Существуют актуальные задачи обработки данных, при решении которых требуется анализировать последовательности объектов, при этом порядок следования объектов играет существенную роль в задаче. Например, это задача распознавания речи, где рассматриваются последовательности звуков или задачи обработки текстов на естественном языке, где анализируются последовательности слов.

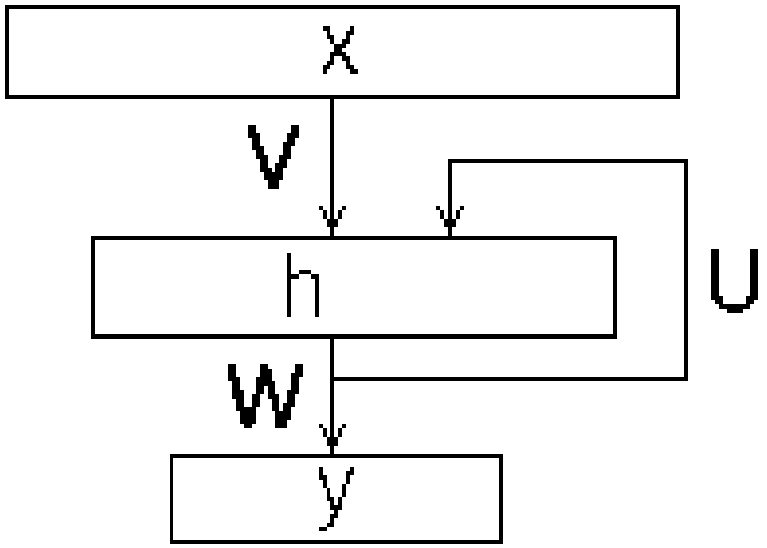
Для решения такого рода задач можно применять рекуррентные нейронные сети, которые в процессе работы могут сохранять информацию о своих предыдущих состояниях.

Простые рекуррентные нейронные сети

Сеть Элмана

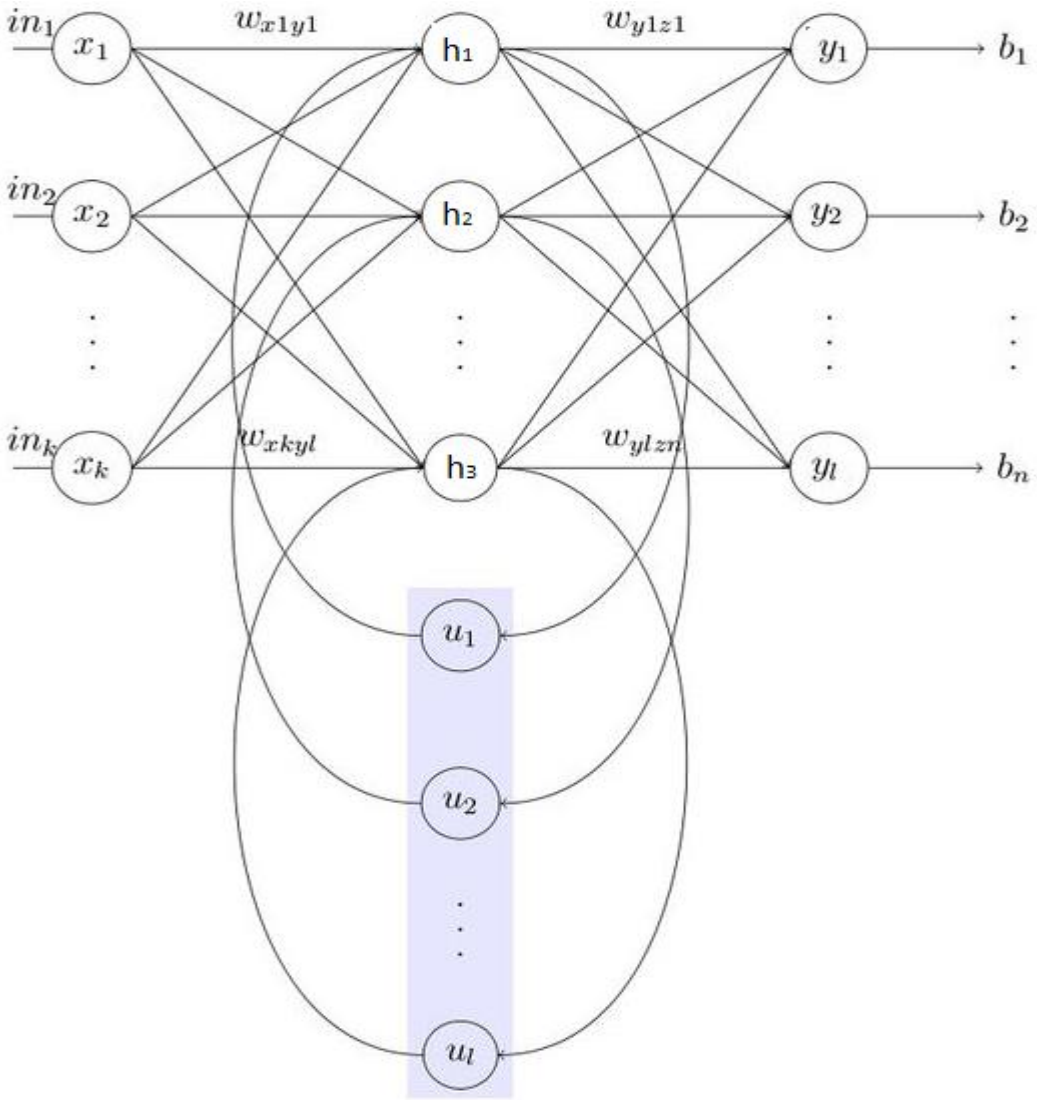
Искусственная нейронная сеть Элмана состоит из трех слоёв - входного (распределительного) слоя, скрытого и выходного (обрабатывающего). При этом скрытый слой имеет обратную связь сам на себя.

Схема нейронной сети Элмана имеет вид:



Простые рекуррентные нейронные сети
Сеть Элмана

Общий вид сети Элмана:



Простые рекуррентные нейронные сети

Сеть Элмана

В отличии от обычной сети прямого распространения, входной образ рекуррентной сети это не один вектор, а последовательность векторов {x(1),…x(n)}. Векторы входного образа в заданном порядке подаются на вход, при этом новое состояние скрытого слоя зависит от его предыдущих состояний. Сеть Элмана можно описать следующими соотношениями:

$$h(t) = f\left(V \cdot x(t) + U \cdot h(t-1) + b_h\right)$$

$$y(t) = g\left(W \cdot h(t) + b_y\right)$$

x(t) - входной вектор, h(t) - состояние скрытого слоя для входа x(t) (h(0)=0), y(t) - выход сети для входа x(t), U - весовая матрица распределительного слоя, W - весовая (квадратная) матрица обратных связей скрытого слоя, b_h - вектор сдвигов скрытого слоя, V- весовая матрица выходного слоя, b_y - вектор сдвигов выходного слоя, f - функция активации скрытого слоя, g - функция активации выходного слоя.

Простые рекуррентные нейронные сети

Сеть Элмана

В зависимости от того как сформировать вход и выход рекуррентной сети, можно разными способами задать схему её работы. Развернём схему рекуррентной сети во времени:

a)"**много в один**" (many-to-one) - скрытый слой последовательно изменяет своё состояние, из его конечного состояния вычисляется выхо сети, используется для классификации текстов;

b) "**один во много**" (one-to-many) - скрытый слой инициализируется одним входом, из цепочки его последующих состояний генерируются выходы сети, используется для аннотирования изображений;

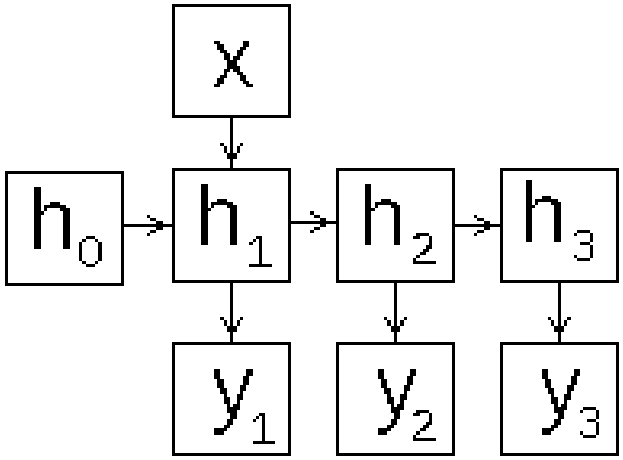
c) "**много во много**" (many-to-many) - на каждый вход сеть выдаёт выход который зависит от предыдущих входов, используется для классификации видео;

d) "**много во много**" (many-to-many) - скрытый слой последовательно изменяет своё состояние, его конечное состояние служит инициализацией для выдачи цепочки результатов, используется для создания систем машинного перевода и чат-ботов.

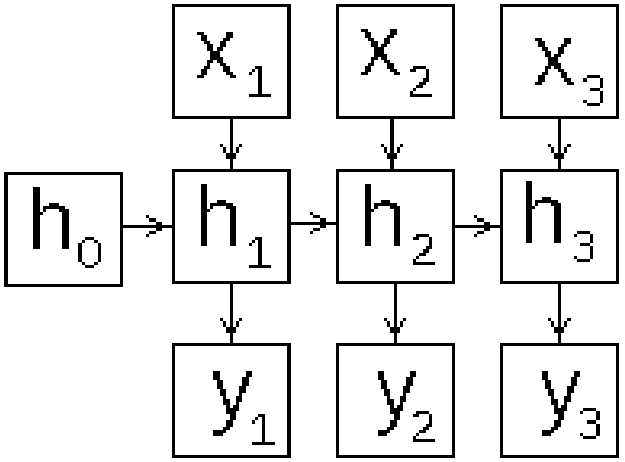
Простые рекуррентные нейронные сети

Сеть Элмана

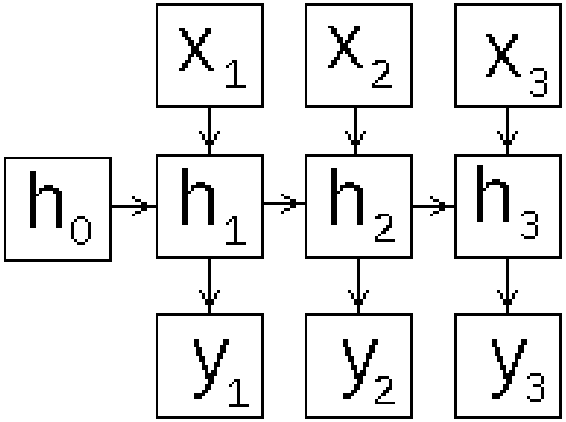
a)



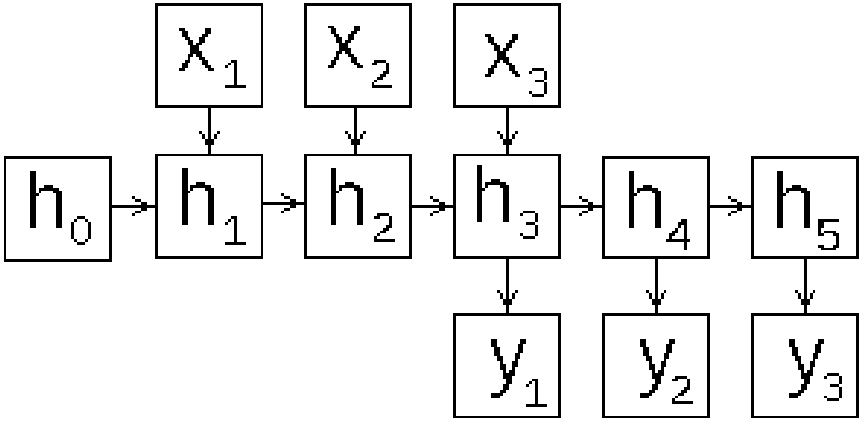
b)



c)



d)



Простые рекуррентные нейронные сети

Сеть Элмана

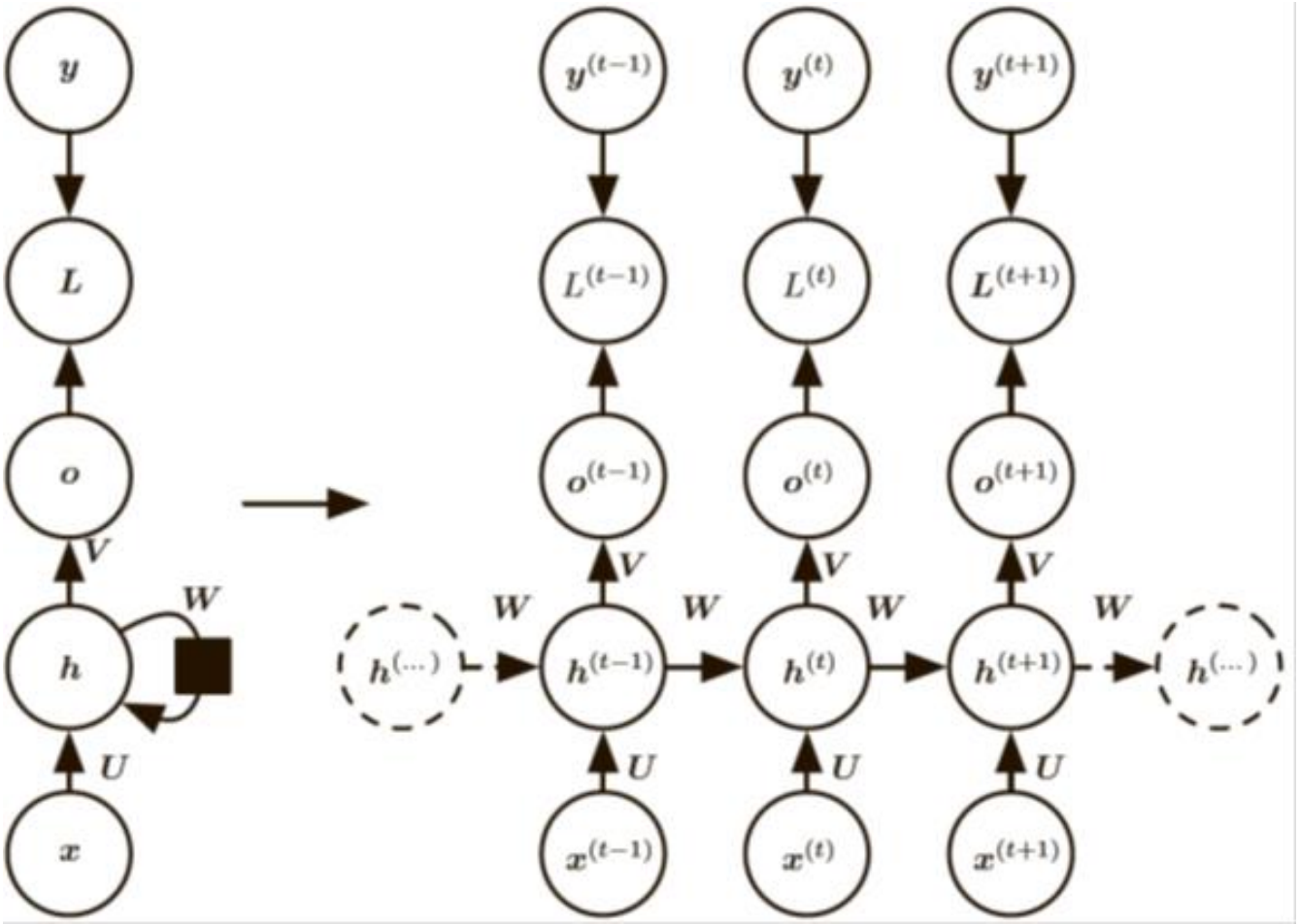
Рассмотрим метод обучения рекуррентной нейронной сети Элмана по схеме many-to-one для реализации классификатора объектов, заданных последовательностями векторов. Для обучения сети Элмана применяются те же градиентные методы, что и для обычных сетей прямого распространения, но с определёнными модификациями для корректного вычисления градиента функции ошибки. Он вычисляется с помощью модифицированного метода обратного распространения, который носит название Backpropagation through time (метод обратного распространения с разворачиванием сети во времени, BPTT).

Идея метода - развернуть последовательность, превратив рекуррентную сеть в "обычную.

Простые рекуррентные нейронные сети

Метод обучения рекуррентной нейронной сети (BPTT)

Рассмотрим рекуррентную нейронную сеть и ее функцию потерь, представленную в виде рекуррентных связей (слева). (Справа) То же в виде развернутого во времени графа вычислений, в котором каждая вершина ассоциирована с одним моментом времени.



Простые рекуррентные нейронные сети

Метод обучения рекуррентной нейронной сети (BPTT)

Представленный граф – это граф вычислений потерь при обучении рекуррентной сети, которая отображает входную последовательность значений $\boldsymbol{x}$ в соответствующую выходную последовательность значений $\boldsymbol{o}$. Функция потерь L измеряет, насколько далеко каждый элемент $\boldsymbol{o}$ отстоит от соответствующей метки $\boldsymbol{y}$. В случае применения к выходам функции softmax можно предполагать, что $\boldsymbol{o}$ – ненормированные логарифмические вероятности. Внутри функция L вычисляет softmax($\boldsymbol{o}$) и сравнивает эту величину с меткой $\boldsymbol{y}$. В РНС имеются связи между входным и скрытым слоями, параметризованные матрицей весов $\boldsymbol{U}$, рекуррентные связи между скрытыми блоками, параметризованные матрицей весов $\boldsymbol{W}$, и связи между скрытым и выходным слоями, параметризованные матрицей весов $\boldsymbol{V}$.

Простые рекуррентные нейронные сети
Метод обучения рекуррентной нейронной сети (BPTT)

Рассмотренная сеть универсальна в том смысле, что любая функция, вычислимая машиной Тьюринга, может быть вычислена и такой рекуррентной сетью конечного размера. Результат можно прочитать из РНС после выполнения числа шагов, асимптотически линейно зависящего от числа временных шагов машины Тьюринга и от длины входной последовательности. Когда РНС используется как машина Тьюринга, она принимает на входе двоичную последовательность, а ее выходы можно дискретизировать для получения двоичного результата. «Входом» машины Тьюринга является спецификация вычисляемой функции, поэтому той же сети, которая моделирует машину Тьюринга, достаточно для решения всех задач. Теоретическая РНС, применяемая в доказательстве, умеет моделировать неограниченный стек, представляя его активации и веса рациональными числами неограниченной точности.

Простые рекуррентные нейронные сети
Метод обучения рекуррентной нейронной сети (BPTT)

Теперь выведем уравнения прямого распространения для РНС, изображенной на рисунке. Будем предполагать, что в качестве функции активации используется гиперболический тангенс. Кроме того, будем предполагать, что выход дискретный, как если бы РНС применялась для предсказания слов или символов. Естественный способ представления дискретных величин – рассматривать выход как ненормированные логарифмические вероятности каждого возможного значения дискретной величины. Тогда на этапе постобработки можно применить операцию softmax и получить вектор $\hat{y}$ нормированных вероятностей. Прямое распространение начинается с задания начального состояния $h^{(0)}$.

Простые рекуррентные нейронные сети
Метод обучения рекуррентной нейронной сети (BPTT)

Затем для каждого временного шага от t = 1 до t = τ применяем следующие уравнения обновления:

$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} & \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) & \hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)}), \end{aligned} \qquad (1)$$

где параметрами являются векторы смещения $\boldsymbol{b}$ и $\boldsymbol{c}$, а также матрицы весов $\boldsymbol{U}$, $\boldsymbol{V}$ и $\boldsymbol{W}$ соответственно для связей между входным и скрытым слоями, между скрытым и выходным слоями и между скрытыми блоками. Это пример рекуррентной сети, которая отображает входную последовательность на выходную той же длины. Полная потеря для данной входной последовательности $\boldsymbol{x}$ в совокупности с последовательностью меток $\boldsymbol{y}$ равна сумме потерь по всем временным шагам.

Простые рекуррентные нейронные сети
Метод обучения рекуррентной нейронной сети (BPTT)

Например, если L(t) – отрицательное логарифмическое правдоподобие y(t) при условии **x**(1), …, **x**(t), то

$$L(\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)}\}) = \sum_t L^{(t)} = -\sum_t \log p_{\text{model}}(y^{(t)} \mid \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\}), \quad (2)$$

где $p_{\text{model}}(y^{(t)} \mid \{x^{(1)}, \ldots, x^{(t)}\}$ берется из элемента $y^{(t)}$ выходного вектора модели $\hat{\boldsymbol{y}}^{(t)}$. Вычисление градиента этой функции потерь относительно параметров – дорогая операция. Для вычисления градиента необходимо выполнить прямое распространение, двигаясь слева направо по развернутому графу, а затем обратное распространение, двигаясь справа налево по тому же графу. Время работы имеет порядок $O(\tau)$, и его нельзя уменьшить за счет распараллеливания, потому что граф прямого распространения принципиально последовательный.

Простые рекуррентные нейронные сети
Метод обучения рекуррентной нейронной сети (BPTT)

Состояния, вычисленные во время прямого прохода, необходимо хранить до повторного использования на обратном проходе, поэтому объем потребляемой памяти также имеет порядок $O(\tau)$.

Применим обобщенный алгоритм обратного распространения к развернутому графу вычислений. Градиенты, полученные в результате обратного распространения, можно затем использовать в сочетании с любым универсальным градиентным методом для обучения РНС.

Рассмотрим пример вычисления градиентов для уравнений (1) и (2) РНС. В рассматриваемом графе вычислений имеются параметры $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{W}$, $\boldsymbol{b}$ и $\boldsymbol{c}$, а также последовательность вершин, индексированных временем t . Для каждой вершины **N** можно рекурсивно вычислить градиент, зная градиенты, вычисленные в вершинах, следующих за ней в графе. Рекурсия начинается с вершин, непосредственно предшествующих окончательной потере: $\frac{\partial L}{\partial L^{(t)}} = 1$.

Простые рекуррентные нейронные сети
Метод обучения рекуррентной нейронной сети (BPTT)

Предполагается, что выходы $o^{(t)}$ используются в качестве аргумента функции softmax для получения вектора вероятностей выходов $\hat{y}$. Предполагается также, что функция потерь – это отрицательное логарифмическое правдоподобие истинной метки $y^{(t)}$ при известных к этому моменту входах. Градиент $\nabla o(t)L$ по выходам в момент t для всех i, t имеет вид:

$$(\nabla_{o^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - 1_{i,y^{(t)}}.$$

Мы движемся в направлении от конца последовательности к началу. В последний момент времени τ у $h^{(\tau)}$ есть только один потомок, поэтому вычислить градиент можно: $\nabla_{h^{(\tau)}} L = V^\top \nabla_{o^{(\tau)}} L.$

Затем можно совершать итерации назад во времени для обратного распространения градиентов от $t = \tau - 1$ до $t = 1$. Заметим, что потомками $h^{(t)}$ (для $t < \tau$) являются $o^{(t)}$ и $h^{(t+1)}$.

Простые рекуррентные нейронные сети
Метод обучения рекуррентной нейронной сети (BPTT)

Следовательно, градиент равен

$$\nabla_{\boldsymbol{h}^{(t)}} L = \left(\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}\right)^{\top} (\nabla_{\boldsymbol{h}^{(t+1)}} L) + \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\right)^{\top} (\nabla_{\boldsymbol{o}^{(t)}} L)$$

$$= \boldsymbol{W}^{\top} (\nabla_{\boldsymbol{h}^{(t+1)}} L) \operatorname{diag}(1 - (\boldsymbol{h}^{(t+1)})^2) + \boldsymbol{V}^{\top} (\nabla_{\boldsymbol{o}^{(t)}} L),$$

где $\operatorname{diag}(1 - (h^{(t+1)})^2)$ – диагональная матрица с элементами $1 - (h_i^{(t+1)})^2$. Это якобиан функции гиперболического тангенса, ассоциированный со скрытым блоком i в момент t + 1. Получив градиенты по внутренним вершинам графа вычислений, мы можем затем получить градиенты по вершинам параметров. Поскольку параметры разделяются между временными шагами, следует аккуратно подходить к обозначениям аналитических операций с участием этих переменных. Необходимо вычислять вклад в градиент одного ребра графа вычислений.

Простые рекуррентные нейронные сети
Метод обучения рекуррентной нейронной сети (BPTT)

Но оператор $\nabla_W f$, , применяемый в математическом анализе, принимает во внимание вклад $\boldsymbol{W}$ в значение f, вносимый *всеми* ребрами графа вычислений. Для разрешения этой неоднозначности введем фиктивные переменные $\boldsymbol{W}(t)$, определенные как копии $\boldsymbol{W}$, только каждая $\boldsymbol{W}(t)$ используется лишь на временном шаге t. Тогда $\nabla_{W^{(t)}} f$ можно использовать для обозначения вклада весов на шаге t в градиент. В этой нотации градиенты по остальным параметрам имеют вид

$$\nabla_{\boldsymbol{c}} L = \sum_t \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}} \right)^{\top} \nabla_{o^{(t)}} L = \sum_t \nabla_{o^{(t)}} L,$$

$$\nabla_{\boldsymbol{b}} L = \sum_t \left(\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}^{(t)}} \right)^{\top} \nabla_{h^{(t)}} L = \sum_t \operatorname{diag}(1 - (\boldsymbol{h}^{(t)})^2) \nabla_{h^{(t)}} L,$$

$$\nabla_{\boldsymbol{V}} L = \sum_t \sum_t \left(\frac{\partial L}{\partial o_i^{(t)}} \right) \nabla_{\boldsymbol{V}} o_i^{(t)} = \sum_t (\nabla_{o^{(t)}} L) \boldsymbol{h}^{(t)\top},$$

$$\nabla_{\boldsymbol{W}} L = \sum_t \sum_t \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{W^{(t)}} h_i^{(t)}$$

Простые рекуррентные нейронные сети
Метод обучения рекуррентной нейронной сети (BPTT)

$$= \sum_t \operatorname{diag}\left(1 - (\boldsymbol{h}^{(t)})^2\right) (\nabla_{\boldsymbol{h}^{(t)}} L) \boldsymbol{h}^{(t-1)\top},$$

$$\nabla_U L = \sum_t \sum_t \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{U^{(t)}} h_i^{(t)}$$

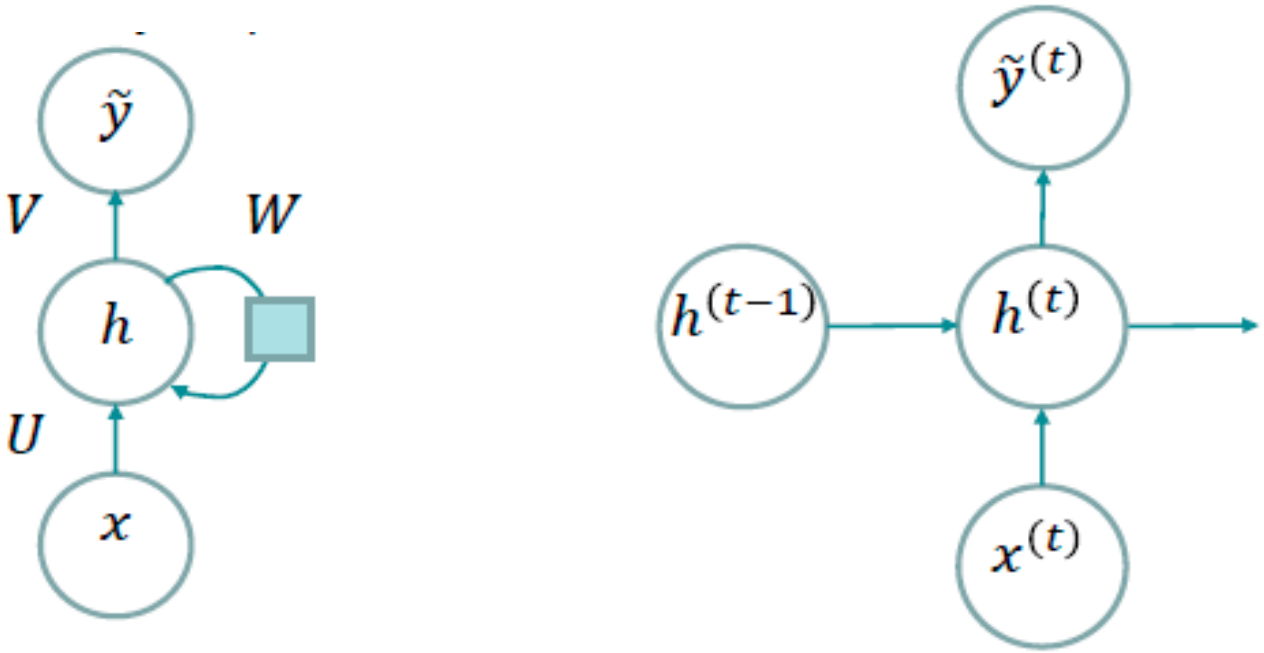
$$= \sum \operatorname{diag}\left(1 - (\boldsymbol{h}^{(t)})^2\right) (\nabla_{\boldsymbol{h}^{(t)}} L) \boldsymbol{x}^{(t)\top}.$$

Нам не нужно вычислять градиент по **x**(t) для обучения, потому что среди его предков в графе вычислений, определяющем потерю, нет параметров.

Глубокие рекуррентные нейронные сети

Вычисления в большинстве типовых рекуррентных сетей можно разложить на три блока параметров и соответствующих им преобразований: входа в скрытое состояние; преобразование от предыдущего скрытого состояния в следующее скрытое состояние; преобразование скрытого состояния.

Обычная рекуррентная сеть описывается следующими соотношениями: $h^{(t)} = f\left(Ux^{(t)} + Wh^{(t-1)} + b\right), \quad \tilde{y}^{(t)} = g\left(Vh^{(t)} + c\right)$



Глубокие рекуррентные нейронные сети

В зависимости от того, насколько сложными (глубокими) являются преобразования, выделяется несколько типов рекуррентных сетей

1. Рекуррентная сеть с глубоким преобразованием входного сигнала в скрытый (Deep Transition RNN, DT-RNN)

$$h^{(t)} = f\left(Ux^{(t)} + Wh^{(t-1)} + b\right)$$
$$= \varphi_L\left(U_L^T\varphi_{L-1}\left(U_{L-1}^T\varphi_{L-2}\left(\dots\varphi_1\left(Ux^{(t)} + Wh^{(t-1)} + b\right)\right)\right)\right)$$

где L–количество слоев сети между входным и скрытым слоями.

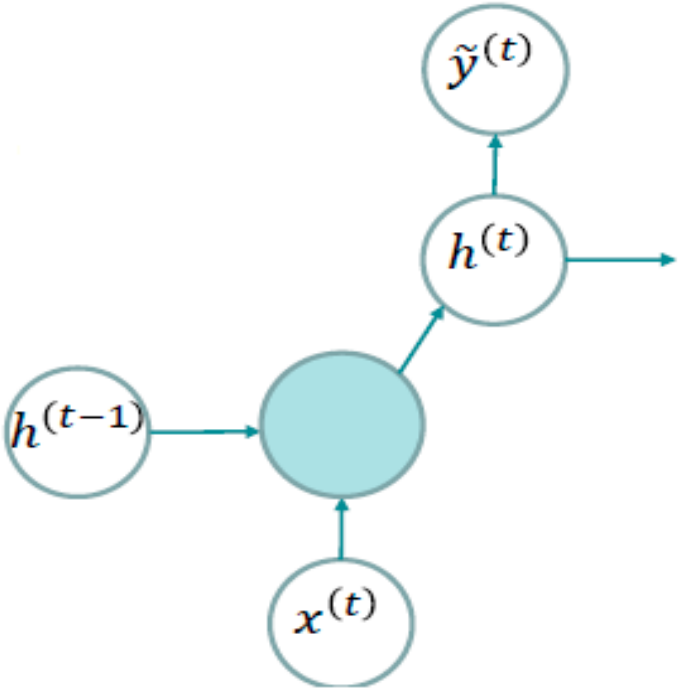
2. Рекуррентная сеть с глубоким преобразованием скрытого сигнала в выходной (Deep Output RNN, DO-RNN)

$$\tilde{y}^{(t)} = g\left(Vh^{(t)} + c\right) = \psi_L\left(V_L^T\psi_{L-1}\left(V_{L-1}^T\psi_{L-2}\left(\dots\psi_1\left(Vh^{(t)} + c\right)\right)\right)\right)$$

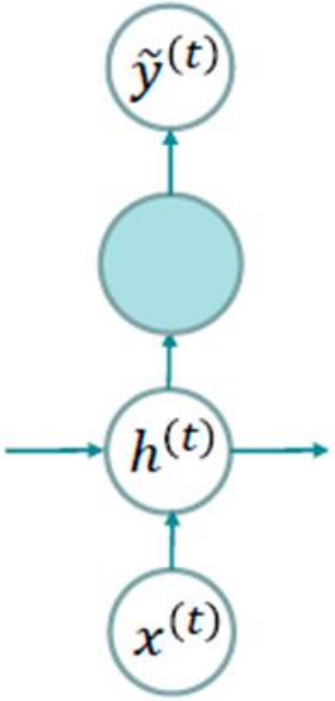
где L–количество слоев сети между скрытым и выходным слоями.

Глубокие рекуррентные нейронные сети

DT-RNN



DO-RNN



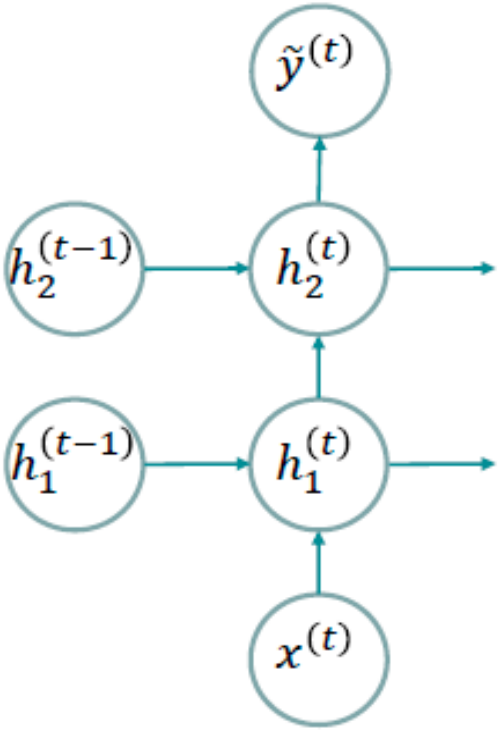
3. Стек из обычных рекуррентных сетей

$$h_l^{(t)} = f_l\left(W_l^T h_l^{(t-1)} + U_l^T h_{l-1}^{(t)}\right)$$

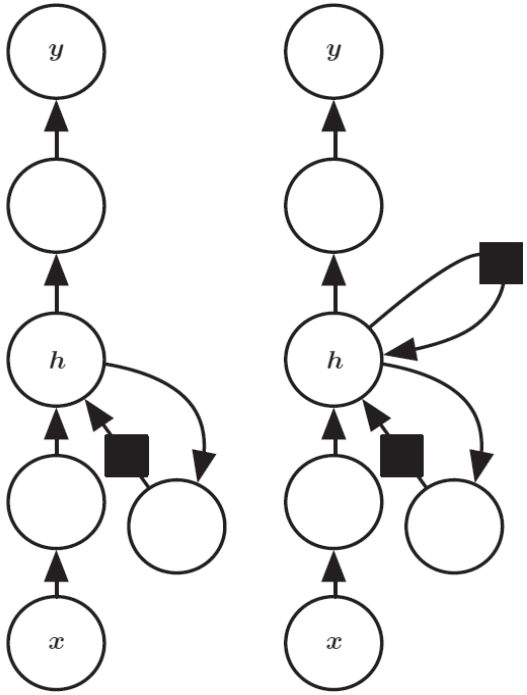
где $h_l^{(t)}$ – скрытое состояние системы на слое с номером l в момент времени t. Эффект удлинения пути можно сгладить путем добавления прямых связей.

Глубокие рекуррентные нейронные сети

Стек RNN



RNN с прямыми связями



Двунаправленные РНС

Все рассмотренные до сих пор рекуррентные сети имели «каузальную» структуру, т. е. на состояние в момент t влияет только информация в прошлые моменты времени $\boldsymbol{x}(1), \ldots, \boldsymbol{x}(t-1)$ и текущий вход $\boldsymbol{x}(t)$. Некоторые модели допускают также влияние прошлых значений $\boldsymbol{y}$ на текущее состояние, если значения $\boldsymbol{y}$ доступны.

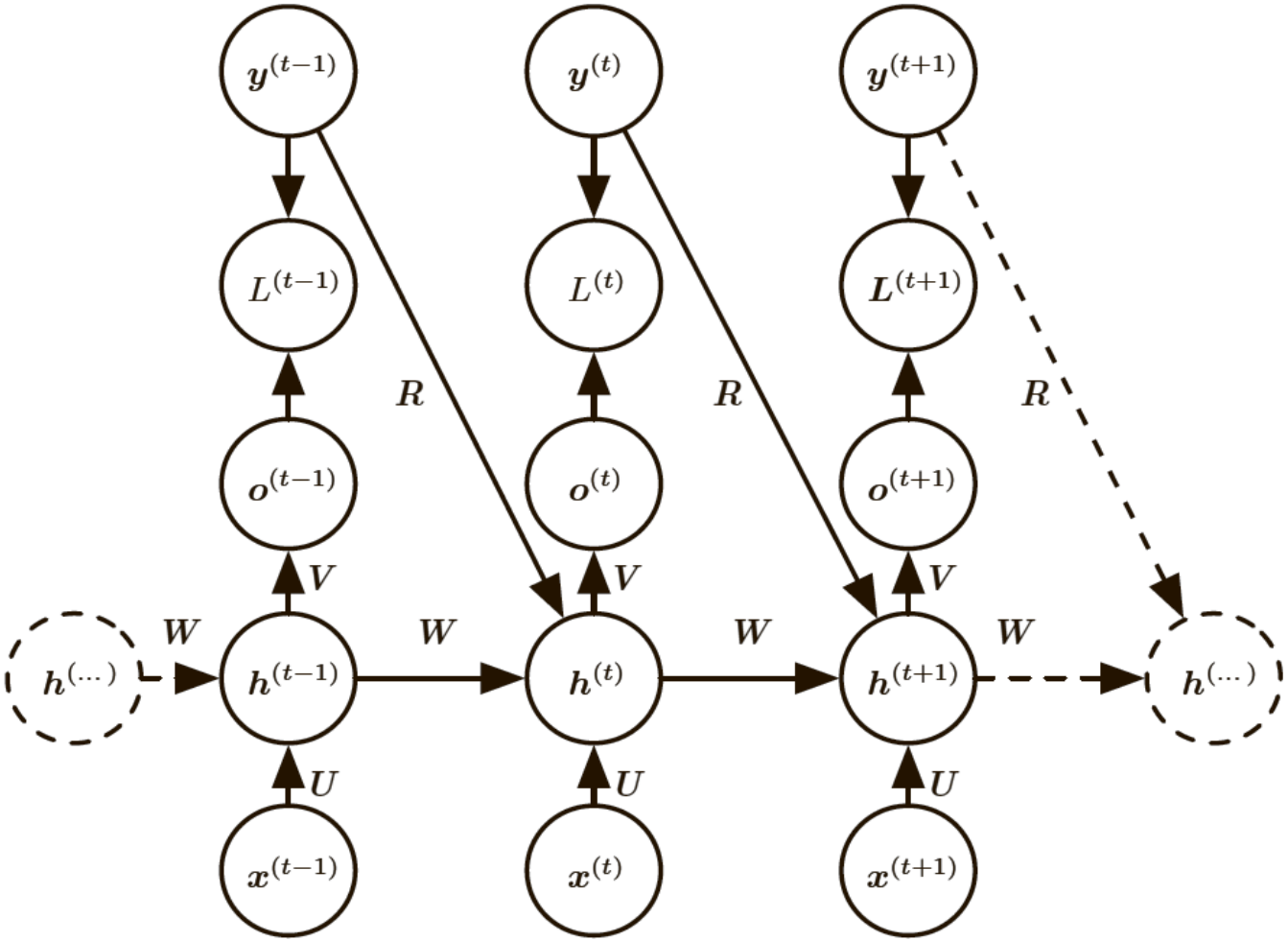
Но во многих приложениях необходимо получать предсказание $\boldsymbol{y}(t)$, которое может зависеть от всей входной последовательности. Например, в задаче распознавания речи правильная интерпретация текущего звука как фонемы может зависеть от нескольких следующих фонем из-за коартикуляции и даже от нескольких следующих слов из-за лингвистических зависимостей между соседними словами: если акустически допустимы две интерпретации текущего слова, то, чтобы различить их, возможно, понадобится заглянуть далеко в будущее (или в прошлое).

Двунаправленные РНС

Двунаправленные рекуррентные нейронные сети были придуманы для удовлетворения именно этой потребности. Они оказались чрезвычайно успешными в распознавании рукописных текстов, речи и биоинформатике.

Как следует из названия, двунаправленные РНС являются комбинацией РНС, движущейся вперед во времени (от начала последовательности к ее концу), и РНС, движущейся в обратном направлении. На рисунке показана типичная двунаправленная РНС; $\boldsymbol{h}(t)$ обозначает состояние той РНС, что движется вперед, а $\boldsymbol{g}(t)$ – той, что движется назад. Это позволяет выходным блокам $\boldsymbol{o}(t)$ вычислять представление, которое зависит *как от прошлого, так и от будущего*, но наиболее чувствительно к входным значениям вблизи момента t; при этом задавать окно фиксированного размера вокруг t необязательно (в отличие от сетей прямого распространения, сверточных сетей и обычных РНС с буфером предвыборки фиксированного размера).

Двунаправленные РНС



Двунаправленные РНС

Формально на момент времени t сеть можно задать следующим образом

$$\begin{aligned} s_t &= \sigma\left(b + W s_{t-1} + U x_t\right), & s'_t &= \sigma\left(b' + W' s'_{t+1} + U' x_t\right), \\ o_t &= c + V s_t + V' s'_t, & y_t &= h\left(o_t\right). \end{aligned}$$

Рассмотренные идеи естественно обобщаются на двумерные входные данные, например изображения, для чего нужны четыре РНС, по одной в каждом направлении: вверх, вниз, влево, вправо. Тогда в каждой точке (i, j) двумерной сети выходной блок O_{ij} мог бы вычислять представление, которое улавливает, в основном, локальную информацию, но может зависеть и от удаленных входов.

По сравнению со сверточной сетью, применение РНС к изображениям обходится дороже, зато может учитывать дальние боковые взаимодействия между элементами одной карты признаков.

РНС. Блоки с утечками

Один из способов включения долгосрочных зависимостей – спроектировать модель, работающую в нескольких временных масштабах, так что одни части модели работают в мелком масштабе и могут обрабатывать мелкие детали, а другие, работающие в крупном масштабе, эффективно передают информацию из отдаленного прошлого в настоящее. Существуют различные стратегии построения мелких и крупных масштабов: добавление прямых связей сквозь время; «блоки с утечками», которые интегрируют сигналы с разными временными постоянными; удаление некоторых связей, используемых для моделирования мелких масштабов.

РНС. Блоки с утечками
Добавление прямых связей сквозь время

Для получения грубого временного масштаба можно добавить прямые связи между переменными в отдаленном прошлом и переменными в настоящем. Идея таких прямых связей восходит к работе Lin et al. (1996) и вытекает из идеи включения задержек в нейронные сети прямого распространения. В обыкновенной рекуррентной сети рекуррентная связь идет из блока в момент t к блоку в момент t + 1. Возможно построить рекуррентные сети с большими задержками.

В рекуррентных сетях, как правило, в результате применения одной и той же операции на каждом шаге длинной временной последовательности создается очень глубокий граф вычислений. При этом градиенты могут исчезать или экспоненциально расти при увеличении числа шагов. Для смягчения этой проблемы введены рекуррентные соединения с временной задержкой d. Теперь градиент экспоненциально убывает как функция τ/d, а не как функция τ.

РНС. Блоки с утечками

Поскольку имеются как связи с задержкой, так и одношаговые связи, градиенты все же могут экспоненциально расти относительно τ. Это позволяет алгоритму обучения улавливать долгосрочные зависимости, хотя и не все такие зависимости хорошо представляются подобным образом.

Блоки с утечкой и спектр разных временных масштабов

Еще один способ получить пути, на которых произведение производных близко к 1, – включать блоки с линейными соединениями с самими собой (самосоединениями), имеющими веса, близкие к единице.

В формуле вычисления скользящего среднего $\mu^{(t)}$ некоторой величины $v^{(t)}$: $\mu^{(t)} \leftarrow \alpha\mu^{(t-1)} + (1-\alpha)v^{(t)}$ параметр α является примером линейного самосоединения $\mu^{(t-1)}$ с $\mu^{(t)}$. Если α близко к 1, то скользящее среднее помнит информацию о прошлом на протяжении долгого времени, а если α близко к 0, то информация о прошлом быстро забывается.

РНС. Блоки с утечками

Скрытые блоки с линейными самосоединениями могут вести себя аналогично скользящему среднему и называются блоками с утечкой, или протекающими.

Прямые связи с пропуском d временных шагов гарантируют, что блок сможет обучиться влиянию значений, находящихся от него на d шагов в прошлом. Использование линейных самосоединений с весом, близким к 1, – другой способ обеспечить блоку доступ к прошлым значениям. И этот подход допускает более плавную и гибкую настройку путем изменения вещественного параметра α, а не целочисленной длины пропуска.

Существуют две основные стратегии задания временных констант для настройки блоков с утечкой. Одна – вручную зафиксировать их значения, например выбрав их из некоторого распределения один раз на этапе инициализации. Вторая – сделать константы свободными параметрами и обучить их.

РНС. Блоки с утечками
Удаление связей

Еще один подход к обработке долгосрочных зависимостей – организация состояния РНС в нескольких временных масштабах с целью упростить протекание информации на большое расстояние в более медленном масштабе.

Эта идея отличается от прямых связей сквозь время, поскольку подразумевается активное удаление связей длины 1 и замена их более длинными. Модифицированные таким способом блоки вынуждены работать в более протяженном временном масштабе. Прямые связи добавляют ребра. Блоки, получившие новые связи, могут обучиться работе в протяженном временном масштабе, но могут поступить и наоборот, предпочтя другие, более короткие связи.

РНС. Блоки с утечками
Удаление связей

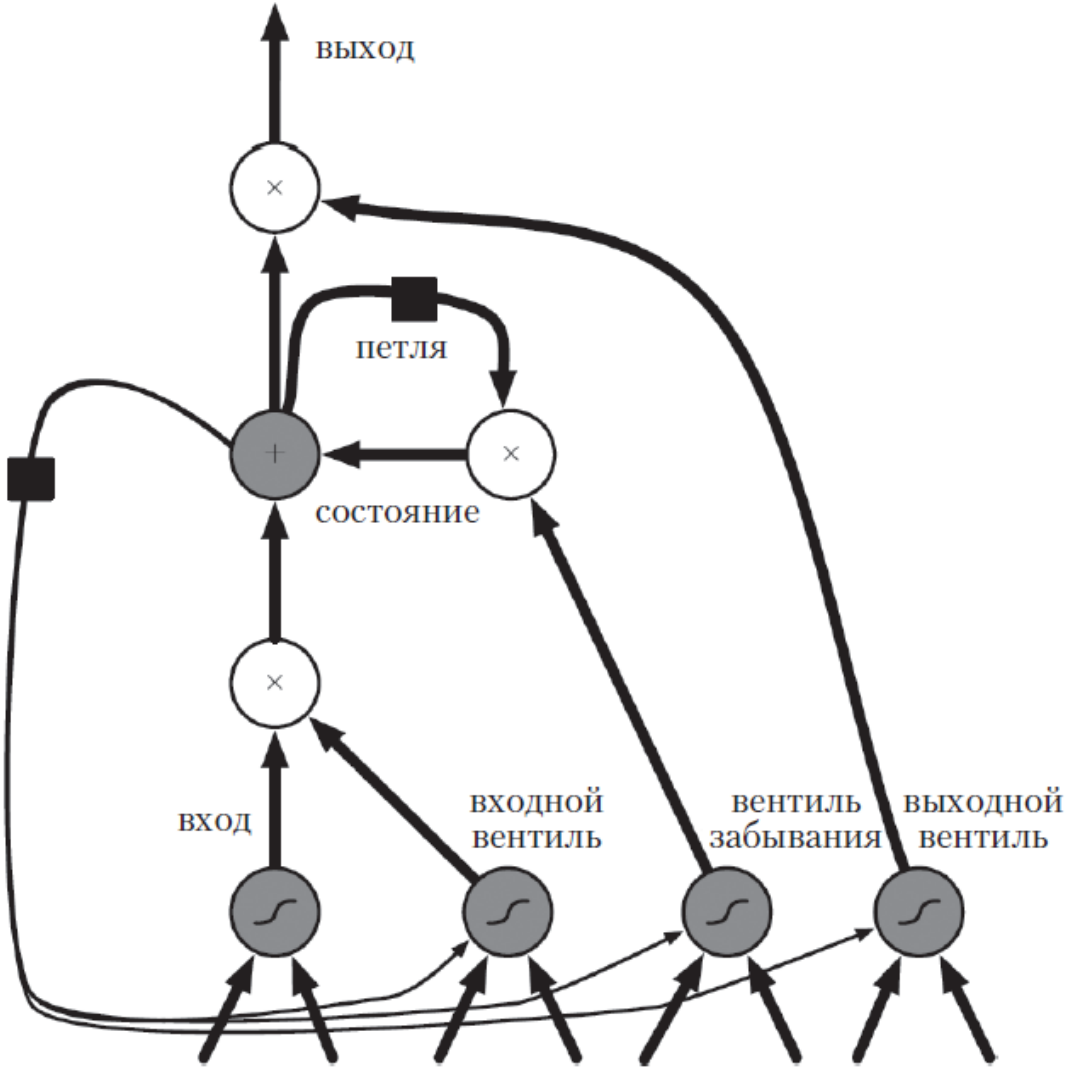
Есть несколько способов принудить группу рекуррентных блоков к работе в разных временных масштабах. Один из них – сделать рекуррентные блоки протекающими, но ассоциировать разные группы блоков с разными фиксированными временными масштабами. Другой способ – производить явные дискретные обновления в разные моменты времени с разной частотой для разных групп блоков. Этот подход показал хорошие результаты на ряде эталонных наборов данных.

Вентильные РНС
Долгая краткосрочная память (LSTM)

Удачная мысль о введении петель для создания путей, по которым градиент может течь длительное время, – основной вклад в первоначальную модель долгой краткосрочной памяти. Позднее было внесено важнейшее дополнение – вес петли должен быть контекстно-обусловленным, а не фиксированным. Сделав вес петли вентильным (управляемым другим скрытым блоком), можно динамически изменять временной масштаб интегрирования. В данном случае имеется в виду, что даже для LSTM с фиксированными параметрами временной масштаб интегрирования может изменяться в зависимости от входной последовательности, поскольку временные константы выводятся самой моделью. Идея LSTM оказалась чрезвычайно успешной во многих приложениях, например: распознавание рукописных текстов, распознавание речи, порождение рукописных текстов, машинный перевод, подписывание изображений и грамматический разбор.

Вентильные РНС. Долгая краткосрочная память

Принципиальная схема LSTM показана на рисунке.



Вентильные РНС. Долгая краткосрочная память

Согласно принципиальной схеме LSTM, ячейки, рекуррентно связанные между собой, заменяют стандартные скрытые блоки в обыкновенных рекуррентных сетях. Входной признак вычисляется регулярным блоком искусственного нейрона. Его значение можно аккумулировать в состоянии, если сигмоидный входной вентиль это допускает. В блоке состояния имеется линейная петля, вес которой управляется вентилем забывания. Выход ячейки можно перекрыть с помощью выходного вентиля. Во всех вентильных блоках имеется сигмоидная нелинейность, тогда как во входном блоке разрешены произвольные сплющивающие нелинейности. Черным квадратом обозначена задержка на одном временном шаге

Вентильные РНС. Долгая краткосрочная память

Есть примеры успешного использования как представленной мелкой, так и более глубоких архитектур.

Рассмотрим схему более подробно. Вместо блока, который просто применяет поэлементную нелинейность к аффинному преобразованию входов и рекуррентным блокам, в рекуррентных LSTM-сетях имеются «LSTM-ячейки», обладающие внутренней рекуррентностью (петлей) в дополнение к внешней рекуррентности РНС.

У каждой ячейки такие же входы и выходы, как у обыкновенной рекуррентной сети, но еще имеются дополнительные параметры и система вентильных блоков, управляющих потоком информации. Самым важным компонентом является блок состояния $s_i^{(t)}$ с линейной петлей, аналогичный описанным выше блокам с утечкой.

Вентильные РНС. Долгая краткосрочная память

Однако теперь вес петли (или ассоциированная временная константа) управляется вентильным блоком забывания $f_i^{(t)}$ (для временного шага t и ячейки i), который присваивает этому весу значение от 0 до 1 с помощью сигмоиды:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)} \right)$$

где – $\boldsymbol{x}^{(t)}$ текущий входной вектор, $\boldsymbol{h}^{(t)}$ – вектор текущего скрытого слоя, содержащий выходы всех LSTM-ячеек, а $\boldsymbol{b}^f, \boldsymbol{U}^f, \boldsymbol{W}^f$ – соответственно смещения, веса входов и рекуррентные веса для вентилей забывания. Таким образом, внутреннее состояние LSTM-ячейки обновляется по следующей формуле, в которой присутствует условный вес петли $f_i^{(t)}$:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)} \right)$$

где $\boldsymbol{b}$, $\boldsymbol{U}$ и $\boldsymbol{W}$ – соответственно смещения, веса входов и рекуррентные веса LSTM-ячейки.

Вентильные РНС. Долгая краткосрочная память

Блок внешнего входного вентиля $g_i^{(t)}$ вычисляется аналогично вентилю забывания (с использованием сигмоиды для получения значения от 0 до 1), но со своими параметрами:

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)} \right)$$

Выход $h_i^{(t)}$ LSTM-ячейки можно перекрыть с помощью выходного вентиля $q_i^{(t)}$ в котором также используется сигмоида:

$$h_i^{(t)} = \tanh(s_i^{(t)}) q_i^{(t)},$$

$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)} \right),$$

и ***b****o*, ***U****o*, ***W****o* – смещения, веса входов и рекуррентные веса соответственно. Один из вариантов – использовать состояние ячейки $s_i^{(t)}$ как дополнительный вход (со своим весом) всех трех вентилей *i*-го блока, как показано на рисунке. Тогда потребуются три дополнительных параметра.

Вентильные РНС

РНС с вентильными рекуррентными блоками (GRU)

В недавнее время разработаны вентильные РНС, блоки которых известны также под названием вентильных рекуррентных блоков (gated recurrent unit – GRU). Основное отличие от LSTM заключается в том, что один вентильный блок одновременно управляет и коэффициентом забывания, и решением об обновлении блока состояния. Уравнения обновления имеют вид:

$$h_i^{(t)} = u_i^{(t-1)} h_i^{(t-1)} + (1 - u_i^{(t-1)}) \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t-1)} + \sum_j W_{i,j} r_j^{(t-1)} h_j^{(t-1)} \right)$$

где ***u*** обозначает вентиль «обновления», а ***r*** – вентиль «сброса». Их значения определяются как обычно:

$$u_i^{(t)} = \sigma\left(b_i^u + \sum_j U_{i,j}^u x_j^{(t)} + \sum_j W_{i,j}^u h_j^{(t)} \right);$$

$$r_i^{(t)} = \sigma\left(b_i^r + \sum_j U_{i,j}^r x_j^{(t)} + \sum_j W_{i,j}^r h_j^{(t)} \right).$$

Вентильные РНС

РНС с вентильными рекуррентными блоками (GRU)

Вентили обновления и сброса могут «игнорировать» части вектора состояния. Вентили обновления действуют как условные интеграторы с утечкой с линейной функцией по любому измерению, т. е. могут либо скопировать вход (один конец сигмоиды), либо полностью проигнорировать его (противоположный конец), заменив новым «целевым состоянием» (к которому интегратор с утечкой желает сойтись).

Вентили сброса контролируют, какие части состояния использовать для вычисления следующего целевого состояния, и вносят дополнительный нелинейный эффект в соотношение между прошлым и будущим состояниями.

Вентильные РНС

РНС с вентильными рекуррентными блоками (GRU)

На эту тему возможно еще много вариаций. Например, выход вентиля сброса (или вентиля забывания) можно разделить между несколькими скрытыми блоками. Или использовать произведение глобального вентиля (управляющего целой группой блоков, например всем слоем) и локального вентиля (управляющего одним блоком) для комбинирования глобального и локального управлений. Однако в нескольких исследованиях архитектурных вариантов LSTM и GRU не найдено решения, которое было бы очевидно лучше обоих на широком круге задач.

Использованные материалы

В презентации использованы материалы, предоставленные:

Бутырским Евгением Юрьевичем, доктором физико-математических наук, профессором кафедры теории управления СПбГУ;

Гришкиным Валерием Михайловичем, кандидат технических наук, доцентом кафедры компьютерного моделирования и многопроцессорных систем.

А также следующие источники:

курс intuit – https://www.intuit.ru/studies/courses/88/88/info

Круглов В.В., Борисов В. В. Искусственные нейронные сети. Теория и практика. - 2-е изд., стереотип. - М.: Горячая линия-Телеком, 2002. - 382 с.