Многослойные нейронные сети без обратной связи

Многослойные сети основаны на архитектурах, состоящих из слоев элементов типа персептрон.

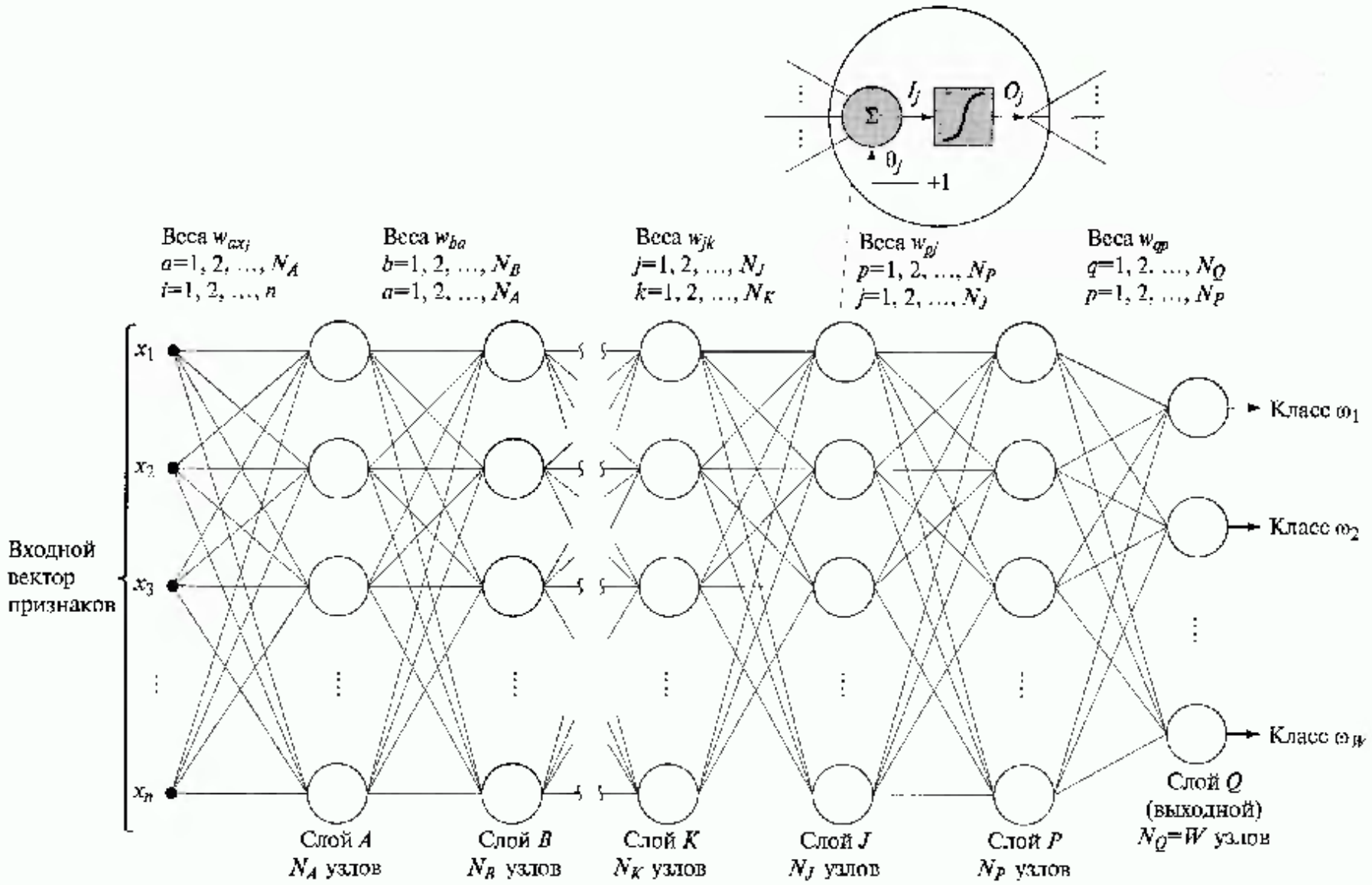
Многослойные сети отличаются тем, что между входными и выходными данными располагаются несколько так называемых скрытых слоев нейронов, добавляющих больше нелинейных связей в модель.

Эти сети строят дискриминантные функции в задачах распознавания образов из нескольких классов. Эти функции не зависят от того разделимы классы или нет.

Среди различных структур нейронных сетей одной из наиболее известных является многослойная структура, в которой каждый нейрон произвольного слоя связан со всеми аксонами нейронов предыдущего слоя или, в случае первого слоя, со всеми входами НС. Такие НС называются полносвязными.

Теоретически число слоев и число нейронов в каждом слое может быть произвольным, однако фактически оно ограничено ресурсами компьютера или специализированной микросхемы, на которых обычно реализуется НС.

Многослойные нейронные сети без обратной связи



Многослойные нейронные сети без обратной связи

Количество нейронов входного слоя **A** соответствует как правило размерности входного вектора признаков $\mathbf{N_A = n}$. Количество нейронов выходного слоя **Q** соответствует количеству распознаваемых классов $\mathbf{N_Q = W}$.

Сеть обучается так, чтобы при подаче вектора признаков x, соответствующего классу ω_i на i-том выходе сети появлялся «высокий» уровень, а на всех остальных выходах – «низкий».

Каждый нейрон имеет тот же вид, что и персептрон за исключением того, что его функция активации непрерывна.

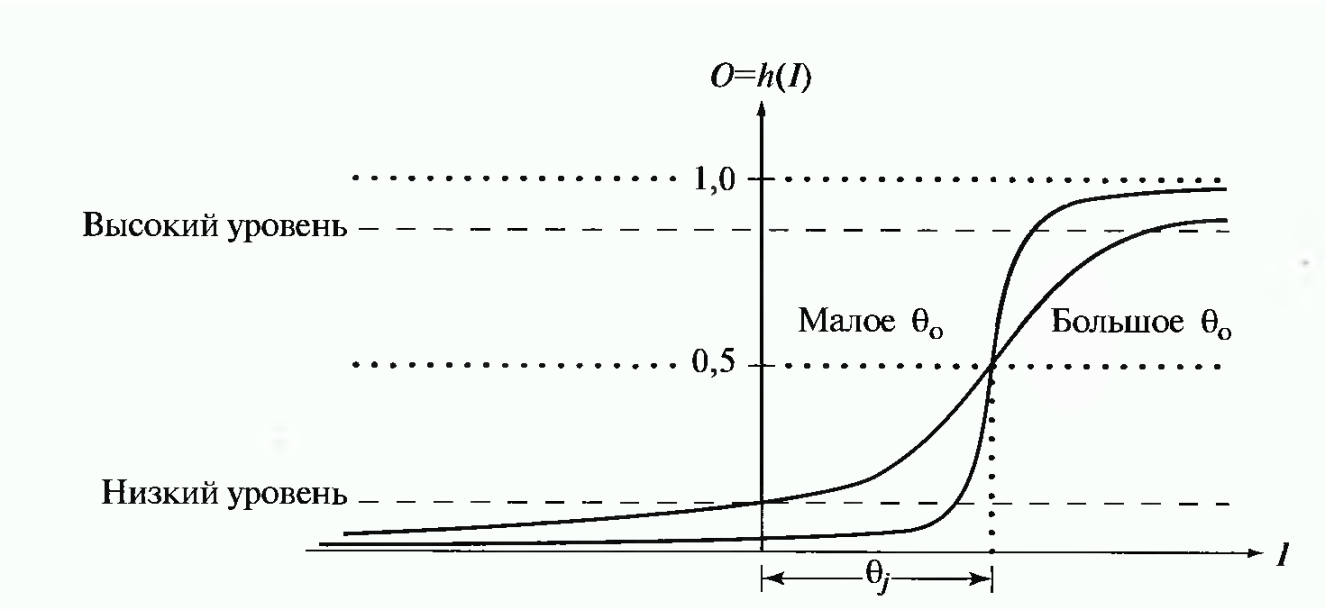
В качестве ее используют непрерывную сигмоидальную функцию со сглаженным порогом. Это обусловлено тем, что для разработки обучающего правила требуется дифференцируемость вдоль всех путей сети.

Многослойные нейронные сети без обратной связи

Поэтому используют следующую функцию активации, которая обладает требуемой дифференцируемостью:

$$h_j(I_j) = \frac{1}{1 + e^{-(I_j + \theta_j)/\theta_0}}, \qquad (1)$$

где $I_j \;\; j = 1,2 \ldots N_j$ - значение на входе активирующего элемента каждого узла слоя **J** нейронной сети, θ_j - смещение, а параметр θ_0 - определяет крутизну сигмоидальной функции.



Многослойные нейронные сети без обратной связи

Смещение θ_j здесь аналогично весовому коэффициенту w_{n+1} персептрона. При этом это смещение рассматривается как дополнительный коэффициент, на который умножается постоянный единичный сигнал, одинаковый для всех узлов сети. Принято не показывать постоянный входной сигнал равный 1, а считать его и его вес θ_j составной частью каждого узла сети.

Входом для любого узла сети является взвешенная сумма сигналов всех элементов предыдущего слоя. Пусть слой **K**, предшествует слою **J** и создает на входе каждого узла **j** слоя **J** сигнал

$$I_j = \sum_{k=1}^{N_K} w_{jk} O_k \qquad (2)$$

для $j = 1,...N_j$, где N_j - число узлов в слое **J**, N_k - число узлов в слое **K**, w_{jk} - веса, модифицирующие сигналы O_k узлов слоя **K** на входе узлов слоя **J.**

Многослойные нейронные сети без обратной связи

Эти выходные сигналы слоя **K** имеют значения

$$O_k = h_k(I_k), \quad k = 1,2,\ldots,N_K .$$

У каждого узла в слое **J** имеется N_K входов, но каждый отдельный вход умножается на свой собственный коэффициент. Так N_K входов первого узла слоя **J** взвешиваются с коэффициентами w_{1k}, входы второго узла этого же слоя взвешиваются с коэффициентами w_{2k} и т.д. Следовательно для преобразования выходных сигналов слоя **K** на входах слоя **J** требуется $N_J \times N_K$ коэффициентов. Чтобы полностью описать узлы слоя **J** надо еще N_J коэффициентов – смещений θ_j

Подстановка выражения (2) в выражение (1) дает функцию активации

$$h_j(I_j) = \frac{1}{1 + e^{-\left(\sum_{k=1}^{N_k} w_{jk} O_k + \theta_j\right)/\theta_0}} . \qquad (3)$$

Многослойные нейронные сети без обратной связи
Проблема обучения

Настройка нейронов последнего слоя в ходе обучения достаточно проста, т.к. желаемые выходные сигналы их известны. Основная проблема – настройка при обучении нейронов **скрытых слоев**.

Есть три варианта решения этой проблемы.

- *Первый вариант* - разработка наборов выходных сигналов, соответствующих входным, для каждого слоя НС, что, является очень трудоемкой операцией и не всегда осуществимо.
- *Второй вариант* - динамическая подстройка весовых коэффициентов синапсов, в ходе которой выбираются, как правило, наиболее слабые связи и изменяются на малую величину в ту или иную сторону, а сохраняются только те изменения, которые повлекли уменьшение ошибки на выходе всей сети. Данный метод требует громоздких рутинных вычислений.
- *Третий вариант* - распространение сигналов ошибки от выходов НС к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы.

Нейронные сети обратного распространения ошибки

Типичная сеть обратного распространения имеет: входной слой; выходной слой; не менее одного скрытого слоя.

Нейроны организованы в послойную структуру с прямой передачей сигнала. Каждый нейрон сети продуцирует взвешенную сумму своих входов, пропускает эту величину через передаточную функцию и выдает выходное значение. Сеть может моделировать функцию практически любой сложности, причем число слоев и число нейронов в каждом слое определяют сложность функции.

Нейронные сети обратного распространения ошибки

Для построения архитектур НС используют следующие правила.

1. Количество входов и выходов сети определяются количеством входных и выходных параметров исследуемого объекта. В отличие от внешних слоев, число нейронов скрытого слоя выбирается эмпирическим путем. В большинстве случаев достаточное количество нейронов составляет $n_{ск} \geq n_{вх} + n_{вых;}$, где $n_{вх}$, $n_{вых}$ - количество нейронов во входном и выходном слоях.

2. Если сложность в отношении между полученными и желаемыми данными на выходе увеличивается, количество нейронов скрытого слоя должна также увеличиться.

3. Если моделируемый процесс может разделяться на много этапов, нужен дополнительный скрытый слой (слои).

Нейронные сети обратного распространения ошибки

После того, как определено число слоев и число нейронов в любом из них, необходимо найти значения для синаптических весов и порогов сети, способных минимизировать погрешность конечного результата.

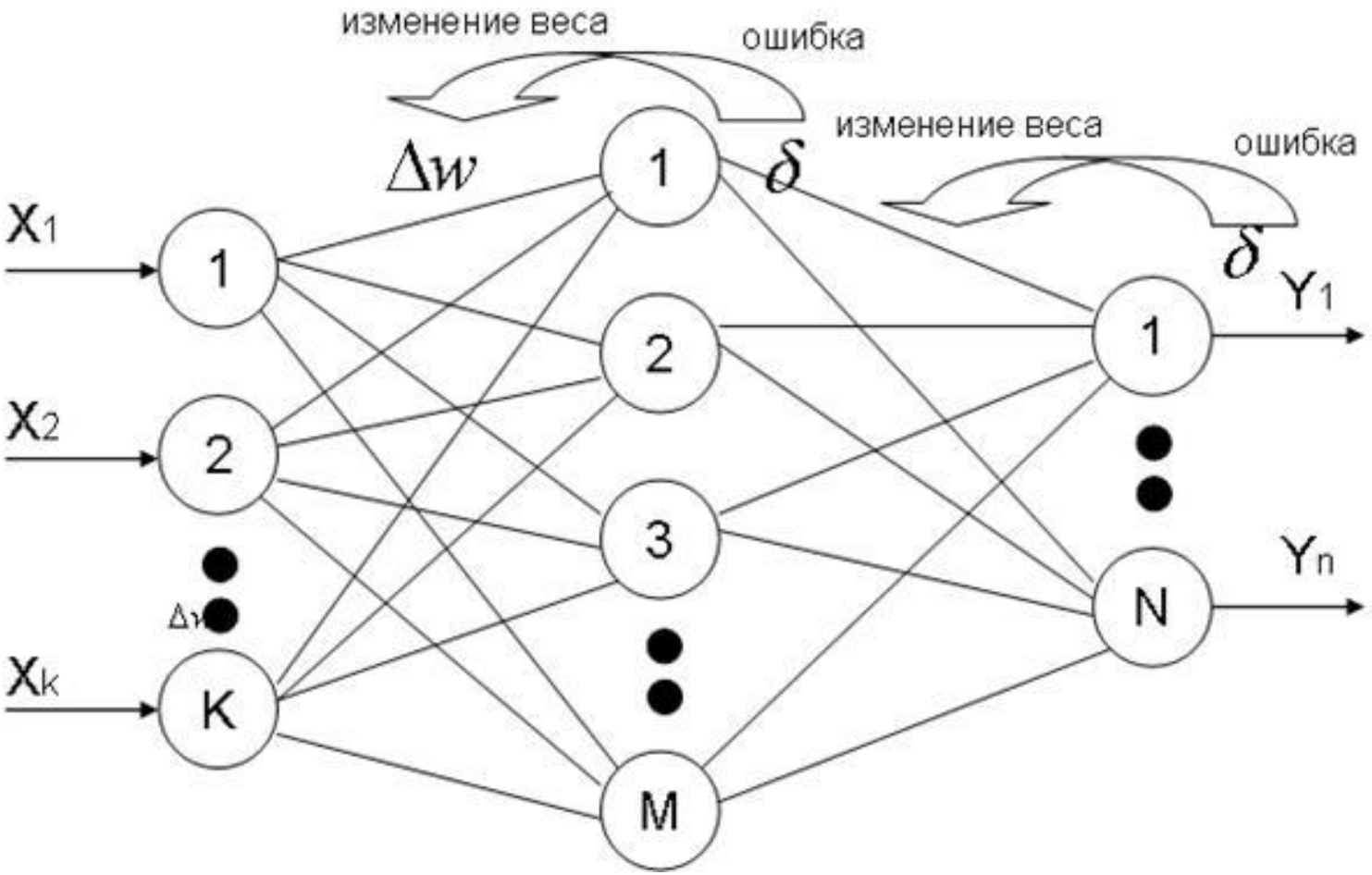
Погрешность для конкретной модели сети определяется путем прохождения через сеть всех обучающих примеров и сравнения выходных значений с желаемыми значениями. Множество погрешностей создает функцию погрешностей, значение которой определяет погрешность сети.

Для разных соединений весов погрешность сети можно изобразить точкой в N+1-измеримом пространстве, все эти точки образовывают поверхность - поверхность состояний.

Обучение нейросети по сути состоит в исследовании этой поверхности

Нейронные сети

Обучение методом обратного распространения ошибки



Нейронные сети

Обучение методом обратного распространения ошибки

Вначале посмотрим выходной слой. Суммарный квадрат ошибки между желаемыми реакциями r_q и выходного слоя **Q** и фактическими его реакциями O_q равен

$$E_Q = \frac{1}{2}\sum_{q=1}^{N_Q}(r_q - O_q)^2,$$

где N_Q - количество нейронов выходного слоя. Цель состоит в разработке правила, аналогичного дельта-правилу, которое позволяло бы при обучении минимизировать эту ошибку.

Обучение методом обратного распространения ошибки

Для этого будем как раньше корректировать веса пропорционально частной производной функции ошибки по этим весам

$$\Delta w_{qp} = -\alpha \frac{\partial E_Q}{\partial w_{qp}}, \qquad (4)$$

где слой P предшествует слою Q, α - положительный коэффициент коррекции.

Функция ошибки E_Q зависит от выходных сигналов O_Q, которые являются функциями входных сигналов I_q

Вычислим производную

$$\frac{\partial E_Q}{\partial w_{qp}} = \frac{\partial E_Q}{\partial I_q} \frac{\partial I_q}{\partial w_{qp}}.$$

С учетом, что $I_j = \sum_{k=1}^{N_K} w_{jk} O_k$ получим

Обучение методом обратного распространения ошибки

$$\frac{\partial I_q}{\partial w_{qp}} = \frac{\partial}{\partial w_{qp}} \sum_{p=1}^{N_P} w_{qp} O_p = O_p .$$

Подставляя в выражение корректировки весов (4) имеем

$$\Delta w_{qp} = -\alpha \frac{\partial E_Q}{\partial I_q} O_p = \alpha \delta_q O_p , \qquad (4.1)$$

где $\delta_q = -\frac{\partial E_Q}{\partial I_q}$.

Найдем $\partial E_Q / \partial I_q$, через формулу производной сложной функции, выражая частную производную через скорость изменения E_Q относительно O_q и скорость изменения O_q относительно I_q

$$\delta_q = -\frac{\partial E_Q}{\partial I_q} = -\frac{\partial E_Q}{\partial O_q} \frac{\partial O_Q}{\partial I_q} . \qquad (5)$$

Обучение методом обратного распространения ошибки

Вспомним ошибку которую минимизируем

$$E_Q = \frac{1}{2}\sum_{q=1}^{N_Q}(r_q - O_q)^2,$$

Тогда $\frac{\partial E_Q}{\partial O_q} = -(r_q - O_q),$ учитывая, что $O_k = h_k(I_k),$

Получаем

$$\frac{\partial O_q}{\partial I_q} = \frac{\partial}{\partial I_q} h_q(I_q) = h'_q(I_q).$$

Подставляя это в (5) получаем выражение для δ_k пропорциональное ошибке $(r_q - O_q)$

$$\delta_q = (r_q - O_q)h'_q(I_q)$$

Обучение методом обратного распространения ошибки

Подставляя эти выражения в (4.1) окончательно получим выражение для корректировки весов

$$\Delta w_{qp} = \alpha(r_q - O_q)h'_q(I_q)O_p = \alpha\delta_q O_p .$$

После того, как задана функция $h_q(I_q)$ все члены в этом выражении становятся известными или могут быть получены из наблюдения за сетью.

Иными словами, предъявляя обучающий образ, мы знаем какой должна быть желаемая реакция каждого выходного узла. Значения на каждом выходном узле мы можем наблюдать также как и его входные сигналы активирующих элементов слоя **Q** а также выходные сигналы предыдущего слоя **P**. Таким образом мы знаем как откорректировать веса последнего слоя.

Двигаясь назад посмотрим, что происходит в предыдущем слое.

Обучение методом обратного распространения ошибки

Действуя также получим для слоя **P**

$$\Delta w_{pj} = \alpha(r_p - O_p)h'_p(I_p)O_j = \alpha\delta_p O_j ,$$

где составляющая ошибки имеет вид

$$\delta_p = (r_p - O_p)h'_p(I_p).$$

Все члены здесь кроме r_p известны, либо могут быть получены из наблюдения. Для r_p мы не можем сформулировать вид значения желаемых реакций. Поэтому надо переформулировать способ получения δ_p на основе известных величин.

Запишем составляющую ошибки слоя **P** в виде

$$\delta_p = -\frac{\partial E_p}{\partial I_p} = -\frac{\partial E_p}{\partial O_p}\frac{\partial O_p}{\partial I_p}.$$

Обучение методом обратного распространения ошибки

Как и ранее

$$\frac{\partial O_p}{\partial I_p} = \frac{\partial h_p(I_p)}{\partial I_p} = h'_p(I_p),$$

При этом $h'_p(I_p)$ вычисляется по функции активации и наблюдаемым входам слоя **P**.

Членом, который порождает r_p является производная $\partial E_p / \partial O_p$, поэтому ее надо выразить так, чтобы она не содержала r_p

$$-\frac{\partial E_p}{\partial O_p} = -\sum_{q=1}^{N_Q} \frac{\partial E_p}{\partial I_q}\frac{\partial I_q}{\partial O_p} = \sum_{q=1}^{N_Q}\left(-\frac{\partial E_p}{\partial I_q}\right)\frac{\partial}{\partial O_p}\sum_{p=1}^{N_P} w_{qp}O_p =$$

$$= \sum_{q=1}^{N_Q}\left(-\frac{\partial E_p}{\partial I_q}\right)w_{qp} = \sum_{q=1}^{N_Q}\delta_q w_{qp},$$

Обучение методом обратного распространения ошибки

Подставив это в выражение для δ_p получим

$$\delta_p = h'_p(I_p)\sum_{q=1}^{N_Q}\delta_q w_{qp}\,.$$

Здесь все известно. В этом выражении важно, что δ_p вычисляется через величины, полученные в слое, следующем за слоем **P**.

После вычисления составляющей ошибки и весов слоя **P,** эти величины можно использовать для предыдущего слоя и т.д. Именно это и называется обратным распространением ошибки.

Итак в целом. Для любого слоя **J** которому предшествует слой K вычисляются веса w_{jk} модифицирующие связи между слоями

Обучение методом обратного распространения ошибки

$$\Delta w_{jk} = \alpha \delta_j O_k .$$

Если слой **J** выходной, то

$$\delta_j = (r_j - O_j) h'_j(I_j).$$

Если слой **J** внутренний, а следующий за ним слой **P** то

$$\delta_j = h'_j(I_j) \sum_{p=1}^{N_P} \delta_p w_{jp}$$

Применяя активирующую функцию с параметром $\theta_0 = 1$ вычисляем $h'_j(I_j) = O_j(1 - O_j),$

Тогда для выходного слоя $\delta_j = (r_j - O_j) O_j (1 - O_j)$

Для внутренних слоев $\delta_j = O_j(1 - O_j) \sum_{p=1}^{N_P} \delta_p w_{jp}$

Обучение методом обратного распространения ошибки

Все веса сети инициализируются произвольно, но не одинаково.

Предъявляется вектор признаков и сигнал распространяется по сети до выхода.

Затем реакции выходных узлов O_q сравниваются с желаемыми r_q и строятся составляющие ошибок δ_q

На втором этапе осуществляется обратный проход по сети, при котором соответствующие сигналы ошибок передаются в каждый узел, что позволяет скорректировать его веса. Эта процедура применяется и к смещениям.

Обычно отслеживают все ошибки в сети, в том числе и связанные с конкретными образами.

При успешном сеансе обучения величина ошибки уменьшается и процедура обучения сходится к устойчивому набору весов.

Обучение методом обратного распространения ошибки

Многокритериальная задача оптимизации в методе обратного распространения рассматривается как набор однокритериальных задач - на каждой итерации происходят изменения значений параметров сети, которые улучшают работу лишь с одним примером обучающей выборки. Такой подход существенно уменьшает скорость обучения.

Модификации алгоритма обратного распространения связаны с использованием разных функций погрешности, разных процедур определения направления и величины шага.

В целом, работа всех сетей сводится к обобщению входных сигналов, принадлежащих n-мерному гиперпространству, по некоторому числу классов. С математической точки зрения это происходит путем разбиения гиперпространства гипер-плоскостями

$$\sum_{i=1}^{n} x_i \cdot w_{ik} = T_k , \quad k=1...m$$

Алгоритм обучения с учителем

1. Проинициализировать элементы весовой матрицы.
2. Подать на входы один из входных векторов, которые сеть должна научиться различать, и вычислить ее выход.
3. Если выход правильный, перейти на шаг 4.

Иначе вычислить разницу между идеальным и полученным значениями выхода: $\delta = Y_I - Y$

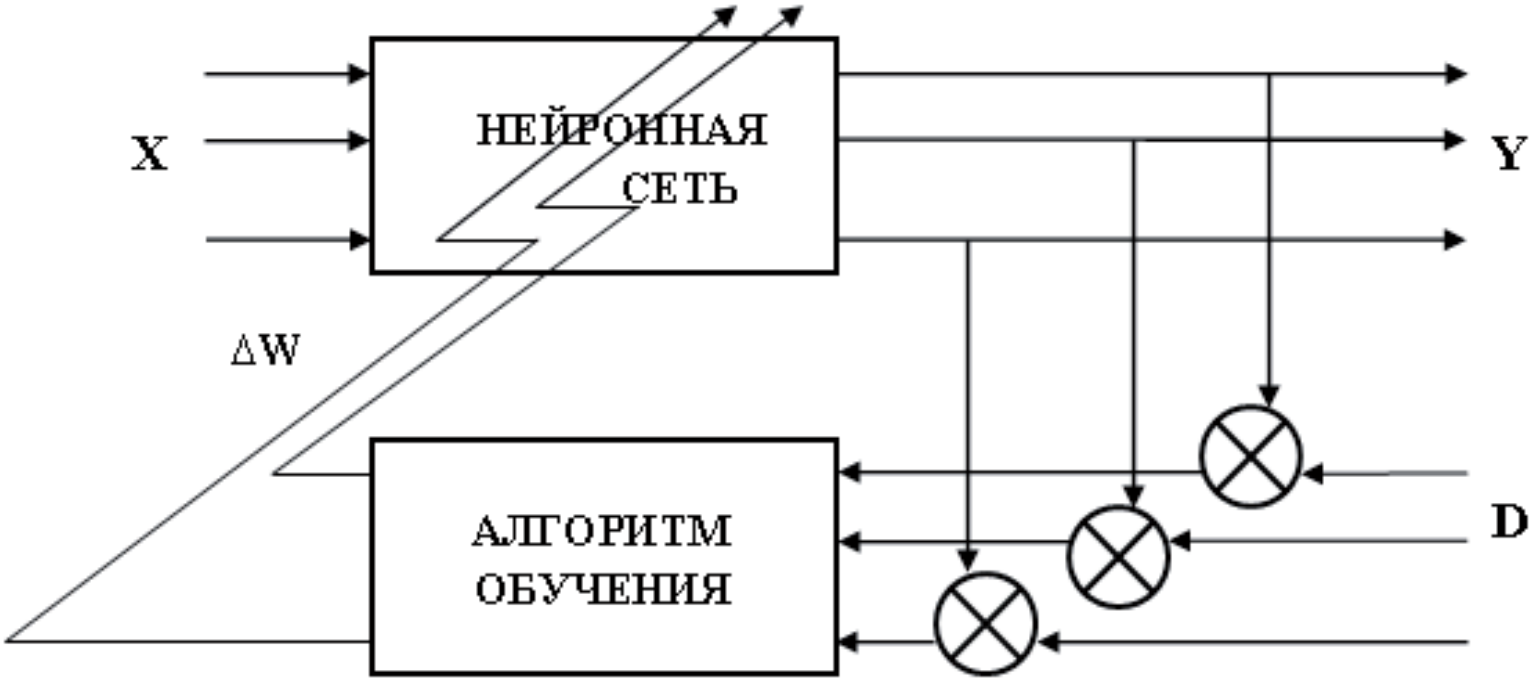
Модифицировать веса в соответствии с формулой:

$$w_{ij}(t+1) = w_{ij}(t) + \nu \cdot \delta \cdot x_i$$

где t и t+1 - номера текущей и следующей итераций; v - коэффициент скорости обучения; i - номер входа; j - номер нейрона в слое.

4. Цикл с шага 2, пока сеть не перестанет ошибаться.

Алгоритм обучения с учителем



Способы ускорения сходимости

1. Выбор начальных весов.
2. Упорядочение данных.
3. Импульс:

$$\Delta W_{ij}(t+1) = \mu * \Delta W_{ij}(t) - (1-\mu)\varepsilon \frac{\partial E}{\partial W_{ij}}.$$

где μ - число в интервале (0,1), которое задается пользователем.

4. Управление величиной шага.
5. Оптимизация архитектуры сети.
6. Масштабирование данных.

Методы обучения без учителя. Метод Хебба

Сигнальный метод обучения Хебба заключается в изменении весов по следующему правилу:

$$w_{ij}(t) = w_{ij}(t-1) + \alpha \cdot y_i^{(n-1)} \cdot y_j^{(n)}$$

где $y_i^{(n-1)}$ зыходное значение нейрона i слоя (n-1), $y_j^{(n)}$ выходное значение нейрона j слоя n; $w_{ij}(t)$ и $w_{ij}(t-1)$ весовой коэффициент синапса, соединяющего эти нейроны на итерациях t и t-1 соответственно; α - коэффициент скорости обучения. Здесь и далее, для общности, под n подразумевается произвольный слой сети.

При обучении по данному методу усиливаются связи между возбужденными нейронами.

Методы обучения без учителя. Дифференциальный метод Хебба

Существует также и дифференциальный метод обучения Хебба.

$$w_{ij}(t) = w_{ij}(t-1) + \alpha \cdot \left[y_i^{(n-1)}(t) - y_i^{(n-1)}(t-1) \right] \cdot \left[y_j^{(n)}(t) - y_j^{(n)}(t-1) \right]$$

Здесь $y_i^{(n-1)}(t)$ и $y_j^{(n)}(t-1)$ - выходное значение нейрона i слоя n-1 соответственно на итерациях t и t-1; $y_j^{(n)}(t)$ и $y_j^{(n)}(t-1)$ - то же самое для нейрона j слоя n. Как видно из формулы, сильнее всего обучаются синапсы, соединяющие те нейроны, выходы которых наиболее динамично изменились в сторону увеличения.

Методы обучения без учителя. Алгоритм обучения Хебба

Полный алгоритм обучения с применением вышеприведенных формул будет выглядеть так:

1. На стадии инициализации всем весовым коэффициентам присваиваются небольшие случайные значения.
2. На входы сети подается входной образ, и сигналы возбуждения распространяются по всем слоям согласно принципам классических прямопоточны х сетей, в результате чего полу-чается выходное значение $y_i^{(n)}$, $i = \overline{0\ M_i - 1}$, $n = \overline{0\ N-1}$, где M - число нейронов в слое i; , а N – число слоев в сети.
3. На основании полученных выходных значений нейронов производится изменение весовых коэффициентов.
4. Цикл с шага 2, пока выходные значения сети не стабилизируются с заданной точностью.

Методы обучения без учителя. Алгоритм Кохонена

Алгоритм Кохонена - предусматривает подстройку синапсов на основании их значений от предыдущей итерации.

$$w_{ij}(t) = w_{ij}(t-1) + \alpha \cdot \left[y_i^{(n-1)} - w_{ij}(t-1) \right]$$

Обучение сводится к минимизации разницы между входными сигналами нейрона. поступающими с выходов нейронов предыдущего слоя $y_i^{(n-1)}$ и весовыми коэффициентами его синапсов.

Полный алгоритм обучения имеет примерно такую же структуру, как и алгоритм Хебба, но на шаге 3 из всего слоя выбирается нейрон, значения синапсов которого максимально походят на входной образ, и подстройка весов по формуле проводится только для него.

Методы обучения без учителя. Алгоритм Кохонена

Эта, так называемая, аккредитация может сопровождаться затормаживанием всех остальных нейронов слоя и введением выбранного нейрона в насыщение. Выбор такого нейрона может осуществляться расчетом скалярного произведения вектора коэффициентов с вектором входных значений. Максимальное произведение дает выигравший нейрон.

Другой вариант - расчет расстояния между этими векторами в p- мерном пространстве, где p - размер векторов.

$$D_j = \sqrt{\sum_{i=0}^{p-1} (y_i^{(n-1)} - w_{ij})^2}$$

где j - индекс нейрона в слое n, i - индекс суммирования по нейронам слоя (n-1), w_{ij} - вес синапса, соединяющего нейроны; выходы нейронов слоя (n-1) являются входными значениями для слоя n.

Методы обучения без учителя. Алгоритм Кохонена

В данном случае, «побеждает» нейрон с наименьшим расстоянием. Иногда слишком часто получающие аккредитацию нейроны принудительно исключаются из рассмотрения, чтобы «уравнять права» всех нейронов слоя. Простейший вариант такого алгоритма заключается в торможении только что выигравшего нейрона.

При использовании обучения по алгоритму Кохонена существует практика нормализации входных образов, а так же - на стадии инициализации - и нормализации начальных значений весов. $x_i = x_i / \sqrt{\sum_{j=0}^{n-1} x_j^2}$

где x_i - i-ая компонента вектора входного образа или вектора весовых коэффициентов, а n - его размерность.

Методы обучения без учителя. Алгоритм Кохонена

Нормализация позволяет сократить длительность процесса обучения. Инициализация весовых коэффициентов случайными значениями может привести к тому, что различные классы, которым соответствуют плотно распределенные входные образы, сольются или, наоборот, раздробятся на дополнительные подклассы в случае близких образов одного и того же класса. Для исключения такой ситуации используется метод выпуклой комбинации, который сводится к преобразованию входных нормализованных образов:

$$x_i = \alpha(t) \cdot x_i + (1 - \alpha(t)) \cdot n^{-0.5}$$

где x_i - i-ая компонента входного образа, n - общее число его компонент, $\alpha(t)$ - коэффициент, изменяющийся в процессе обучения от нуля до единицы.

Метод потенциальных функций

Пусть требуется разделить два непересекающихся образа V1 и V2.

В процессе обучения с каждой точкой пространства изображений, соответствующей единичному объекту из обучающей последовательности, связывается функция $U(\mathbf{X}, \mathbf{X}_i)$. Такие функции называются потенциальными.

Обучающей последовательности объектов соответствует последовательность векторов X1, X2, … , в пространстве изображений с которыми связана последовательность U(X, X1) U(X, X2) ПФ, используемых для построения функций f(X1, X2). По мере увеличения числа объектов в процессе обучения функция f должна стремиться к одной из разделяющих функций. В результате обучения могут быть построены ПФ для каждого образа:

$$U_1(\mathbf{X}) = \sum_{X_i \in V_1} U(\mathbf{X}, \mathbf{X}_i), \; U_2(\mathbf{X}) = \sum_{X_i \in V_2} U(\mathbf{X}, \mathbf{X}_i)$$

Метод потенциальных функций

В качестве разделяющей функции f(X) можно выбрать функцию вида:

$$f(\mathbf{X}) = U_1(\mathbf{X}) - U_2(\mathbf{X})$$

которая положительна для объектов одного образа и отрицательна для объектов другого. Рассмотрим потенциальную функцию

$$U(\mathbf{X}, \mathbf{X}_i) = \sum_{j=1}^{\infty} \lambda_j^2 \varphi_j(\mathbf{X}) \varphi_j(\mathbf{X}_i) = \sum_{j=1}^{\infty} \psi_j(\mathbf{X}) \psi_j(\mathbf{X}_i)$$

где $\varphi_j(\mathbf{X})$ - линейно независимая система функций;

λ_j - действительные числа, отличные от нуля для всех i=1, 2 ;

$\mathbf{X}_i$ - точка, соответствующая i-му объекту из обучающей последовательности.

В процессе обучения предъявляется обучающая последовательность и на каждом n-м такте обучения строится приближение $f_n(X)$, которое характеризуется следующей основной рекуррентной процедурой:

$$f_{n+1}(\mathbf{X}) = q_n f_n(\mathbf{X}) + r_n U(\mathbf{X}_{n+1}, \mathbf{X})$$

Метод потенциальных функций

Значения q_n и r_n, являются фиксированными функциями номера n. Как правило $q_n \equiv 1$, а r_n, выбирается в виде:

$$r_n \equiv \gamma_n \left(S\left(f_n\left(\mathbf{X}_{n+1}\right), f\left(\mathbf{X}_{n+1}\right)\right)\right)$$

где $S(f_n, f)$ - невозрастающие функции, причем $S(f, f) \equiv 0$

$$S(f_n, f)\begin{cases} \le 0, & f_n \ge 0 \\ \ge 0, & f_n \le 0 \end{cases}$$

Коэффициенты γ_n представляют собой неотрицательную числовую последовательность, зависящую только от номера n. Кроме того,

$$\sum_{n=1}^{\infty} \gamma_n = \infty \text{ и } \sum_{n=1}^{\infty} \gamma_n^2 < \infty$$

например, $\gamma_n = 1/n$ или $\gamma_n = const$

Алгоритмы потенциальных функций

1. Полагается $f_0(\mathbf{X})\equiv 0$ (нулевое приближение). Пусть в результате применения алгоритма после n-го шага построена разделяющая функция $f_n(\mathbf{X})$ а на (n+1)-м шаге предъявлено изображение Xn+1 для которого известно действительное значение разделяющей функции f(Xn+1). Тогда функция fn+1(X) строится по следующему правилу:

$$f_{n+1}(\mathbf{X}) = f_n(\mathbf{X}) + \gamma_{n+1} sign(f(\mathbf{X}_{n+1}) - f_n(\mathbf{X}_{n+1})) \cdot U(\mathbf{X},\mathbf{X}_{n+1})$$

2. Принимается что $f_0(\mathbf{X})\equiv 0$. Переход к следующему приближению от функции $f_n(\mathbf{X})$ к $f_{n+1}(\mathbf{X})$ осуществляется в результате следующей рекуррентной процедуры:

$$f_{n+1}(\mathbf{X}) = f_n(\mathbf{X}) + (f(\mathbf{X}_{n+1}) - f_n(\mathbf{X}_{n+1})) \cdot \frac{1}{\lambda} U(\mathbf{X},\mathbf{X}_{n+1}).$$

где λ - произвольная положительная константа, удовлетворяющая условию $\lambda = (1/2) \cdot max(\mathbf{X}, \mathbf{X}_j)$.

Возможно принять $\psi_j(\mathbf{X}) = sign(\sum_{v=1}^{m} \beta_{vj} x_v + \Theta_j)$

и предположить, что xv может иметь только два значения 0 и 1.

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:

https://github.com/vdumoulin/conv_arithmetic

https://habr.com/ru/company/oleg-bunin/blog/340184/

http://bioinformaticsinstitute.ru/sites/default/files/vvedenie_v_deep_learning.pdf