ЛЕКЦИЯ 2

Персептроны

Нейронные сети (НС).
Введение в НС.

Виды активационных функций F

Название	функция	Область значений
линейная	$f(s)= ks$	$(-\infty,+\infty)$
полулинейная	$f(s)= ks , s > 0$ $f(s)= 0 , s <= 0$	$(0,\infty)$
логистическая (сигмоидальная)	$f(s)= 1/(1+e^{-as})$	(0,1)
гиперболический тангенс (сигмоидальная)	$f(s)= (e^{as}-e^{-as}) / (e^{as}+e^{-as})$	(-1,1)
Экспоненциальная	$f(s) = e^{-as}$	$(0,\infty)$
Синусоидальная	$f(s) = \sin(s)$	(-1,1)
Сигмоидальная (рациональная)	$f(s) = s / (a+\|s\|)$	(-1,1)
Шаговая (линейная с насыщением)	$f(s) = -1, s <= -1$ $f(s) = 0, -1 < s < 1$ $f(s) = 1, s >= 1$	(-1,1)
Пороговая	$f(s) = 0, s < 0$ $f(s) = 1, s >= 0$	(0,1)

Виды активационных функций

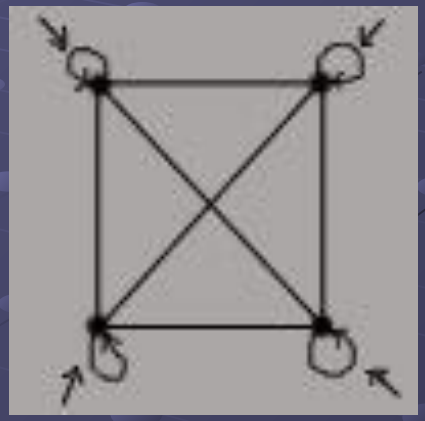
Название	функция	Область значений
Модульная	$f(s) = \|s\|$	$(0,\infty)$
Знаковая (сигнатурная)	$f(s) = sgn(s), s!=0$ $f(s) = -1, s = 0$	(-1,1)
Квадратичная	$f(s) = s^2$	(-1,1)
Радиально-базисная	$f(s) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(X-\overline{X})}{2\sigma^2}}$	$(-\infty,\infty)$

Классификация нейронных сетей

№1 Классификация по топологии

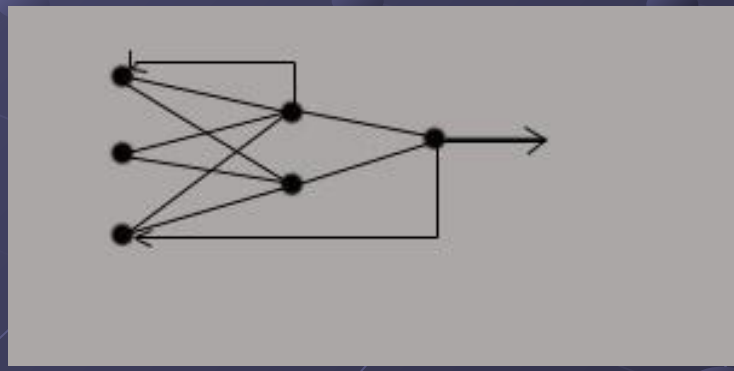
1.1 Полносвязные сети:

Каждый нейрон передает свой выходной сигнал всем остальным нейронам в сети, включая самого себя.



1.2 Многослойные или слоистые:

Сеть разбивается на слои, где каждый слой содержит совокупность нейронов с едиными входными сигналами. Слои нумеруются слева на право, начиная с



Многослойные сети делятся на:

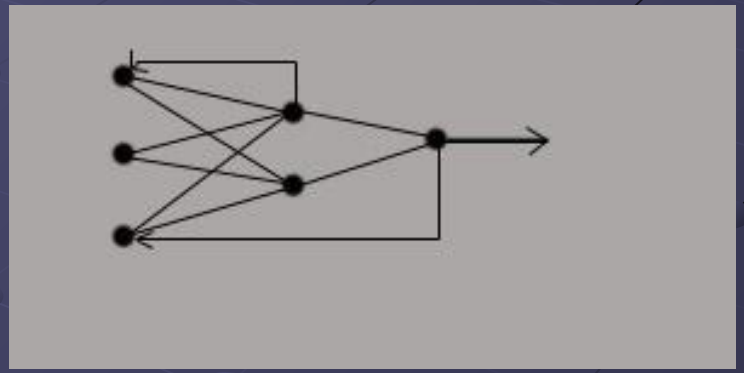
1.2.1 Монотонные

.

1.2.2 Сети без обратных связей:

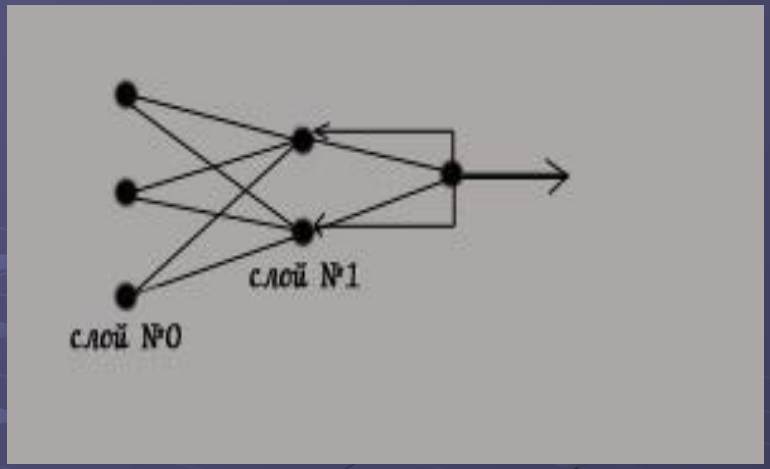
Обычная многослойная сеть, с 1-им ограничением, что нейроны q-ого слоя могут передавать сигнал нейронам (q+1)-ого слоя и никакому другому слою.

1.2.3 Сети с обратными связями:



1.2.3.1 Слоисто циклические:

Многослойная сеть у которой выход передает свои сигналы 1-му слою



1.2.3.2 Слоисто-полносвязные:

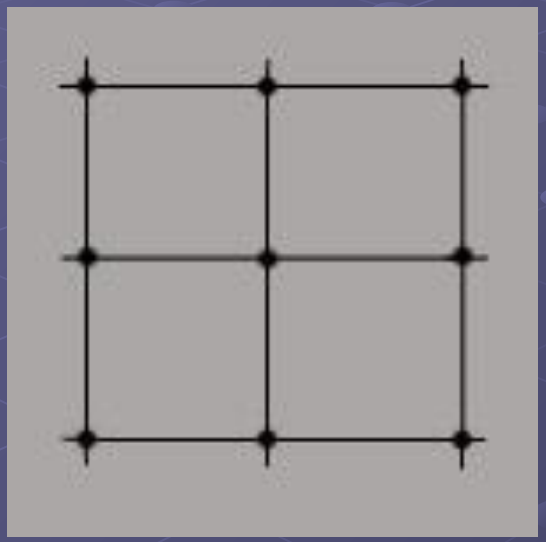
Состоит из слоев, каждый из которых представляет собой полносвязную сеть.

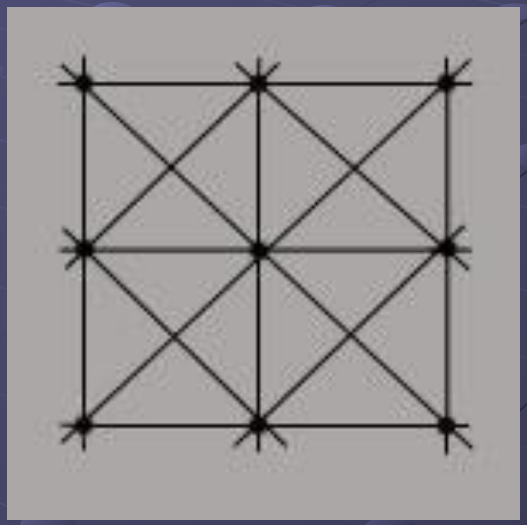
1.2.3.3 Полносвязные-слоистые:

По своей структуре такие же как и 1.2.3.2, а функционально, в них не разделяют фазы обмена внутри слоя и передачи следующему, на каждом такте нейроны всех слоев применяют сигналы от нейронов как своего слоя, так и последующих.

1.3 Слабосвязанные сети:

В слабосвязанных сетях нейроны располагаются в узлах прямоугольной форму или гексагональной решетки.
Каждый нейрон связан с 4-мя (окрестность фон Неймана), 6-ью (окрестность Голея) или 8-мью (окрестность Мура) своими ближайшими соседями.





№2 Классификация по типам структур:

2.1 Гомогенные:
Нейроны одного типа
с единой функцией активации.

2.2 Гетерогенные

Нейроны с различными функциями активации.

№3 Классификация по типу сигнала:

Бинарные - двоичный сигнал

Аналоговые.

№4 Классификация по типу работы

4.1 Синхронные
В момент времени t, лишь один

нейрон меняет свое состояние.

4.2 Асинхронные
Группа нейронов
меняет свое состояние.
как правило группа,
значит весь слой.

Алгоритмы обучения НС делятся на

обучение “с учителем"

"без учителя"

Схема обучения НС с учителем

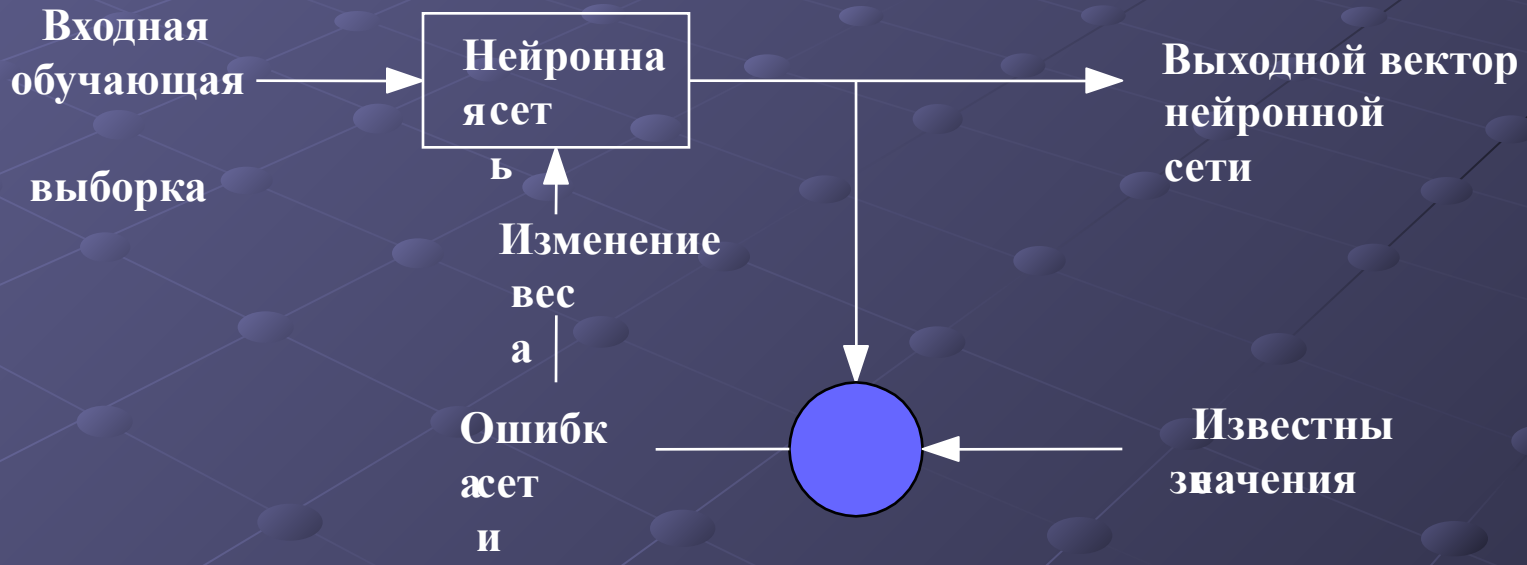


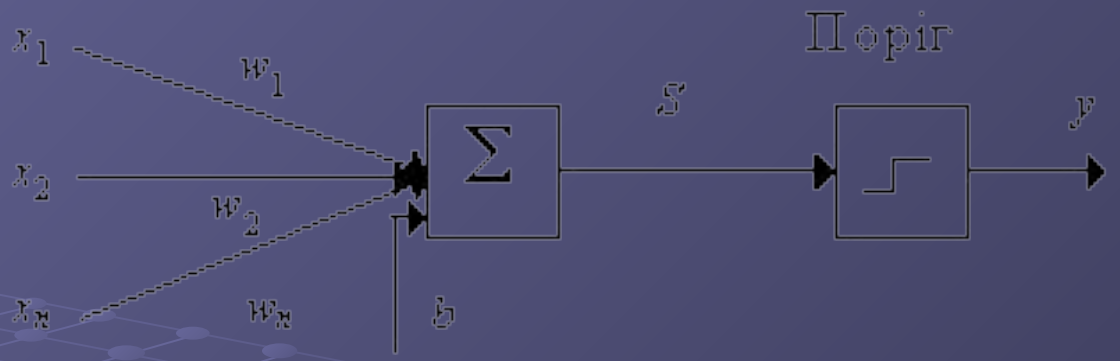
Схема обучения НС без учителя

АЛГОРИТМЫ ОБУЧЕНИЯ НС

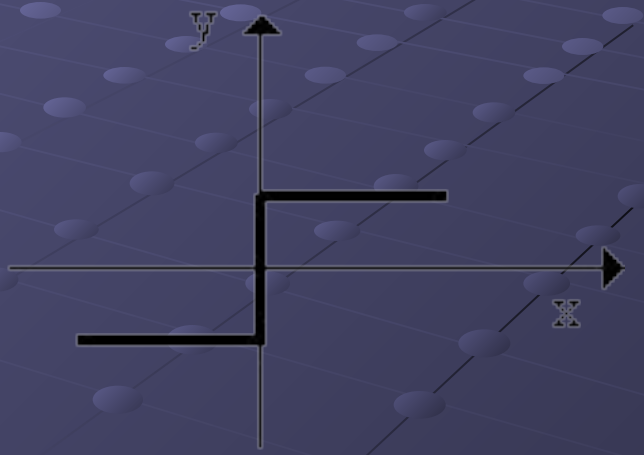
Пара-дигма	Правило обучения	Архитектура	Алгоритм обучения	Решаемая задача
С учите-лем	Коррекция ошибки	Однослойный и многослойный перцептрон	Алгоритмы обучения перцептрона Обратное распространение Adaline и Madaline	Классификация Распознавание образов Аппроксимация функций Прогнозирование Управление
	Больцмана	Рекуррентная	Алгоритм обучения Больцмана	Классификация Распознавание образов
	Хебба	Многослойная прямого распространения	Линейный дискриминантный анализ	Анализ данных Классификация Распознавание образов
	Соревнование	Соревнование	Векторное квантование	Категоризация в середине класса Сжатие данных
		Сеть ART	ARTMap	Классификация Распознавание образов

Парадигма	Правило обучения	Архитектура	Алгоритм обучения	Решаемая задача
Без учителя	Коррекция ошибки	Многослойная прямого распространения	Проекция Саммона	Категоризация в середине класса Анализ данных
	Хебба	прямого распространения или соревнование	Анализ главных компонентов	Анализ данных Сжатие данных
		Сеть Хопфилда	Обучение ассоциативной памяти	Ассоциативная память
	Соревнование	Соревнование	Векторное квантование	Категоризация Сжатие данных
		SOM Кохонена	SOM Кохонена	Категоризация Анализ данных
		Сеть ART	ART1, ART2	Категоризация
Смешанная	Коррекция ошибки и соревнование	Сеть RBF	Алгоритм обучения RBF	Классификация Распознавание образов Аппроксимация функций Прогнозирование

ПЕРСЕПТРОН
Розенблатта



$$y_j = \begin{cases} 1, & \sum_i W_{ij} x_i > \theta_j \\ 0, & \sum_i W_{ij} x_i \le \theta_j \end{cases}$$



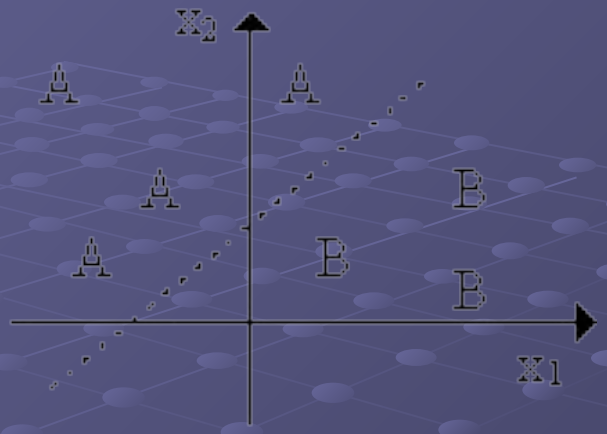
При заданных значениях весов и порогов, нейрон имеет определенное значение выходной активности для каждого возможного вектора входов. Множество входных векторов, при которых нейрон активен (y=1), отделено от множества векторов, на которых нейрон пассивен (y=0) *гиперплоскостью*, уравнение которой :

$$\sum_i W_{ij} x_i - \theta_j = 0$$

Метод обучения состоит в итерационной подстройке матрицы весов, последовательно уменьшающей ошибку в выходных векторах. Алгоритм включает несколько шагов:

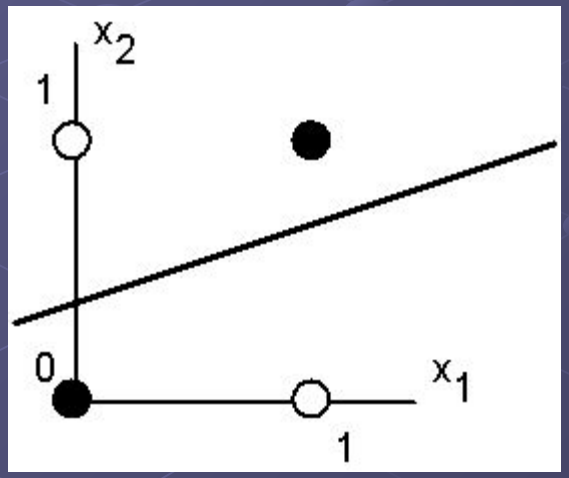
Шаг 0.	Начальные значения весов всех нейронов $W(t=0)$ полагаются случайными.
Шаг 1.	Сети предъявляется входной образ x^{α}, в результате формируется выходной образ $\mathbf{y}^{\alpha} \neq y^{\alpha}$
Шаг 2.	Вычисляется вектор ошибки $\delta^{\alpha} = \left(y^{\alpha} - \mathbf{y}^{\alpha}\right)$ сети на выходе. Дальнейшая идея состоит в том, что изменение вектора весовых коэффициентов в области малых ошибок должно быть пропорционально ошибке на выходе, и равно нулю если ошибка равна нулю.
Шаг 3.	Вектор весов модифицируется по следующей формуле: $W(t+\Delta t) = W(t) + \eta x^{\alpha} \cdot \left(\delta^{\alpha}\right)^{T}$. Здесь $0 < \eta < 1$ - темп обучения.
Шаг 4.	Шаги 1-3 повторяются для всех обучающих векторов. Один цикл последовательного предъявления всей выборки называется *эпохой*. Обучение завершается по истечении нескольких эпох, а) когда итерации сойдутся, т.е. вектор весов перестает изменяться, или б) когда полная просуммированная по всем векторам абсолютная ошибка станет меньше некоторого малого значения.

Классификация на основе ПЕРСЕПТРОНА Розенблатта



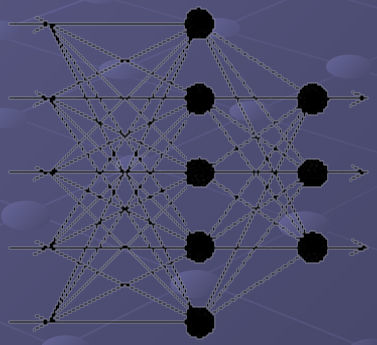
Ситуация линейной неразделимости множеств белых и черных точек

Логическая функция “исключающее или”



X_1	X_2	Y
0	0	0
1	0	1
0	1	1
1	1	0

Архитектура MLP FeedForward BackPropagation была разработана в начале 1970-х лет несколькими независимыми авторами: Вербор (Werbor); Паркер (Parker); Румельгарт (Rumelhart), Хинтон (Hinton) и Вильямс (Williams). Сейчас, парадигма BackPropagation наиболее популярная, эффективная и легкая модель обучения для сложных, многослойных сетей. Типичная сеть MLP BackPropagation имеет входной слой, выходной слой и по крайней мере один скрытый слой. Теоретически, ограничений относительно числа скрытых слоев не существует, но практически применяют один или два.



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. Определение числа промежуточных слоев и числа нейронов в них является важным при моделировании сети.

Большинство исследователей и инженеров, применяя архитектуру к определенным проблемам используют общие правила, в частности:

Количество входов и выходов сети определяются количеством входных и выходных параметров исследуемого объекта, явления, процесса, и т.п.. В отличие от внешних слоев, число нейронов скрытого слоя выбирается эмпирическим путем. В большинстве случаев достаточное количество нейронов составляет

$$n_{ск} < n_{вх} + n_{вых},$$

где $n_{вх}$, $n_{вых}$ - количество нейронов во входном и, соответственно, в выходном слоях.

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

Если моделируемый процесс может разделяться на много этапов, нужен дополнительный скрытый слой (слои). Если процесс не разделяется на этапы, тогда дополнительные слои могут допустить перезапоминание и, соответственно, неверное общее решение.

В процессе функционирования нейронная сеть формирует выходной сигнал Y в соответствии с входным сигналом X, реализуя некоторую функцию Y = G(X). Если архитектура сети задана, то вид функции G определяется значениями синаптических весов и смещений сети.

Обучение состоит в поиске (синтезе) функции G, близкой к оптимальной в смысле минимальной ошибки E. Если выбрано множество обучающих примеров – пар

(X^k, Y^k) (где k = 1,2,…,N) и способ вычисления функции ошибки E, то обучение нейронной сети превращается в задачу многомерной оптимизации, имеющую очень большую размерность. При этом, поскольку функция E может иметь произвольный вид, обучение в общем случае представляет собой многоэкстремальную невыпуклую задачу оптимизации.

В качестве ошибки сети принимается некоторая функция ошибки на выходном элементе (элементах) сети.

Квадратичная(Sum-squared) ошибка полагается равной сумме квадратов разностей между целевыми t_i и фактическими o_i выходными значениями каждого *k*-го выходного элемента по всем *i = 1..n* образцам на *l*-й эпохе. При обучении сетей такая функция ошибок является стандартной.

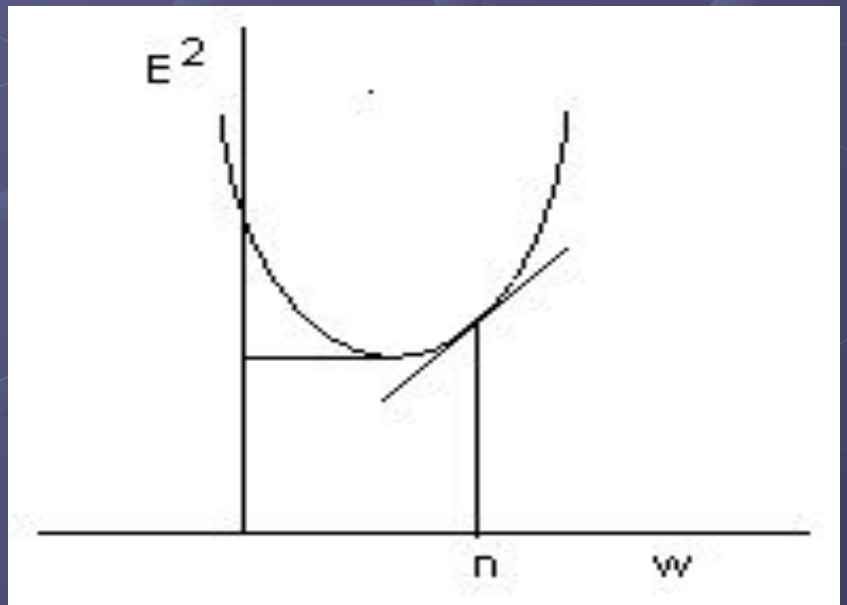
$$SSE_l = \sum_{k=1}^{m} \sum_{i=1}^{n} (o_{ki} - t_{ki})^2$$

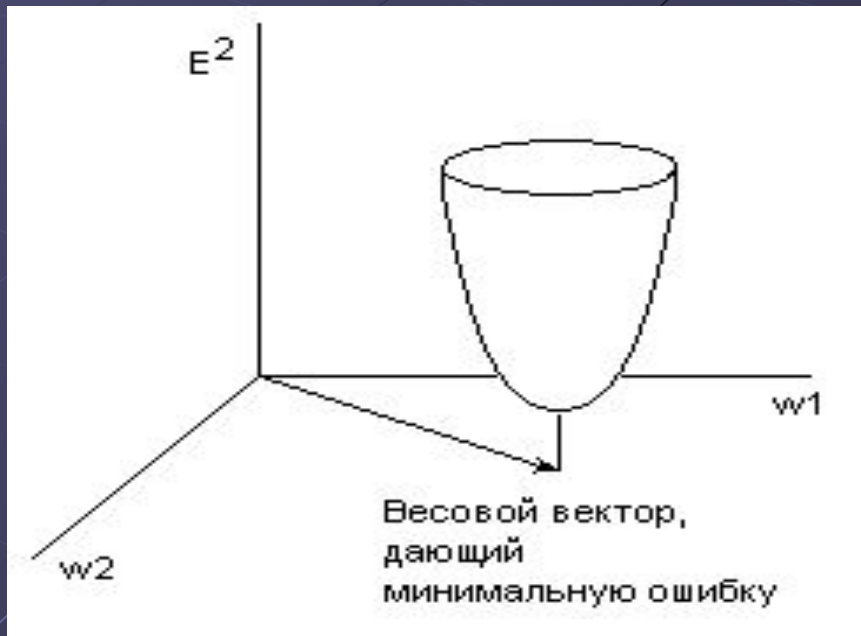
После того как задали архитектуру сети, необходимо найти параметры, при которых ошибка минимальна. В линейных моделях можно определить параметры, дающие *абсолютный* минимум ошибки. С нелинейными моделями дело обстоит гораздо сложнее. Настраивая сеть, нельзя быть уверенным, что алгоритм обучения достиг *глобального* минимума, иными словами, утверждать, что нельзя добиться лучшего результата.

Поэтому для контроля обучения сети полезна *поверхность ошибок*.

Каждому из весов и порогов сети (их общее число обозначим через N) соответствует одно измерение в многомерном пространстве. (N+1)-мерное измерение соответствует ошибке сети. Для данного набора весов соответствующую ошибку сети можно изобразить точкой в (N+1)-мерном пространстве. В итоге все такие точки образуют некоторую поверхность – поверхность ошибок. Цель обучения нейронной сети в том, чтобы найти самую низкую точку этой поверхности. В случае линейной модели с суммой квадратов в качестве функции ошибок, поверхность ошибок представляет собой параболоид. В общем случае она может иметь локальные минимумы (точки, самые низкие в некоторой своей окрестности, но лежащие выше глобального минимума), седловые точки и т.д. Отталкиваясь от некоторой начальной конфигурации весов и порогов, алгоритм производит поиск глобального минимума. Для этого вычисляется градиент в данной точке и используется для продвижения вниз по склону на поверхности ошибок. В конце концов, алгоритм приводит к некоторой точке, которая, однако, может оказаться лишь точкой локального минимума. Поэтому следует поэкспериментировать с несколькими начальными конфигурациями.

Перед началом обучения весовые коэффициенты устанавливаются равными некоторым случайным значениям. При этом точка, представляющая начальное состояние сети, может оказаться в любом месте на поверхности ошибок, но очень маловероятно, чтобы она оказалась в точке минимума этой поверхности. В процессе обучения сеть должна корректировать весовые коэффициенты так, чтобы максимально уменьшить значения общей ошибки. То есть, весовые коэффициенты должны корректироваться в том направлении, в котором спуск вниз по поверхности ошибок происходит быстрее всего. На рисунке эта идея иллюстрируется для одного веса, где n обозначает время или итерацию. Прямая линия представляет производную ошибки в зависимости от веса в момент времени n. Для случая двух весовых коэффициентов получается чашевидная поверхность ошибок, подобная показанной на рисунке.





распространения ошибок

Шаг 1. Весам сети присваиваются небольшие начальные значения, например значениями из диапазона между -0.3 и +0.3, полученные с помощью генератора случайных чисел, распределённых по нормальному, равномерному или иному закону.

Шаг 2. Выбирается очередная обучающая пара (X,Y) из обучающего множества; вектор X подаётся на вход сети. Обучение предполагается управляемым, поскольку с каждым входным образцом X связывается целевой выходной образец Y.

Шаг 3. Вычисляется результат на каждом j-м выходе сети.

Шаг 4. Вычисляется разность между требуемым t_j (целевым, Y) и реальным o_j (вычисленным) выходом сети, т.е. каждый из выходов сети вычитается из соответствующего компонента целевого вектора (выходного образца) с целью получения ошибки. После этого вычисляется локальный градиент ошибки для каждого элемента сети, кроме входных. Для выходных элементов используется формула (4)

$$\delta_j = (t_j - o_j)\, o_j\, (1 - o_j)$$

а для элементов внутренних слоёв – формула (5)

$$\delta_k = o_j\, (1 - o_j) \sum_{k+1} \delta_{k+1} w_{k+1} j$$

Шаг 5. Веса сети корректируются так, чтобы минимизировать ошибку. Сначала веса выходного слоя, затем, с использованием правила дифференцирования сложной функции и своеобразного вида производной сигмоидальной функции (или другой функции активации), веса предыдущего слоя и т.п. Вычисление приращения веса выполняется по формуле (6)

$$\Delta w_{ij}(n+1) = \eta \delta_j o_j$$

Чтобы уменьшить вероятность того, что изменения весов приобретут осциллирующий характер, вводится инерционный член α (momentum), добавляемый в пропорции, соответствующей предыдущему изменению веса. Так что значение весового коэффициента связи между *i*-м и *j*-м нейронами на *(n+1)*-м шаге тренировки составит

$$\Delta w_{ij}(n+1) = \eta \delta_j o_i + \alpha \Delta w_{ij}(n)$$

Таким образом, изменение веса на шаге *(n+1)* оказывается зависящим от изменения веса на шаге *n*. (формула 7)

$$w_{ij}(n+1) = \Delta w_{ij}(n+1) + w_{ij}(n)$$

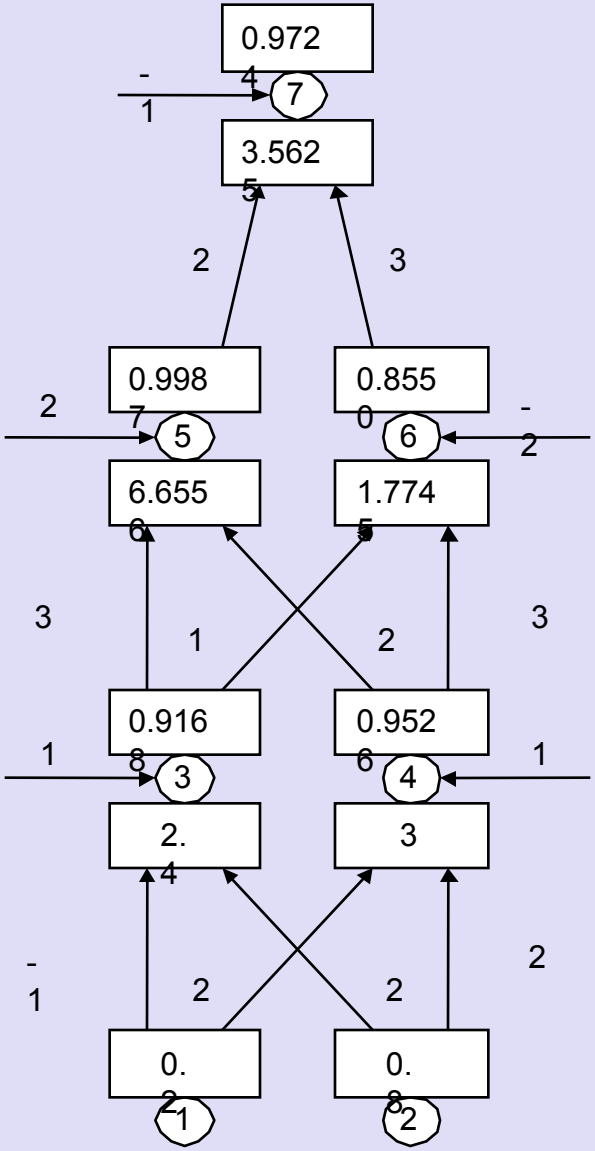
Шаг 6. Шаги со 2-го по 5-й повторяются для каждой пары тренировочного множества до тех пор, пока не выполнится условие останова. Например, обучение продолжается до тех пор, пока изменение усреднённой квадратичной ошибки (RMS) не окажется меньше некоторого допустимого значения при переходе от одной эпохи к следующей (или не выполнится другое условие останова). При этом допустимое значение 0.01 будет означать, что усреднённая квадратичная ошибка соседних эпох не должна отличаться более чем на ±0.01. Если в процессе обучения наступает момент, когда ошибка в сети попадает в рамки допустимого изменения, говорят, что наблюдается *сходимость*. Другим критерием окончания обучения можно считать наступление момента, когда выход для каждого учебного образца оказывается в рамках допустимого отклонения от соответствующего целевого выходного образца.

Вычисления в сети выполняются последовательно послойно.

Шаги 2 и 3 можно рассматривать как «проход вперёд», так как сигнал распространяется по сети от входа к выходу. Шаги 4 и 5 составляют «обратный проход», поскольку здесь вычисляемый сигнал ошибки распространяется обратно по сети и используется для подстройки весов.

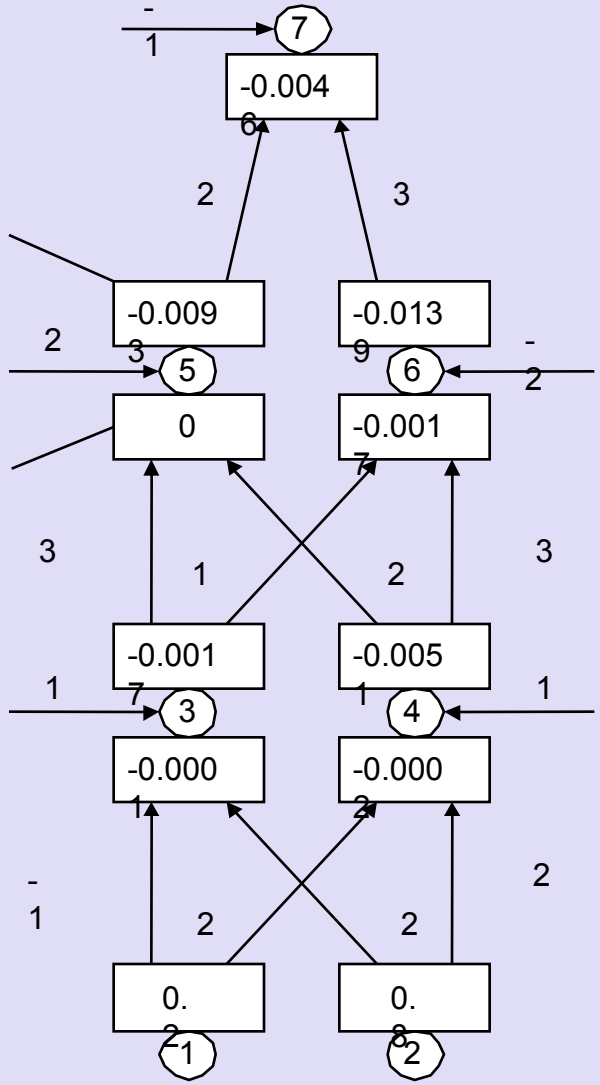
Прямой проход

Целевым является значение 0.8, поэтому ошибка для выходного элемента равна (0.800 - 0.9724) x 0.9724 (1 - 0.9724) = -0.0046



Ошибка предыдущего слоя

Фактическая ошибка



Входные элементы с линейной функцией активации передают входной сигнал на свои выходы без изменений. Они также не имеют порогов.

Совокупный входной сигнал будем считать по формуле .

$$net_j = w_0 + \sum_{i=0} x_i w_{ij}$$

Для третьего элемента w(0,3)=1, w(1,3)=-1, w(2,3)=2,

Net3 = (1*1) + (-1*0.2) + (2*0.8) = 2.4

Выходной сигнал третьего элемента рассчитаем по формуле (2)

O3 = 1/(1 + exp(-2.4))= 0.9168

Выполним подобные вычисления для остальных элементов сети

Net4 = (1*1) + (2 * 0.2) + (2 * 0.8) = 3

O4 = 1/(1+exp(-3)) = 0.9526

Net5 = (2 * 1) + (3 * 0.9168) + (2 * 0.9526) = 6.6556

O5 = 1/(1+exp(-6.6556)) = 0.9987

Net6 = (-2*1) + (1 * 0.9168) + (3 * 0.9526) = 1.7745

O6 = 1/(1+exp(-1.7745)) = 0.8550

Net7 = (-1*1) + (2*0.9987) + (3 * 0.8550) = 3.5625

O7 = 1/(1+exp(-3.5625)) = 0.9724

Градиент ошибки выходного элемента рассчитаем по формуле (4)

δ7 = 0.9724 * (1 – 0.9724) * (0.8 – 0.9724) = -0.0046.

Градиенты ошибок скрытых элементов рассчитаем по формуле (5)

δ6 = 0.8550 * (1 – 0.8550) * (3 * -0.0046) = -0.0017

δ5 = 0.9987 * (1 – 0.9987) * (2*-0.0046) = 0

δ4 = 0.9526*(1-0.9526)*(2*0+3*-0.0017) = -0.0051

δ3 = 0.9168*(1-0.9168)*(3*0+1*-0.0017) = -0.0001

Вычислим значения приращений весовых коэффициентов по формуле (6). Для первого образца нет необходимости добавлять инерционный член, поскольку для этого образца нет изменения предыдущего веса.

$\Delta w(0,7) = 0.07 * (-0.0046) = -0.0003$

$\Delta w(5,7) = 0.07 * (-0.0046) * 0.9987 = -0.0003$

$\Delta w(6,7) = 0.07 * (-0.0046) * 0.8550 = -0.0003$

$\Delta w(0,6) = 0.07 * (-0.0017) * 1 = -0.0001$

$\Delta w(3,6) = 0.07 * (-0.0017) * 0.9168 = -0.0001$

$\Delta w(4,6) = 0.07 * (-0.0017) * 0.9526 = -0.0001$

$\Delta w(0,5) = 0.07 * 0 * 1 = 0$

$\Delta w(3,5) = 0.07 * 0 * 0.9168 = 0$

$\Delta w(4,5) = 0.07 * 0 * 0.9526 = 0$

$\Delta w(0,4) = 0.07 * (-0.0002) * 1 = -0.000014$

$\Delta w(1,4) = 0.07 * (-0.0002) * 0.2 = -0.0000028$

$\Delta w(2,4) = 0.07 * (-0.0002) * 0.8 = -0.0000112$

$\Delta w(0,3) = 0.07 * (-0.0001) * 1 = -0.000007$

$\Delta w(1,3) = 0.07 * (-0.0001) * 0.2 = -0.0000014$

$\Delta w(2,3) = 0.07 * (-0.0001) * 0.8 = -0.0000056$

Вычислим новые значения весовых коэффициентов по формуле (7)

w(0,7) = -1 + -0.0003 = -1.0003

w(5,7) = 2 + -0.0003 = 1.9997

w(6,7) = 3 + -0.0003 = 2.9997

w(0,6) = -2 + -0.0001 = -2.0001

w(3,6) = 1 + -0.0001 = 0.9999

w(4,6) = 3 + -0.0001 = 2.9999

w(0,5) = 2 + 0 = 2

w(3,5) = 3 + 0 = 3

w(4,5) = 2 + 0 = 2

w(0,4) = 1 + -0.000014 = 0.999986

w(1,4) = 2 + -0.0000028 = 1.9999972

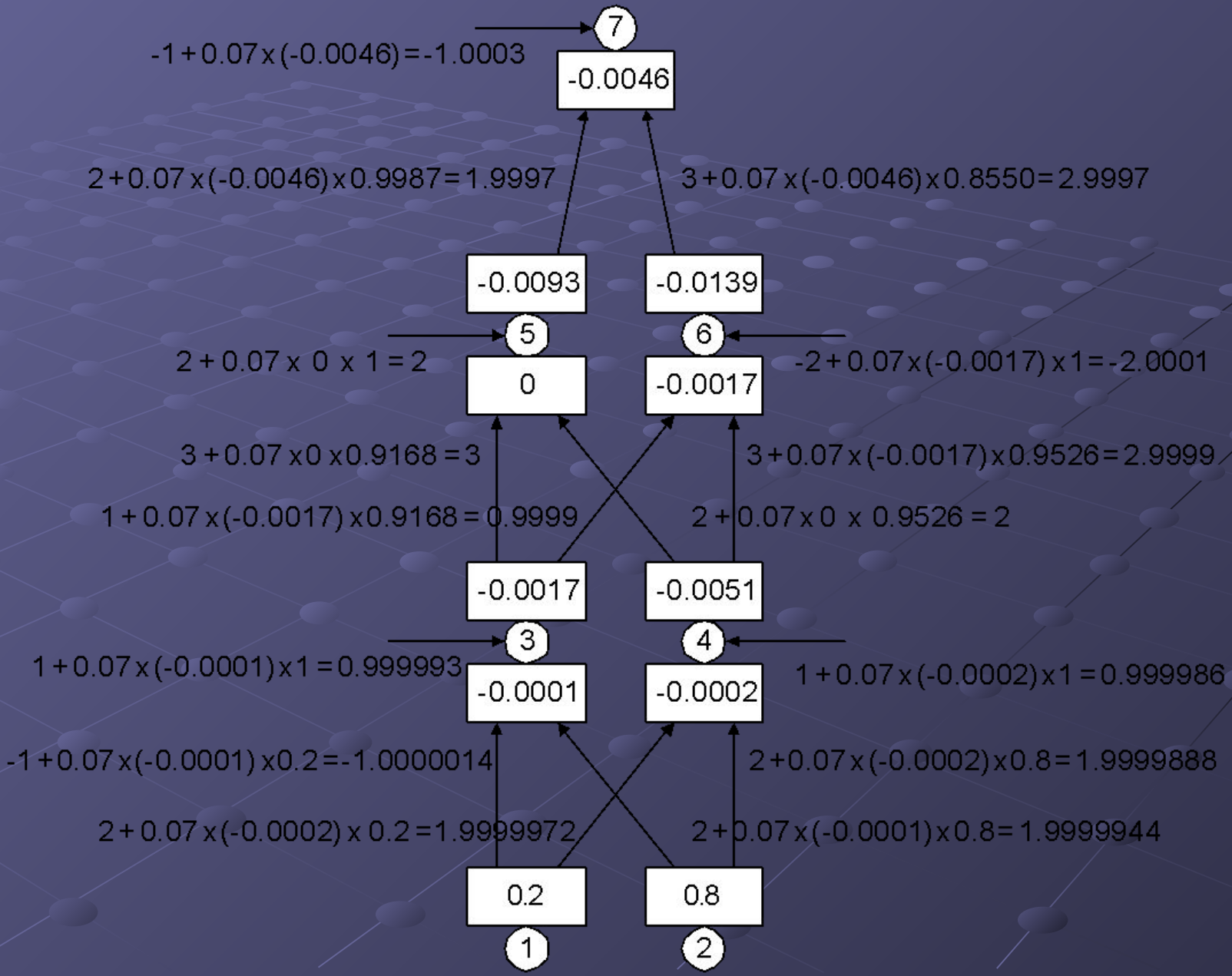
w(2,4) = 2 + -0.0000112 = 1.9999888

w(0,3) = 1 + -0.000007 = 0.999993

w(1,3) = -1 + -0.0000014 = -1.0000014

w(2,3) = 2 + -0.0000056 = 1.9999944

Расчет новых значений весов



И ЭТО ТОЛЬКО НАЧАЛО :)