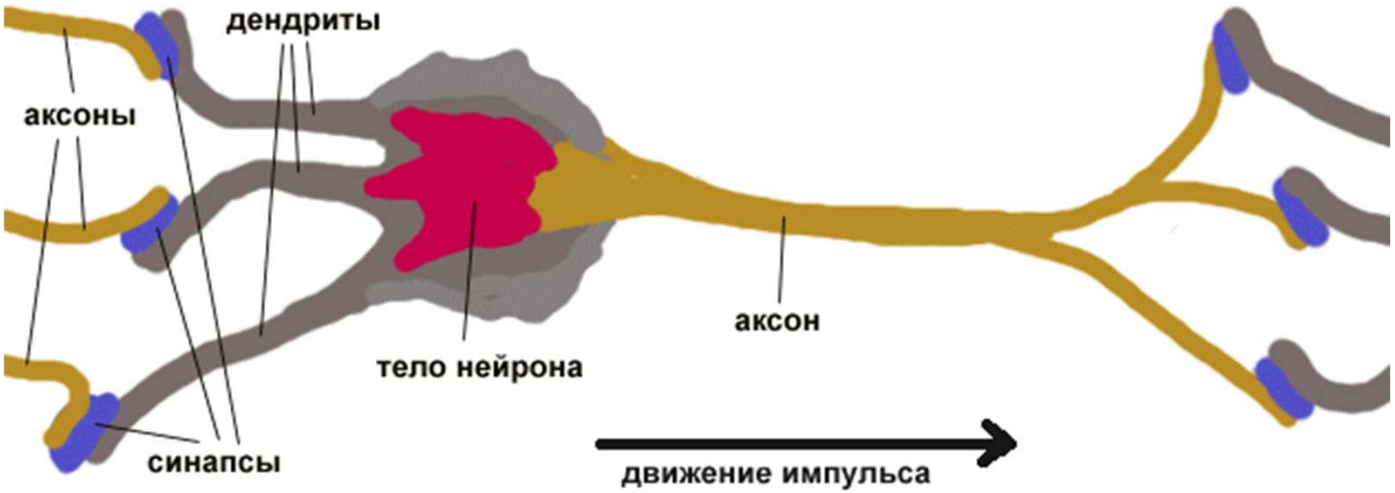
Лекция 1

Домашова Д.В.

Нейронные сети



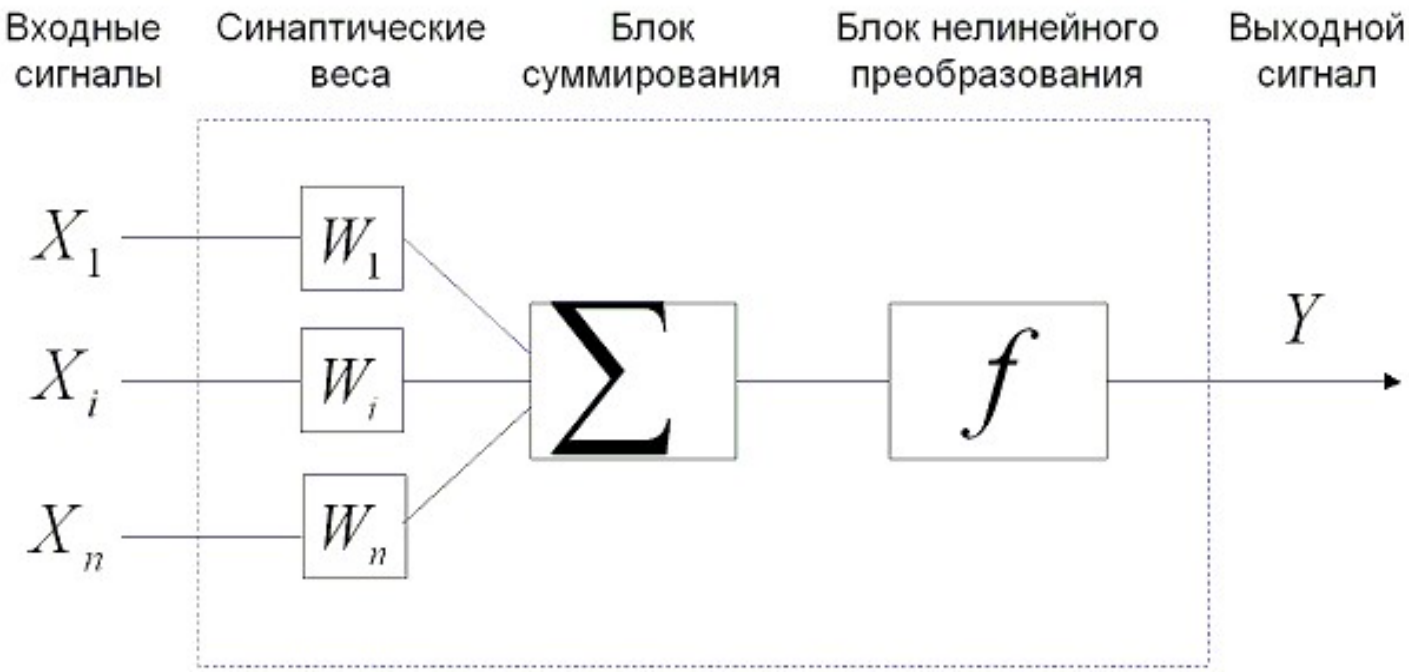
Нервная система человека, построенная из элементов, называемых нейронами, имеет ошеломляющую сложность.
Около 10^{11} нейронов участвуют в примерно 10^{15} передающих связях, имеющих длину метр и более.
Уникальной способностью нейрона является прием, обработка и передача электрохимических сигналов по нервным путям, которые образуют коммуникационную систему мозга.

Нейронные сети

- Дендриты идут от тела нервной клетки к другим нейронам, где они принимают сигналы в точках соединения, называемых синапсами.
- Принятые синапсом входные сигналы подводятся к телу нейрона. Здесь они суммируются, причем одни входы стремятся возбудить нейрон, другие – воспрепятствовать его возбуждению.
- Когда суммарное возбуждение в теле нейрона превышает некоторый порог, нейрон возбуждается, посылая по аксону сигнал другим нейронам.

Искусственный нейрон

- **Нейрон** состоит из элементов трех типов: **умножителей** (синапсов), **сумматора** и **нелинейного преобразователя**.



Искусственный нейрон

- **Синапсы** осуществляют связь между нейронами, умножают входной сигнал на число, характеризующее силу связи (вес синапса).
- **Сумматор** выполняет сложение сигналов, поступающих по синаптическим связям от других нейронов, и внешних входных сигналов.
- **Нелинейный преобразователь** реализует нелинейную функцию одного аргумента- выхода сумматора.
- Эта функция называется **функцией активации** или **передаточной функцией** нейрона.

Искусственный нейрон

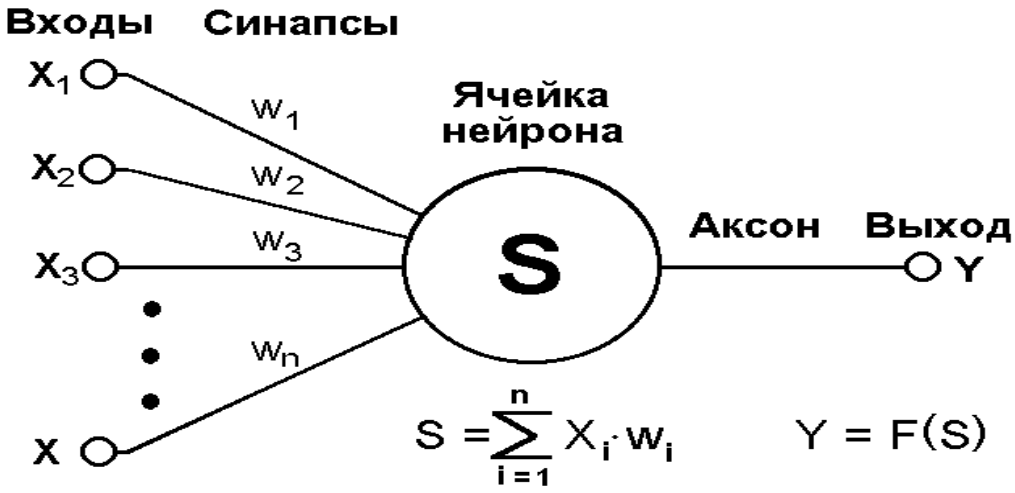
Алгоритм работы:

- нейрон получает набор (вектор) входных сигналов.
- в теле нейрона оценивается суммарное значение входных сигналов.
- каждый вход характеризуется некоторым весовым коэффициентом, определяющим важность поступающей по нему информации (нейрон не просто суммирует значения входных сигналов, а вычисляет скалярное произведение вектора входных сигналов и вектора весовых коэффициентов).
- нейрон формирует выходной сигнал, интенсивность которого зависит от значения вычисленного скалярного произведения.
- Если оно не превышает некоторого заданного порога, то выходной сигнал не формируется вовсе - нейрон «не срабатывает».

Математическая модель нейрона

$$s = \sum_{i=1}^{n} w_i x_i + b$$

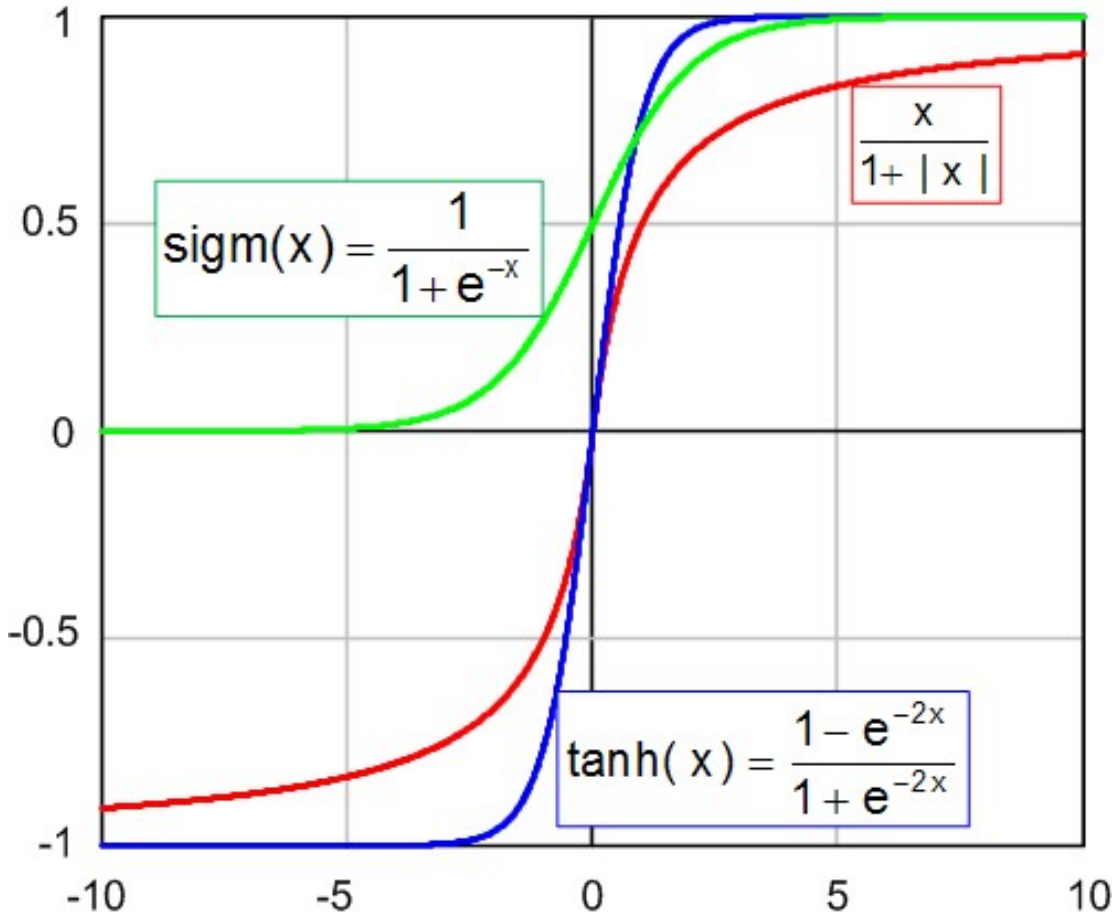
$$y = f(s)$$



- В общем случае **входной сигнал**, **весовые коэффициенты** и **смещение** могут принимать действительные значения, а во многих практических задачах- некоторые фиксированные значения.
- **Выход (y)** определяется видом **функции активации** и может быть как действительным, так и целым.
- На **входной сигнал (s)** нелинейный преобразователь отвечает **выходным сигналом f(s),** который представляет собой выход нейрона.

Функции активации нейрона F(s)

$$F(s) = \frac{1}{1 + e^{-as}}$$



- Одной из наиболее распространенных является нелинейная функция активации с насыщением, так называемая **логистическая** функция или **сигмоид**

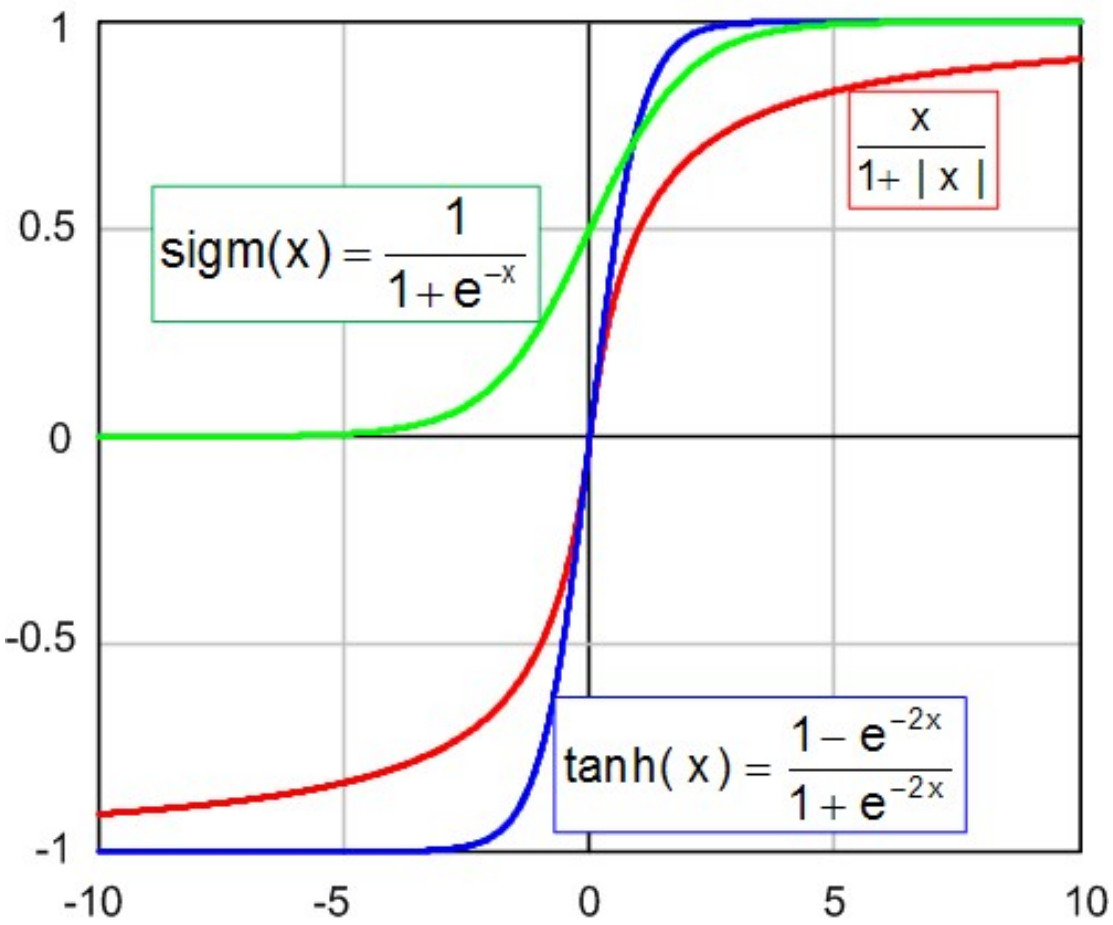
Функции активации нейрона F(s)

$$F(s) = \frac{1}{1 + e^{-as}}$$

- При уменьшении *a* **сигмоид** становится более пологим, в пределе *a=0* вырождаясь в горизонтальную линию на уровне 0,5;
- при увеличении *a* сигмоид приближается к виду функции единичного скачка с некоторым порогом.
- Из выражения для сигмоида очевидно, что выходное значение нейрона лежит в диапазоне (0,1).
- Для производной сигмоидной функции верно следующее соотношение:

$$f'(x) = \alpha \cdot f(x) \cdot (1 - f(x))$$

Функции активации нейрона F(s)



- Другой широко используемой активационной функцией является гиперболический тангенс.

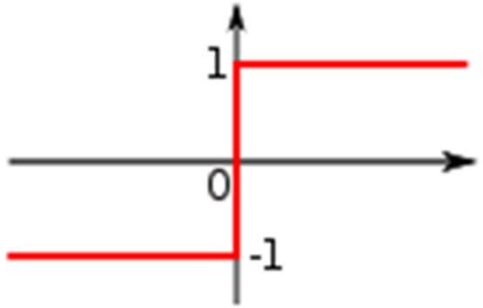
Функции активации нейрона F(s)

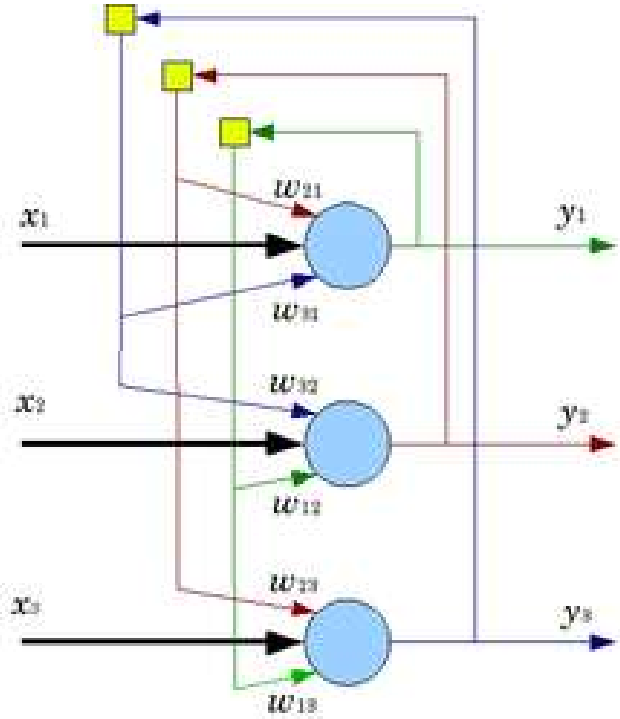
$$\operatorname{th} x = \frac{\operatorname{sh} x}{\operatorname{ch} x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

- Подобно логистической функции гиперболический тангенс является s-образной функцией, но он симметричен относительно начала координат, и в точке о значение выходного сигнала равно нулю.
- В отличие от логистической функции гиперболический тангенс принимает значения различных знаков, что оказывается выгодным для ряда сетей.

Функции активации нейрона F(s)

$$y_i = \begin{cases} 1, \\ -1 \end{cases}$$



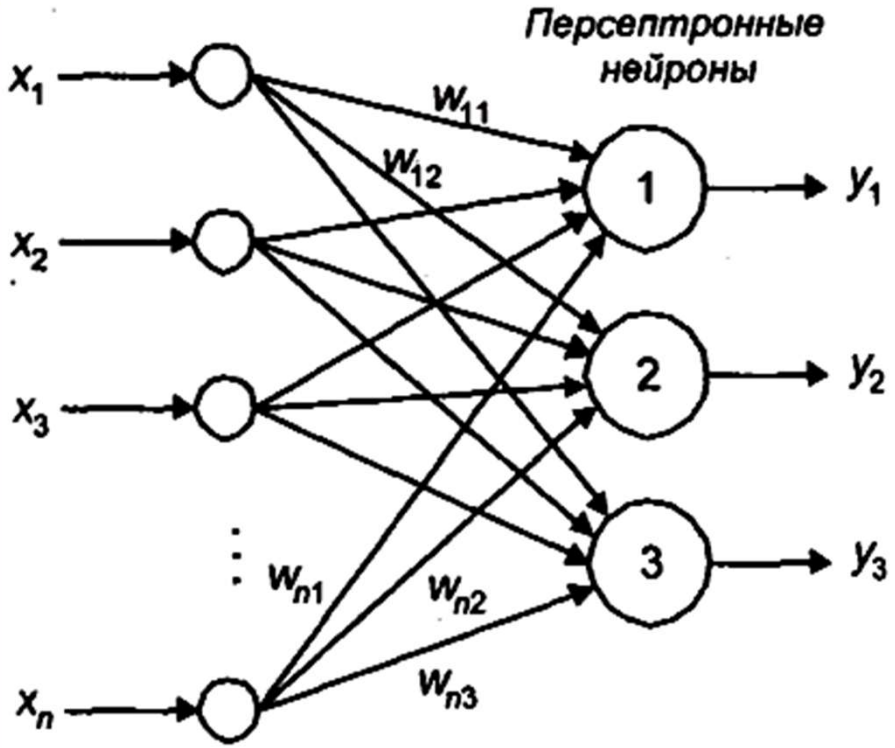


- передаточная функция сети Хопфилда. Тип входного и выходного сигналов: биполярные (+1 и -1).

Модель нейронной сети. Персептрон.

- **Персептро́н**, или перцептрон (англ. perceptron от лат. perceptio — восприятие; нем. perzeptron) — математическая и компьютерная модель восприятия информации мозгом (кибернетическая модель мозга), предложенная Фрэнком Розенблаттом в 1957 году и реализованная в виде электронной машины «Марк-1» в 1960 году.
- Перцептрон стал **одной из первых моделей нейросетей**, а «Марк-1» — первым в мире нейрокомпьютером.
- Несмотря на свою простоту, перцептрон **способен обучаться** и решать довольно сложные задачи.
- Основная математическая задача, с которой он справляется, — это линейное разделение любых нелинейных множеств, так называемое обеспечение линейной сепарабельности.

Нейронные сети



$$y_j = f\left(\sum_{i=1}^{n} x_i\, w_{ij}\right),\ j = 1\ldots3.$$

$$f(x) = \frac{1}{1 + e^{-\alpha x}}$$

$$f'(x) = \alpha \cdot f(x) \cdot (1 - f(x))$$

Персептрон - один слой искусственных нейронов, соединенных с помощью весовых коэффициентов с множеством входов.

На входы подаются входные сигналы, поступающие далее по синапсам на нейроны, которые образуют единственный слой этой сети.

$x_1, x_2 \ldots$-известные значения, y_1, y_2-определяемые значения

На выходах сети формируются выходные сигналы

Математическая модель работы нейрона в случае если F имеет вид жесткой ступеньки

- Работа всех сетей сводится к классификации (обобщению) входных сигналов, принадлежащих n-мерному гиперпространству, по некоторому числу классов.
- С математической точки зрения это происходит путем разбиения гиперпространства гиперплоскостями (запись для случая однослойного перцептрона)
- Каждая полученная область является областью определения отдельного класса. Число таких классов для одной НС перцептронного типа не превышает 2^m, где m – число выходов сети.

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

Математическая модель работы нейрона в случае если F имеет вид жесткой ступеньки

- С математической точки зрения это происходит путем разбиения гиперпространства гиперплоскостями. Каждая полученная область является областью определения отдельного класса.
- Число таких классов для персептрона не превышает 2m, где m - число его выходов.
- По сути, нейронная сеть или персептрон - это алгоритм, использующий уравнение линейного неравенства (линейного фильтра), с помощью которого можно причислить исследуемый объект к тому или иному классу или, наоборот, исключить его из этого самого класса объектов.

Математическая модель работы нейрона в случае если F имеет вид жесткой ступеньки

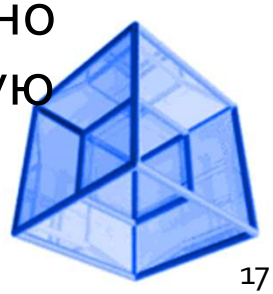
- Геометрически плоскость описывается линейным уравнением. Например, в трехмерном пространстве уравнение плоскости относительно координат X, Y и Z имеет вид:

$$A \cdot X + B \cdot Y + C \cdot Z + D = 0$$

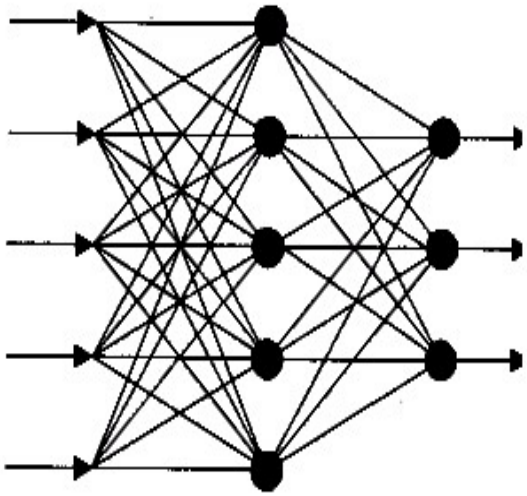
- Координаты всех точек, расположенных по одну сторону от плоскости в этом пространстве, удовлетворяют неравенству:

$$A \cdot X + B \cdot Y + C \cdot Z + D > 0$$

- Соответственно, весовые коэффициенты в неравенстве нейронной сети - это константы, которые задают **уравнение некой плоскости в многомерном пространстве** признаков объектов. А с помощью самого неравенства можно точно определить, лежат ли эти объекты по одну или по другую сторону заданной плоскости.



Нейронные сети



- В многослойных сетях нейроны объединяются в слои.
- Слой содержит совокупность нейронов с едиными входными сигналами.
- Число нейронов в слое может быть любым и не зависит от количества нейронов в других слоях.
- **Определение числа скрытых слоев и числа нейронов в каждом слое для конкретной задачи является неформальной задачей.**
- Нейроны определенным образом соединены друг с другом и с внешней средой с помощью связей, определяемых весовыми коэффициентами.
- Внешние входные сигналы подаются на входы нейронов входного слоя, а выходами сети являются выходные сигналы последнего слоя.

Нейронные сети

- Обучение НС может вестись с учителем или без него.
- В первом случае сети предъявляются значения как входных, так и желательных выходных сигналов, и она по некоторому внутреннему алгоритму подстраивает веса своих синаптических связей.
- Во втором случае выходы НС формируются самостоятельно, а веса изменяются по алгоритму, учитывающему только входные и производные от них сигналы.

Нейронные сети: обучение с учителем

- Процесс функционирования нейронной сети зависит от величин синаптических связей (значения wij).

Обучение с учителем: для заданной структуры сети, соответствующей какой-либо задаче, необходимо найти оптимальные значения всех весовых коэффициентов для некоторого набора известных значений входов и выходов (обучающие примеры).

- Обучение нейронной сети является задачей многомерной оптимизации, для решения которой используются существующие оптимизационные методы.
- От качества обучения зависит способность сети решать поставленные перед ней задачи во время функционирования.

Нейронные сети: обучение с учителем

Алгоритм обучения:

- ШАГ 1. Задать исходные значения весовых коэффициентов (случайные значения).
- ШАГ 2. Подать на входы один из входных векторов, которые сеть должна научиться различать, и вычислить ее выход (поочередно в случайном порядке предъявляются все возможные входные вектора).
- ШАГ 3. Если выход правильный, перейти на шаг 4. Иначе - модифицировать веса по некоторому правилу.
- Шаг 4. Цикл с шага 2, пока сеть не перестанет ошибаться.

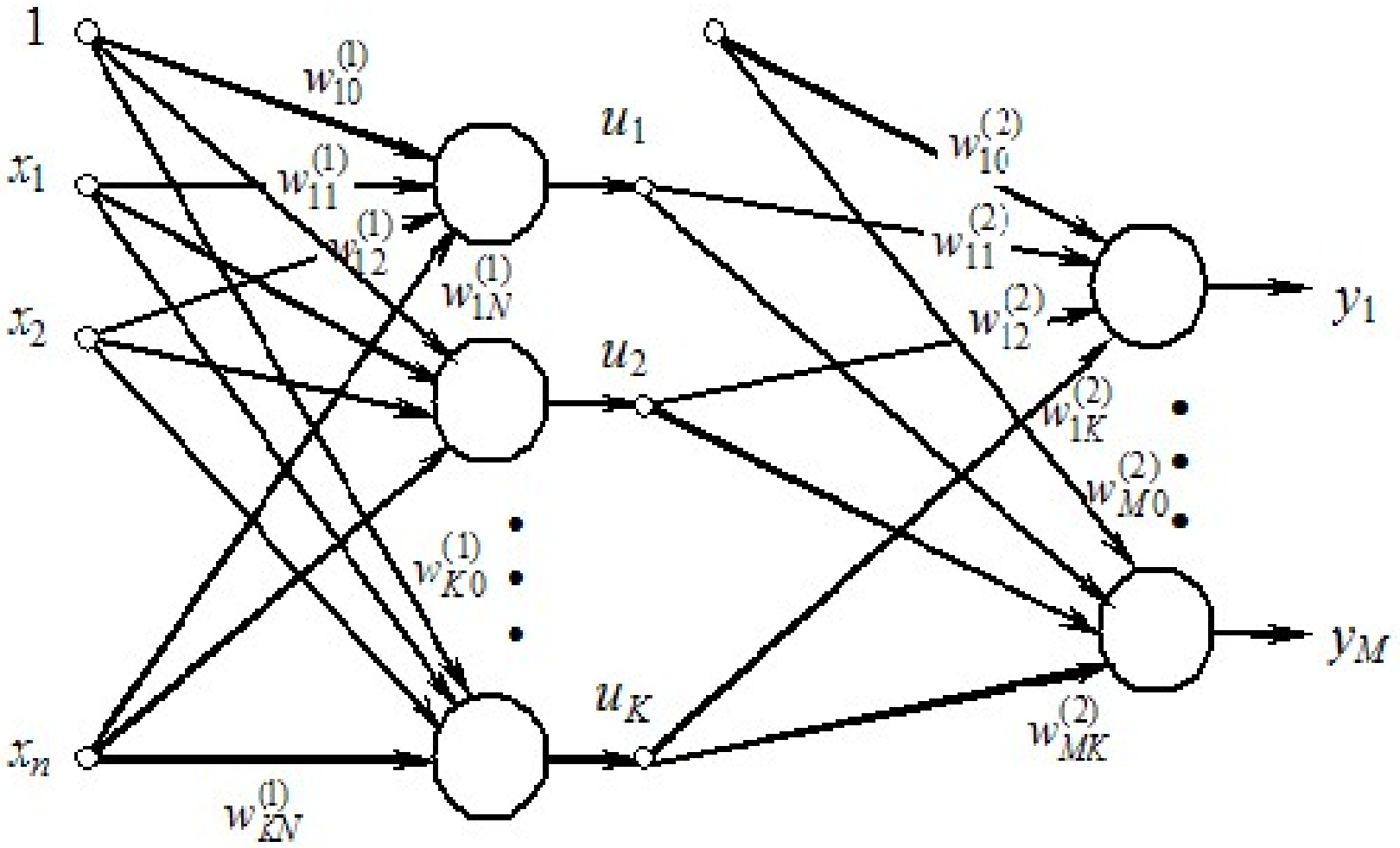
К сожалению, нельзя заранее определить число итераций, которые потребуется выполнить, а в некоторых случаях и гарантировать полный успех.

Алгоритм обратного распространения ошибки

Метод обратного распространения ошибки предложен несколькими авторами независимо в 1986 г. для многослойных сетей с прямой передачей сигнала (feed-forward).

Многослойная нейронная сеть способна осуществлять любое отображение входных векторов в выходные

Алгоритм обратного распространения ошибки



Алгоритм обратного распространения ошибки

Цель обучения состоит в подборе таких значений весов и для слоев сети, чтобы при заданном входном векторе x получить на выходе значения сигналов y_l, которые с требуемой точностью будут совпадать с ожидаемыми значениями d_l для $l = 1, 2,\ldots, M$.

Если рассматривать единичный сигнал как одну из компонент входного вектора x, то веса смещения можно добавить в векторы весов соответствующих нейронов обоих слоев. При таком подходе выходной сигнал i-го нейрона скрытого слоя описывается функцией

$$u_i = f\left(\sum_{j=0}^{N} w_{ij}^{(1)} x_j\right)$$

в которой индекс 0 соответствует смещению, причем

$u_0 \equiv 1, \quad x_0 \equiv 1.$

Алгоритм обратного распространения ошибки

В выходном слое *l*-й нейрон вырабатывает выходной сигнал, определяемый как:

$$y_l = f\left(\sum_{i=1}^{m} w_{li}^{(2)} u_i\right) = f\left(\sum_{i=1}^{K} w_{li}^{(2)} f\left(\sum_{j=1}^{N} w_{ij}^{(1)} x_j\right)\right)$$

Из формулы , приведенной выше, следует, что на значение выходного сигнала влияют веса обоих слоев, тогда как сигналы, вырабатываемые в скрытом слое, не зависят от весов выходного слоя.

Алгоритм обратного распространения ошибки

- **Алгоритм обратного распространения ошибки** - итеративный градиентный алгоритм обучения, который используется с целью минимизации среднеквадратичного отклонения текущих от требуемых выходов многослойных нейронных сетей с последовательными связями.
- На каждом шаге алгоритма на вход сети поочередно подаются все обучающие примеры, реальные выходные значения сети сравниваются с требуемыми значениями, и вычисляется ошибка.
- Значение ошибки, а также градиента поверхности ошибок используется для корректировки весов, после чего все действия повторяются.
- Процесс обучения прекращается либо когда пройдено определенное количество шагов обучения, либо когда ошибка достигнет некоторого определенного малого уровня, либо когда ошибка перестанет уменьшаться.
- **Недостаток** - на каждой итерации происходят изменения значений параметров сети, улучшающие работу лишь с одним примером обучающей выборки. Такой подход существенно уменьшает скорость обучения.

Алгоритм обратного распространения ошибки

Согласно методу наименьших квадратов, минимизируемой целевой функцией ошибки НС является величина:

$$E(w) = \frac{1}{2}\sum_{j}\sum_{p}(y_{jp}^{(N)} - d_{jp})^2$$

где $y_{jp}^{(N)}$ – реальное выходное состояние нейрона j выходного слоя N нейронной сети при подаче на ее входы p-го образа;

d_{jp} – идеальное (желаемое) выходное состояние этого нейрона.

Суммирование ведется по всем нейронам выходного слоя и по всем обрабатываемым сетью образам.

Алгоритм обратного распространения ошибки

Минимизация ведется методом градиентного спуска, что означает подстройку весовых коэффициентов следующим образом: $w_{ij}(t+1) = w_{ij}(t) - \alpha \cdot \frac{\partial E(k)}{\partial w_{ij}(t)}$

i=1,2,…,n; j=1,2,…,p,

Здесь w_{ij} – весовой коэффициент синаптической связи, соединяющей i-ый нейрон слоя n-1 с j-ым нейроном слоя n,
α – коэффициент скорости обучения, $0<\alpha<1$,
E – среднеквадратичная ошибка сети для одного из p образов, определяемая по формуле:

$$E = \frac{1}{2}\sum_{l=1}^{m}(y_l - d_l)^2$$

Алгоритм обратного распространения ошибки

- Алгоритм обратного распространения — способ ускоренного расчета компонент градиента.
- Идея метода в том, чтобы представить E в виде сложной функции и последовательно рассчитать частные производные по формуле для сложной функции:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \cdot \frac{\partial s_j}{\partial w_{ij}}$$

Алгоритм обратного распространения ошибки

$$E(w) = \frac{1}{2}\sum_{l=1}^{m}[y_l - d_l]^2 =$$

$$= \frac{1}{2}\sum_{l=1}^{m}\left[f\left(\sum_{i=1}^{k} w_{li}^{(2)} u_i\right) - d_l\right]^2 =$$

$$= \frac{1}{2}\sum_{l=1}^{m}\left[f\left(\sum_{i=1}^{k} w_{li}^{(2)} f\left(\sum_{j=1}^{n} w_{ij}^{(1)} x_j\right)\right) - d_l\right]^2$$

Алгоритм обратного распространения ошибки

$$E(w) = \frac{1}{2}\sum_{l=1}^{m}[y_l - d_l]^2 =$$

$$= \frac{1}{2}\sum_{l=1}^{m}\left[f\left(\sum_{i=1}^{k} w_{li}^{(2)} u_i\right) - d_l\right]^2 =$$

$$= \frac{1}{2}\sum_{l=1}^{m}\left[f\left(\sum_{i=1}^{k} w_{li}^{(2)} f\left(\sum_{j=1}^{n} w_{ij}^{(1)} x_j\right)\right) - d_l\right]^2$$

1) Для весов выходного слоя

$$\frac{\partial E}{\partial w_{li}^{(2)}} = (y_l - d_l)\frac{df\left(s_l^{(2)}\right)}{ds_l^{(2)}}\frac{ds_l^{(2)}}{dw_{li}^{(2)}} = (y_l - d_l)\frac{df\left(s_l^{(2)}\right)}{ds_l^{(2)}}u_i$$

$$s_l^{(2)} = \sum_{i=1}^{k} w_{li}^{(2)} u_i \quad \text{Пусть: } \delta_l^{(2)} = (y_l - d_l)\frac{df\left(s_l^{(2)}\right)}{ds_l^{(2)}}$$

Тогда $\frac{\partial E}{\partial w_{li}^{(2)}} = \delta_l^{(2)} u_i$

Алгоритм обратного распространения ошибки

2) для весов нейронов скрытого слоя

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \sum_{l=1}^{m} (y_l - d_l) \frac{dy_l}{du_i} \frac{du_i}{dw_{ij}^{(1)}} = \sum_{l=1}^{m} (y_l - d_l) \frac{df\left(s_l^{(2)}\right)}{ds_l^{(2)}} w_{li}^{(2)} \frac{df\left(s_i^{(1)}\right)}{ds_i^{(1)}} x_j$$

$$\frac{\partial E}{\partial w_{ij}^{(1)}} = \delta_i^{(1)} x_j,$$

где $\delta_i^{(1)} = \sum_{l=1}^{m} \underbrace{(y_l - d_l) \frac{df\left(s_l^{(2)}\right)}{ds_l^{(2)}}}_{\delta_l^{(2)}} w_{li}^{(2)} \frac{df\left(s_i^{(1)}\right)}{ds_i^{(1)}} =$

$$= \sum_{l=1}^{m} \delta_l^{(2)} w_{li}^{(2)} \frac{df\left(s_i^{(1)}\right)}{ds_i^{(1)}} = \sum_{l=1}^{m} \delta_l^{(2)} w_{li}^{(2)} y_i'$$

$$\frac{df\left(s_i^{(1)}\right)}{ds_i^{(1)}} = y_i'$$

Алгоритм обратного распространения ошибки

1. Подать на входы сети один из возможных образов и в режиме обычного функционирования НС, когда сигналы распространяются от входов к выходам, рассчитать значения последних.

2. Рассчитать $\delta^{(N)}$ для выходного слоя.

$$\delta_j^{(N)} = (y_j - d_j) y_j'$$

Рассчитать изменения веса $\Delta w^{(N)}$ слоя N

$$\Delta w_{ij}(N) = -\eta \delta_j^{(N)} y_i^{(N-1)}.$$

3. Рассчитать $\delta^{(n)}$ и $\Delta w^{(n)}$ для всех остальных слоев, n=N-1,...1.

$$\delta_i^{(n)} = y_i^{(n)'} \sum \delta_j^{(n+1)} w_{ij}^{(n+1)}$$

$$\Delta w_{ij}(n+1) = -\eta \delta_j^{(n+1)} y_i^{(n)}$$

4. Скорректировать все веса в НС

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

5. Если ошибка сети существенна, перейти на шаг 1.

Нейронные сети

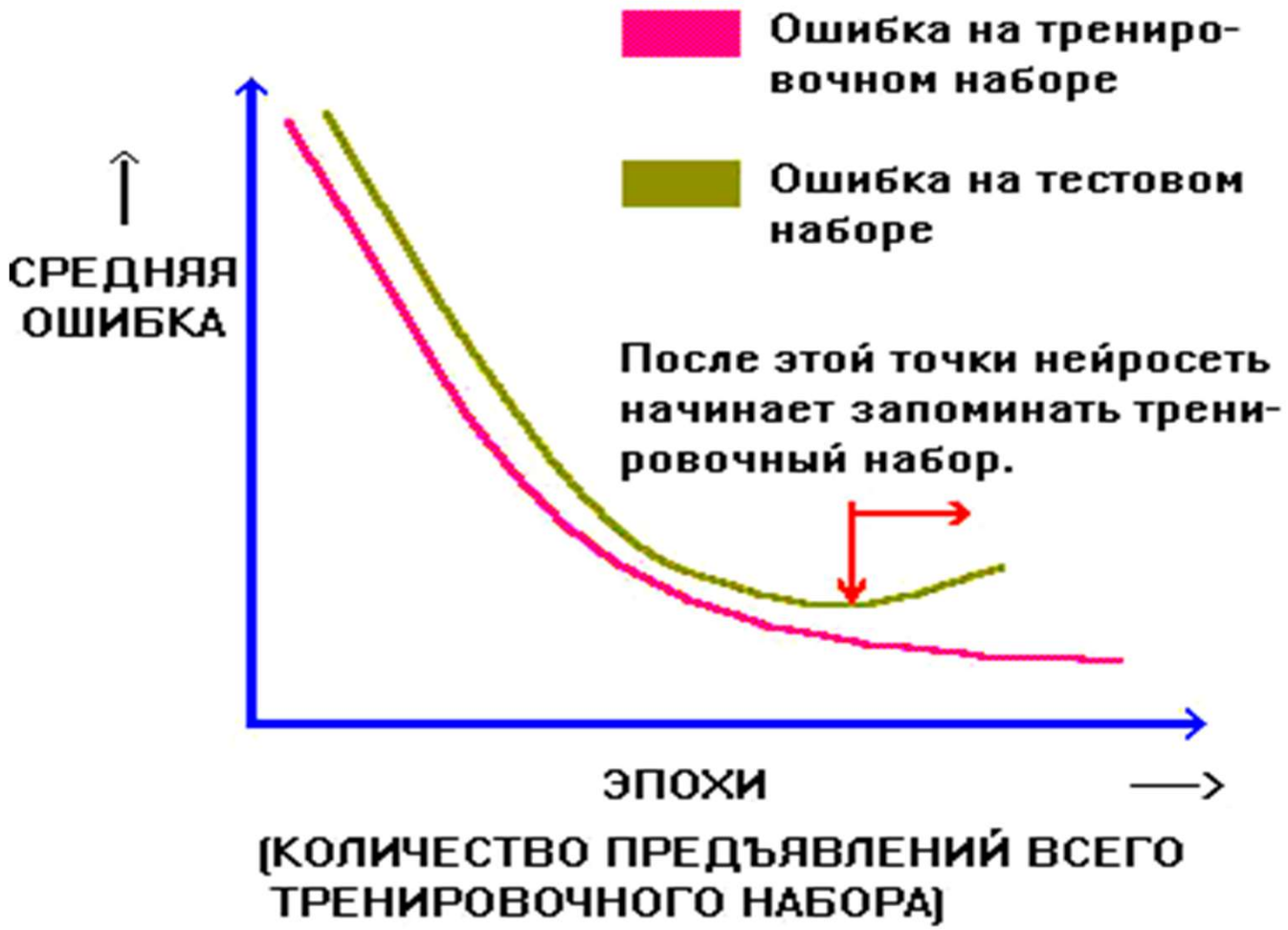
Проблемы:

- В процессе обучения большие положительные или отрицательные значения весов могут сместить рабочую точку на сигмоидах нейронов в область насыщения.
- Малые величины производной от логистической функции могут привести к остановке обучения.
- Применение метода градиентного спуска не гарантирует нахождения глобального минимума целевой функции.
- Приращения весов и, следовательно, скорость обучения для нахождения экстремума должны быть бесконечно малыми, однако в этом случае обучение будет происходить неприемлемо медленно.
- Слишком большие коррекции весов могут привести к постоянной неустойчивости процесса обучения.

Нейронные сети

- **Проблемы обобщения и переобучения нейронной сети:**
- **Обобщение** - способность нейронной сети делать точный прогноз на данных, не принадлежащих исходному обучающему множеству
- **Переобучение** - чрезмерно точная подгонка, которая имеет место, если алгоритм обучения работает слишком долго, а сеть слишком сложна для такой задачи или для имеющегося объема данных.
- Сети с большим числом весов моделируют более сложные функции и, следовательно, склонны к переобучению.
- Сети же с небольшим числом весов могут оказаться недостаточно гибкими, чтобы смоделировать имеющиеся зависимости.

Переобучение сети



Нейронные сети

- Более сложная сеть дает меньшую ошибку, но это может свидетельствовать не о хорошем качестве модели, а о переобучении сети.
- Используется тестовая кросс-проверка - резервируется часть обучающей выборки, которая используется для независимого контроля результата в ходе алгоритма.
- По мере обучения сети ошибка обучения убывает, как и ошибка на тестовом множестве. Если же тестовая ошибка перестала убывать или даже стала расти, то сеть начала слишком близко аппроксимировать данные (переобучилась) и обучение следует остановить.
- Следует уменьшить число скрытых элементов и/или слоев, ибо сеть является слишком мощной для данной задачи.
- Если обе ошибки (обучения и кросс-проверки) не достигнут достаточного малого уровня, то переобучения не произошло, а сеть, напротив, является недостаточно мощной для моделирования имеющейся зависимости.

Нейронные сети

- Количество скрытых нейронов сильно зависит от количества фактов обучающей выборки. Если обучающая выборка мала, а количество нейронов велико, то сеть начинает "запоминать" факты (переобучение).
- Обратная ситуация может привести к тому, что сеть никогда не обучится.
- Для решения задачи о количестве нейронов в скрытом слое и размере обучающей выборки предлагается следующая методика.

Количество обучающих фактов F:	**Количество скрытых нейронов H:**
2 * (I + H + O) < = F < = 10 * (I + H + O), **где:** **I - количество входов сети,** **H - количество спрятанных нейронов,** **O - количество выходов сети.**	**F / 10 - I - O < = H < = F / 2 - I - O,** **где:** **I - количество входов сети,** **F - количество фактов обучающей выборки,** **O - количество выходов сети.**

Нейронные сети

Этапы решения практических задач с использованием нейронных сетей:

1. Формализация задачи: определение проблемы и выбор вектора параметров

Определить, какой смысл вкладывается в компоненты входного вектора x. Входной вектор должен содержать формализованное условие задачи, т.е. всю информацию, необходимую для получения ответа.

Выбрать выходной вектор y таким образом, чтобы его компоненты содержали полный ответ поставленной задачи.

2. Определение и подготовка исходных данных.
3. Преобразования и анализ исходных данных (масштабируемость данных, преобразование качественных данных в числовые, проверка коррелируемости, анализ периодичности).
4. Задание структуры сети
5. Обучение сети (динамическое сокращение ошибки обучения, контроль "здоровья" сети, включение шумов в обучающий процесс, изменение порядка набора обучающих фактов).
6. Использование сети

Нейронные сети. Решение задач.

- **Для полноценного и быстрого обучения сети наиболее значимым является адекватное представление входных и обучающих данных**. Практически во всех случаях представление сети "сырых" данных приводят к неустойчивому обучению и значительной ошибке выхода. Более того, сеть может вообще никогда не научиться тому, что после соответствующих мер изучается сетью практически мгновенно.
- К **основным преобразованиям** исходных данных относятся масштабируемость данных, проверка коррелируемости, анализ периодичности. Их цель облегчить нейросетевой модели поиск функциональной зависимости между входными и выходными величинами.

Нейронные сети

Формализация задачи: определение проблемы и выбор вектора параметров

Многослойный перцептрон может рассчитывать выходной вектор **y** для любого входного вектора **x**, т.е. давать значение некоторой векторной функции $\mathbf{y} = \mathbf{f}(\mathbf{x})$. Следовательно, условие любой задачи, которая может быть поставлена перцептрону, должно являться множеством векторов $\mathbf{x}^1 \ldots \mathbf{x}^p$.

Решением задачи будет множество векторов $\mathbf{y}^1 \ldots \mathbf{y}^p$,

$\mathbf{y}^s = f(\mathbf{x}^s)$, где $s = 1..p$ — номер предъявленного образа.

Все, что способен сделать перцептрон — это сформировать отображение

$X \rightarrow Y$ для $\forall x \in X$.

Множество векторов $\mathbf{x}^1 \ldots \mathbf{x}^p$ — *формализованное условие задачи*, а множество $\mathbf{y}^1 \ldots \mathbf{y}^p$ — *формализованное решение*.

Задача формализации, т.е. выбора смысла, которым наделяются компоненты входного и выходного векторов, пока решается только человеком на основе практического опыта. Жестких рецептов формализации для нейронных сетей пока не создано.

Нейронные сети Примеры формализации задач

1. Задача классификации.

Пусть есть некоторый объект, который характеризуется несколькими параметрами $x_1..x_n$. Пусть также есть M классов объектов, $C1..CM$. Мы наблюдаем объект и можем рассчитать или измерить его параметры. Вектор **X** характеризует наблюдаемый объект:

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix}$$

На основании вектора **X** мы должны решить, к какому классу отнести объект, т.е. выбрать Cm, к которому принадлежит объект, характеризуемый набором параметров **x**.

Решение задачи можно представить в виде вектора:

$$C = \begin{pmatrix} c_1 \\ c_2 \\ \cdots \\ c_M \end{pmatrix}$$

Здесь c_m— вероятность, с которой объект относится к классу Cm .
Если рассматривать c_m как вероятности, то должны выполняться условия (1).

$$0 \le c_m \le 1 \qquad \sum_{m=1}^{M} c_m = 1 \qquad (1)$$

Нейронные сети. Решение задач.

Второй способ формализации задачи классификации (некорректный)

Пусть у сети только один выход, и пусть его смысл — номер класса m для вектора $\mathbf{p}$, предъявленного на входе.

Следовательно, сеть обучается зависимости $m(\mathbf{p})$.

Если обучение прошло успешно, то когда на вход сети подан вектор $\mathbf{p}$, характеризующий объект, на выходе будет получено число m, и принимается решение о принадлежности $\mathbf{p}$ к классу Cm.

Нейронные сети. Решение задач.

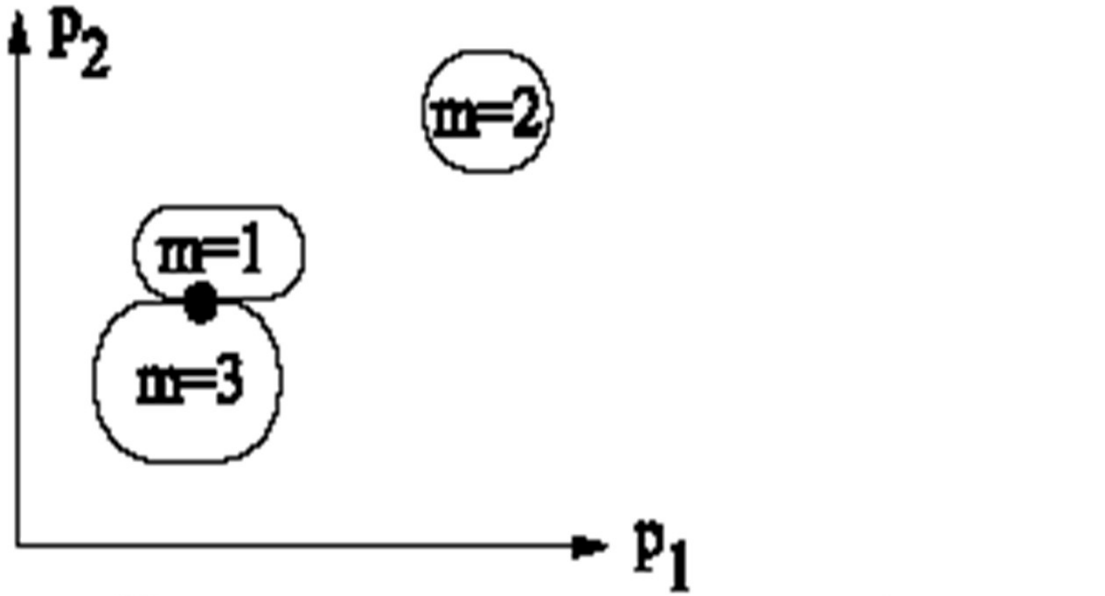


Рис. . Пример некорректной классификации.

Пусть требуется разделить объекты по двум признакам, p1, p2, на три класса, *m*=1, *m*=2, *m*=3. Если входной вектор **p** примет значение, обозначенное жирной точкой, то выход сети, при правильном обучении, примет значение *m*=2, т.е. объект будет отнесен к классу 2, совершенно неподходящему.

Нейронные сети. Решение задач.

- Данное явление возникает из-за того, что сеть склонна интерполировать входные и выходные данные.
- Если функции активации плавные, весовые коэффициенты не слишком большие, и количество слоев не слишком велико, то выход сети тоже будет гладким и непрерывным.
- Для близких **p** будут получены близкие *m* на выходе. Но при решении задачи классификации (когда используются категориальные данные) такое допущение бывает неверным. Отсюда неправильное решение.
- Чтобы избежать ошибок, нужно применить другие способы формализации.

Нейронные сети. Решение задач.

- Пусть задана функция, определенная на интервале времени 0…t_0, где t_0 — текущее значение времени. Требуется предсказать значение функции при $t > t_0$.
- Чтобы применить многослойный перцептрон для прогнозирования, время придется дискретизировать.
- Будем считать известными значения функции в моменты времени:

$$\begin{pmatrix} x_0 = f(t_0) \\ x_1 = f(t_0 - \delta_1) \\ x_2 = f(t_0 - \delta_1 - \delta_2) \\ \dots \\ x_n = f(t_0 - \delta_1 - \dots \delta_n) \end{pmatrix} = \mathbf{x}, \quad \delta_i > 0.$$

Будем предсказывать значение функции в момент времени (to + δo) для ∀δo > o.

δo называется интервалом прогнозирования. Решением задачи будем считать значение f (to + δo)=y.

Нейронные сети. Решение задач.

- На НС задача прогнозирования формализуется через задачу распознавания образов.
- Данные о прогнозируемой переменной за некоторый промежуток времени образуют образ, класс которого определяется значением прогнозируемой переменной в некоторый момент времени за пределами данного промежутка т.е. значением переменной через интервал прогнозирования.
- Метод окон предполагает использование двух окон Wi и Wo с фиксированными размерами n и m соответственно.
- Эти окна, способны перемещаться с некоторым шагом по временной последовательности исторических данных, начиная с первого элемента, и предназначены для доступа к данным временного ряда, причем первое окно Wi, получив такие данные, передает их на вход нейронной сети, а второе – Wo – на выход.
- Получающаяся на каждом шаге пара: Wi -> Wo используется как элемент обучающей выборки (распознаваемый образ, или наблюдение).

Нейронные сети. Решение задач.

- Пусть есть данные о еженедельных продажах (k = 16):

100 94 90 96 91 94 95 99 95 98 100 97 99 98 96 98

- Зададим n = 4, m = 1, s = 1.
- С помощью метода окон для нейронной сети будет сгенерирована следующая обучающая выборка:

100 94 90 96 -> 91

94 90 96 91 -> 94

90 96 91 94 -> 95

96 91 94 95 -> 99

91 94 95 99 -> 95

и т.д.

- Каждый следующий вектор получается в результате сдвига окон Wi и Wo вправо на один элемент (s = 1).

Нейронные сети. Решение задач.

Аппроксимация многомерной функции (восстановление значений).

Рассмотрим многомерную функцию **y = f(x)**, где вектор **y** имеет NO компонент, а вектор **x** — NI компонент.

Самый простой способ формализации — использовать сеть с NI входами и NO выходами.

Компоненты вектора **x** подаются на вход сети, **y** — снимаются на выходе.

Сеть обучается на известных значениях функции **f**.

Предобработка данных

Максимизация энтропии как цель предобработки

Рассмотрим произвольную компоненту нормированных (предобработанных) данных: $\tilde{x}_i$. Среднее количество информации, приносимой каждым примером $\tilde{x}_i^{\alpha}$, равно энтропии распределения значений этой компоненты $H(\tilde{x}_i)$. Если эти значения сосредоточены в относительно небольшой области единичного интервала, информационное содержание такой компоненты мало. В пределе нулевой энтропии, когда все значения переменной совпадают, эта переменная не несет никакой информации. Напротив, если значения переменной $\tilde{x}_i^{\alpha}$ равномерно распределены в единичном интервале, информация такой переменной максимальна.

Общий принцип предобработки данных для обучения, таким образом, состоит в максимизации энтропии входов и выходов. Этим принципом следует руководствоваться и на этапе кодирования нечисловых переменных.

Предобработка данных

Максимизация энтропии как цель предобработки

Типы нечисловых переменных

Можно выделить два основных типа нечисловых переменных: *упорядоченные* (называемые также *ординальными* - от англ. *order* - порядок) и *категориальные*. В обоих случаях переменная относится к одному из дискретного набора классов $\{c_1, \ldots, c_n\}$. Но в первом случае эти классы упорядочены - их можно *ранжировать*: $c_1 \succ c_2 \succ \ldots \succ c_n$, тогда как во втором такая упорядоченность отсутствует. В качестве примера упорядоченных переменных можно привести сравнительные категории: плохо - хорошо - отлично, или медленно - быстро. Категориальные переменные просто обозначают один из классов, являются *именами* категорий. Например, это могут быть имена людей или названия цветов: белый, синий, красный.

Алгоритм кодирования ординальных переменных.

Ординальные переменные более близки к числовой форме, т.к. числовой ряд также упорядочен. Соответственно, для кодирования таких переменных остается лишь поставить в соответствие номерам категорий такие числовые значения, которые сохраняли бы существующую упорядоченность. Естественно, при этом имеется большая свобода выбора - любая монотонная функция от номера класса порождает свой способ кодирования. Какая же из бесконечного многообразия монотонных функций - наилучшая?

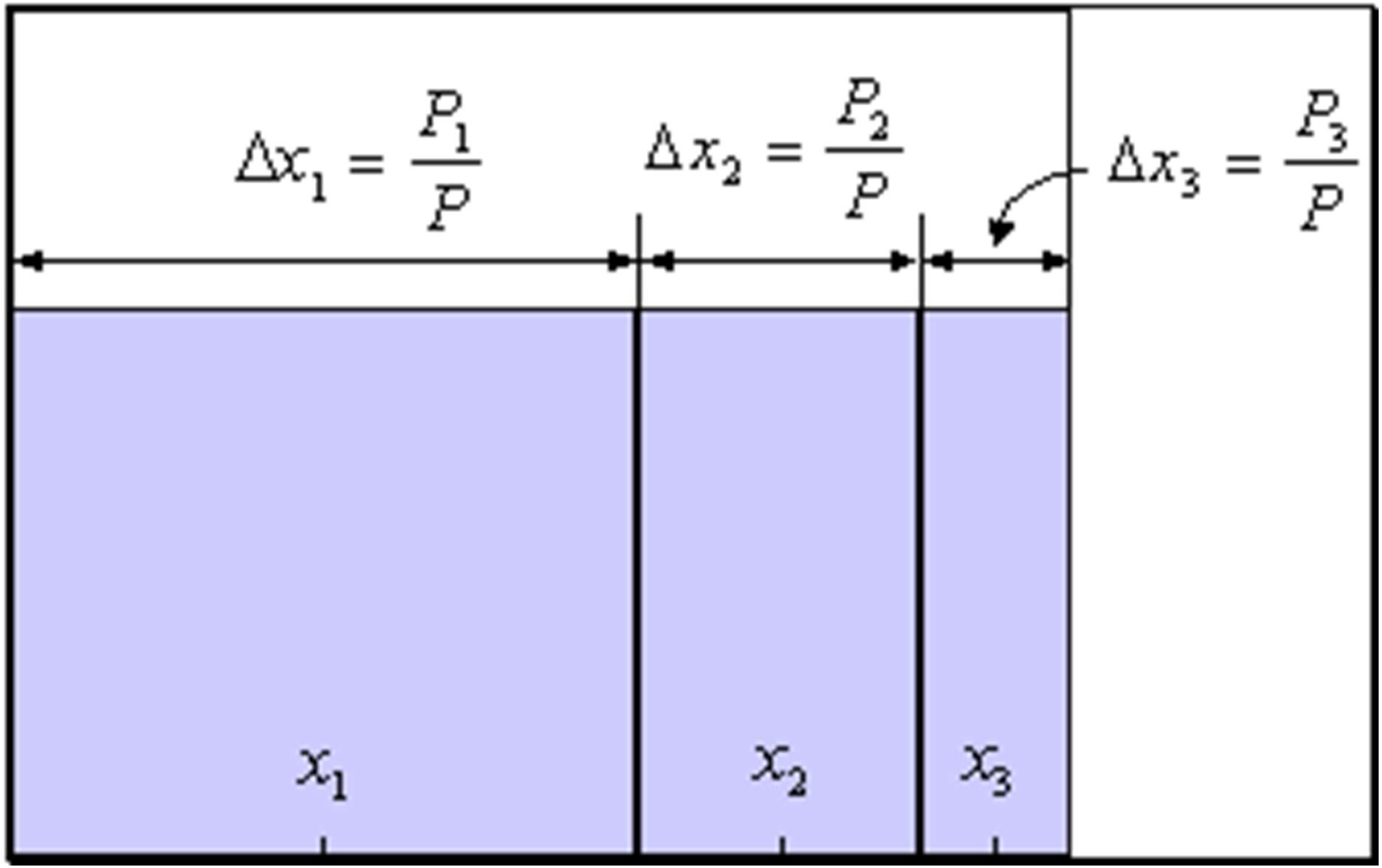
В соответствии с изложенным выше общим принципом, мы должны стремиться к тому, чтобы максимизировать энтропию закодированных данных. При использовании сигмоидных функций активации все выходные значения лежат в конечном интервале - обычно $[0,1]$ или $[-1,1]$. Из всех статистических функций распределения, определенных на конечном интервале, максимальной энтропией обладает равномерное распределение.

Применительно к данному случаю это подразумевает, что кодирование переменных числовыми значениями должно приводить, по возможности, к равномерному заполнению единичного интервала закодированными примерами. (Захватывая заодно и этап нормировки.) При таком способе "оцифровки" все примеры будут нести примерно одинаковую информационную нагрузку.

Алгоритм кодирования ординальных переменных.

- Единичный отрезок разбивается на *n* отрезков (по числу классов) с длинами, пропорциональными числу примеров каждого класса в обучающей выборке: $\Delta x = {P_k}/{P}$,
- где P_k - число примеров класса *k*, а *P* -общее число примеров.
- Центр каждого такого отрезка будет являться численным значением для соответствующего ординального класса

Алгоритм кодирования ординальных переменных



Кодирование категориальных переменных

- Категориальные переменные также можно закодировать как ординальные, пронумеровав их произвольным образом.
- Однако, такое навязывание несуществующей упорядоченности только затруднит решение задачи. Оптимальное кодирование не должно искажать структуры соотношений между классами. Если классы не упорядочены, такова же должна быть и схема кодирования.
- Наиболее естественной выглядит и чаще всего используется на практике двоичное кодирование, когда имена n категорий кодируются значениями n бинарных нейронов, причем

первая категория кодируется как (1,0, …,0),

вторая категория кодируется как (0,1,…,0) и т.д. вплоть до

n- ной: (0,…,0,1).

Можно использовать биполярную кодировку, в которой нули заменяются на -1.

Легко убедиться, что в такой симметричной кодировке расстояния между всеми векторами-категориями равны.

Нормировка и предобработка данных

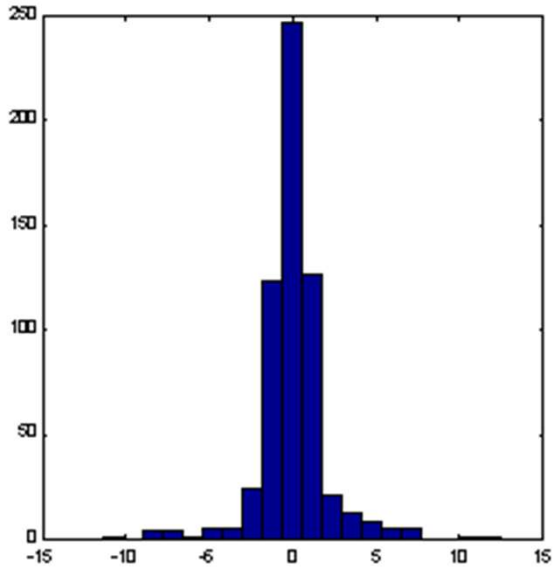
- Как входами, так и выходами нейросети могут быть совершенно разнородные величины. Очевидно, что результаты нейросетевого моделирования не должны зависеть от единиц измерения этих величин.
- А именно, чтобы сеть трактовала их значения единообразно, все входные и выходные величины должны быть приведены к единому - единичному - масштабу.
- Кроме того, для повышения скорости и качества обучения полезно провести дополнительную предобработку данных, выравнивающую распределение значений еще до этапа обучения.

Нормировка и предобработка данных

- Приведение данных к единичному масштабу обеспечивается нормировкой каждой переменной на диапазон разброса ее значений.
- В простейшем варианте это - линейное преобразование:

$$\tilde{x}_i = \frac{x_i - x_{i,\min}}{x_{i,\max} - x_{i,\min}}$$

Линейная нормировка **оптимальна, когда значения переменной плотно заполняют определен**ный интервал. Но подобный "прямолинейный" подход применим далеко не всегда. Так, если в данных имеются относительно редкие выбросы, намного превышающие типичный разброс, именно эти выбросы определят согласно предыдущей формуле масштаб нормировки. Это приведет к тому, что основная масса значений нормированной переменной сосредоточится вблизи нуля



Нормировка и предобработка данных

- Нормировка на основе статистических характеристик данных, такие как среднее и дисперсия:

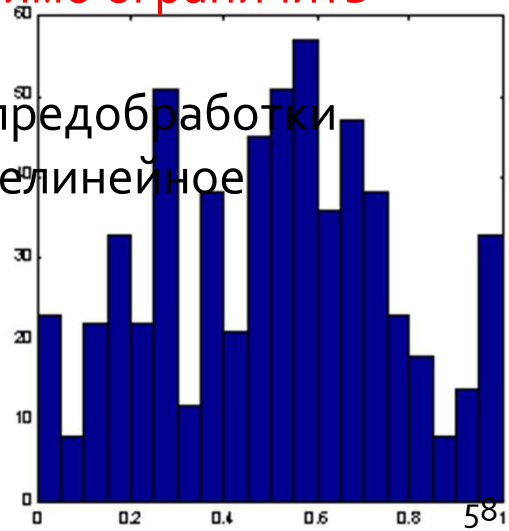
$$\tilde{x}_i = \frac{x_i - \bar{x}_i}{\sigma_i}, \qquad \bar{x}_i \equiv \frac{1}{P}\sum_{\alpha=1}^{P} x_i^{\alpha}, \quad \sigma_i^2 \equiv \frac{1}{P-1}\sum_{\alpha=1}^{P}\left(x_i^{\alpha} - \bar{x}_i\right)^2$$

Однако, теперь нормированные величины не принадлежат гарантированно единичному интервалу, более того, максимальный разброс значений переменной заранее не известен. Для входных данных это может быть и не важно, но выходные переменные будут использоваться в качестве эталонов для выходных нейронов.

В случае, если выходные нейроны - сигмоидные, они могут принимать значения лишь в единичном диапазоне. Чтобы установить соответствие между обучающей выборкой и нейросетью в этом случае необходимо ограничить диапазон изменения переменных.

Естественный выход из этой ситуации - использовать для предобработки данных функцию активации тех же нейронов. Например, нелинейное преобразование

$$\tilde{x}_i = f\left(\frac{x_i - \bar{x}_i}{\sigma_i}\right), \qquad f(a) = \frac{1}{1+e^{-a}}$$



Обучение без учителя
Алгоритмы обучения Хебба

- Процесс обучения, как и в случае обучения с учителем, заключается в подстраивании весов синапсов.
- Подстройка синапсов может проводиться только на основании информации, доступной в нейроне, то есть его состояния и уже имеющихся весовых коэффициентов.

Обучение без учителя
Алгоритмы обучения Хебба

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

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

где

- $y_i^{(n-1)}$ – выходное 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.

Обучение без учителя
Алгоритмы обучения Хебба

Дифференциальный метод обучения Хебба

$$w_{ij}(t) = w_{ij}(t-1) + \alpha \cdot \left[y_i^{(n-1)}(t) - y_i^{(n-1)}(t-1) \right] \cdot \left[y_j^{(n)}(t) - y_j^{(n)}(t-1) \right] \quad (2)$$

- $y_i^{(n-1)}(t)$ и $y_i^{(n-1)}(t-1)$ – 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.
- сильнее всего обучаются синапсы, соединяющие те нейроны, выходы которых наиболее динамично изменились в сторону увеличения

Обучение без учителя
Алгоритмы обучения Хебба

- 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 прямопоточных (feedforward) сетей, то есть для каждого нейрона рассчитывается взвешенная сумма его входов, к которой затем применяется активационная (передаточная) функция нейрона, в результате чего получается его выходное значение $y_i^{(n)}$, i=0…M_i-1, где M_i – число нейронов в слое i; n=0…N-1, а N – число слоев в сети.
- 3. На основании полученных выходных значений нейронов по формуле (1) или (2) производится изменение весовых коэффициентов.
- 4. Цикл с шага 2, пока выходные значения сети не застабилизируются с заданной точностью.
- Применение этого нового способа определения завершения обучения, отличного от использовавшегося для сети обратного распространения, обусловлено тем, что подстраиваемые значения синапсов фактически не ограничены.
- На втором шаге цикла попеременно предъявляются все образы из входного набора.

Нейронная сеть Кохонена

Самоорганизующиеся карты

(Self Organizing Maps – SOM), разработанные

Т. Кохоненом [Kohonen, 1982], представляют собой мощный аналитический инструмент, объединяющий в себе две основные парадигмы анализа –

- кластеризациию
- проецирование, т.е. визуализацию многомерных данных на плоскости.

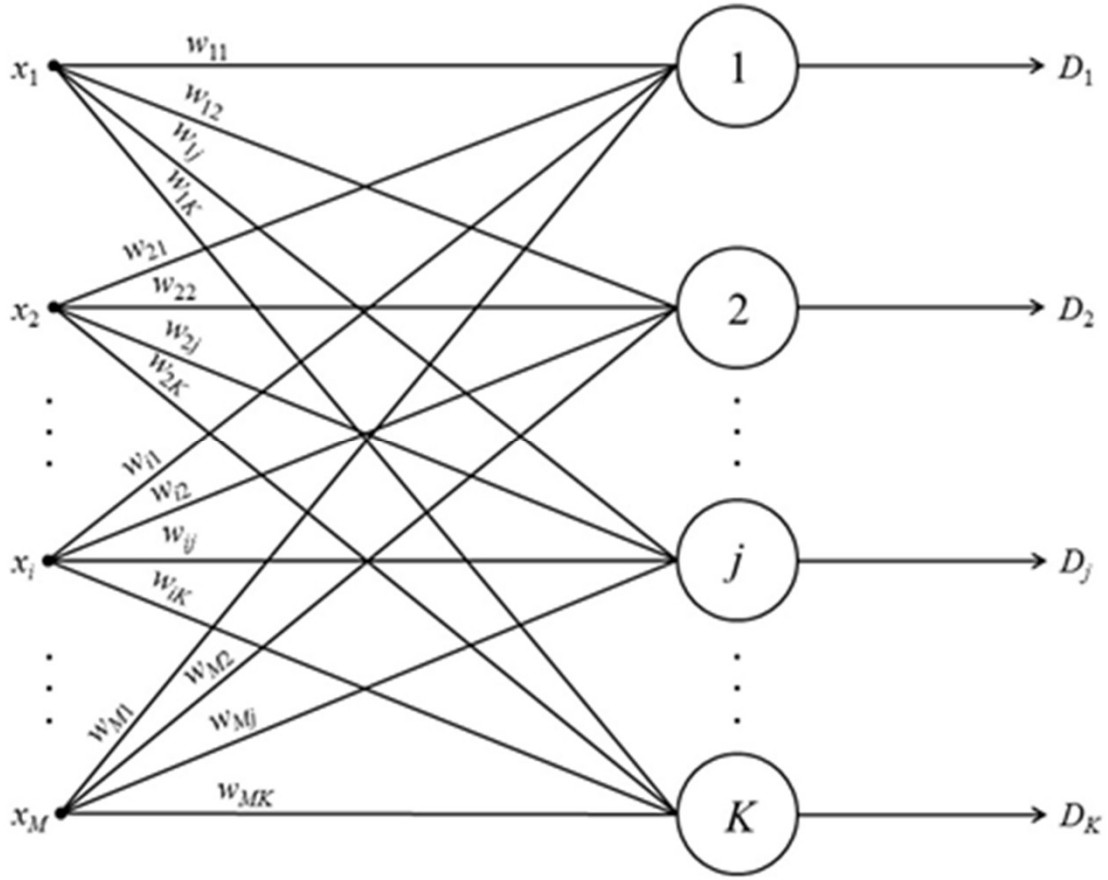
Сеть Кохонена распознает кластеры в многомерных обучающих данных и относит все данные к тем или иным кластерам, используя алгоритм проецирования с сохранением топологического подобия.

При этом те элементы выборки, которые находятся в относительной близости в исходном многомерном пространстве, оказываются рядом и в пространстве с более низкой размерностью.

Нейронная сеть Кохонена Архитектура

Сеть Кохонена имеет всего два слоя: ***входной*** и ***выходной***, составленный из радиальных нейронов упорядоченной структуры (выходной слой называют также ***слоем топологической карты***).

K – количество классов



Нейронная сеть Кохонена

- Процесс обучения, как и в случае обучения с учителем, заключается в подстраивании весов синапсов методом последовательных приближений на основании их значений от предыдущей итерации.
- Обучение сводится к минимизации разницы между входными сигналами нейрона и весовыми коэффициентами его синапсов.
- Корректируются только веса нейрона-победителя (нейронов, лежащих в некоторой окрестности нейрона-победителя)

Нейронная сеть Кохонена
Алгоритм обучения

- На стадии инициализации всем весовым коэффициентам присваиваются небольшие случайные значения.
- На входы сети подается входной образ
- Из всего выходного слоя выбирается нейрон – **нейрон-победитель**, значения синапсов которого максимально походят на входной образ
- Для него осуществляется подстройка весов синапсов с применением формулы

$$w_{ij}(t) = w_{ij}(t-1) + \alpha(x_i - w_{ij}(t-1))$$

  - Эта, так называемая, аккредитация может сопровождаться "затормаживанием" всех остальных нейронов слоя и введением выбранного нейрона в насыщение. Иными словами, в ходе обучения модифицируется не только нейрон -"победитель", но, в меньшей степени, и его соседи из некоторой окрестности.
- Цикл повторяется с шага 2, где попеременно предъявляются все образы из входного набора пока выходные значения сети не будут стабилизированы с заданной точностью.

Нейронная сеть Кохонена
Выбор нейрона-победителя

- Расчет скалярного произведения вектора весовых коэффициентов с вектором входных значений (уровень входного возбуждения). Максимальное произведение дает выигравший нейрон.

$$S_j = \sum_{i=1}^{m} x_i \, w_{ij} = \|x\| \cdot \left\|w^j\right\| cos\varphi$$

- Расчет расстояния между этими векторами в m-мерном пространстве, где m – размер векторов. В данном случае, "побеждает" нейрон с наименьшим расстоянием.

$$D_j = \sqrt{\sum_{i=1}^{m} (x_i - w_{ij})^2}$$

где

j – индекс нейрона, i – индекс входа, w_{ij} – вес синапса

Нейронная сеть Кохонена
Нормализация входных значений

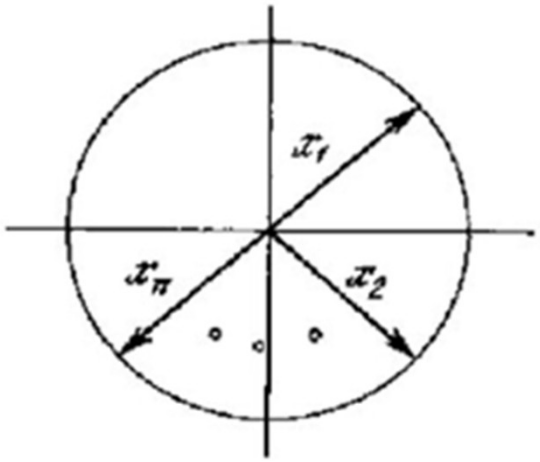
При использовании обучения по алгоритму Кохонена на стадии инициализации существует практика нормализации:

- входных образов

$$x_j = x_j / \sqrt{\sum_{l=1}^{m} x_l^2}$$

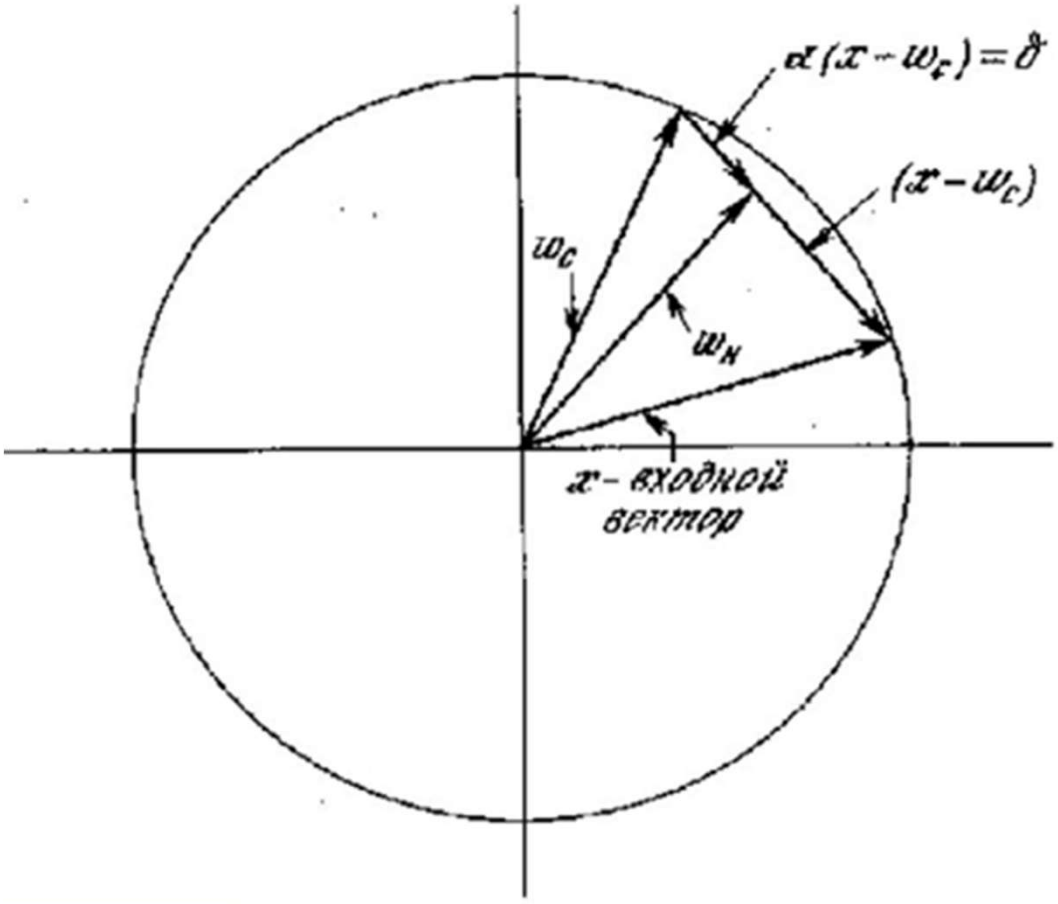
- начальных значений весовых коэффициентов

$$w_{ij} = w_{ij} / \sqrt{\sum_{l=1}^{k} w_{lj}^2}$$



Нейронная сеть Кохонена
Смысл обучения

$$w_H = w_C + \alpha[x - w_C]$$



Закрепить за каждым классом объектов один нейрон

Нейронная сеть Кохонена

- *Замечание 1* 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.
- *Замечание 2* Размер окрестности и скорость обучения – суть функции, значения которых уменьшаются со временем

После выбора нейрона j с минимальным расстоянием D_j обучается не только этот нейрон, но и его соседи, расположенные в окрестности R.

Величина R на первых итерациях очень большая, так что обучаются все нейроны, но с течением времени она уменьшается до нуля.

Таким образом, чем ближе конец обучения, тем точнее определяется группа нейронов, отвечающих каждому классу образов.

Нейронная сеть Кохонена

Метод выпуклой комбинации

Инициализация весовых коэффициентов случайными значениями может привести к тому, что

1) различные классы, которым соответствуют плотно распределенные входные образы, сольются или, наоборот,
2) раздробятся на дополнительные подклассы в случае близких образов одного и того же класса.

- Суть метода выпуклой комбинации сводится к тому, что входные нормализованные образы подвергаются преобразованию:

$$x_i = \alpha(t) \cdot x_i + (1 - \alpha(t)) \cdot \frac{1}{\sqrt{m}}$$

- где x_i – i-ая компонента входного образа, m – общее число его компонент, $\alpha(t)$ – коэффициент, изменяющийся в процессе обучения от **нуля** до **единицы**, в результате чего вначале на входы сети подаются **практически одинаковые образы**, а с течением времени они все больше **сходятся к исходным**.
- Весовые коэффициенты устанавливаются на шаге инициализации равными величине

$$w_o = \frac{1}{m}$$

- где – размерность вектора весов для нейронов инициализируемого слоя.

Нейронная сеть Кохонена

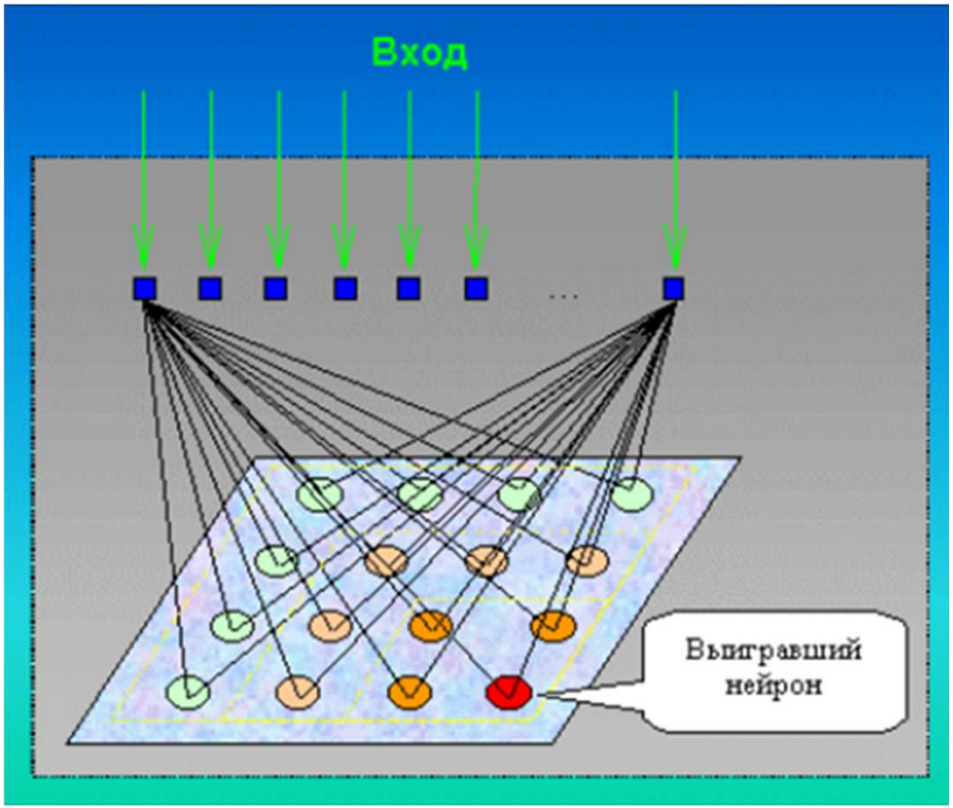
На основе метода обучения Кохонена строятся нейронные сети особого типа – так называемые самоорганизующиеся структуры – self-organizing feature maps

- Уникальность метода *самоорганизующихся карт* состоит в преобразовании n-мерного пространства в двухмерное. Применение двухмерных сеток связано с тем, что существует проблема отображения пространственных структур большей размерности.
- Имея такое представление данных, можно визуально определить наличие или отсутствие взаимосвязи во входных данных.
- Нейроны карты Кохонена располагают в виде двухмерной матрицы, раскрашивают эту матрицу в зависимости от анализируемых параметров нейронов.

Нейронная сеть Кохонена

Качество проецирования многомерных данных на плоскость достигается в SOM на нескольких уровнях: сохранение топологии (т.е. на множествах точек исходных данных и нейронов обученной сети структура соседства одинакова), сохранение порядка (т.е. расстояния между эквивалентными парами точек пропорциональны) и сохранение метрических свойств при сжатии пространства.

"Проекционный экран" в результате обучения приобретает свойства упорядоченной структуры, в которой величины синапсов нейронов плавно меняются вдоль двух измерений.
Цвет и расположение фрагментов двумерной решетки используется для анализа закономерностей, связываемых с компонентами набора данных.
В частности, с каждым узлом (нейроном) могут ассоциироваться локальные сгущения исходных объектов, которые могут служить потенциальными центрами кластеров.



Нейронная сеть Кохонена Применение

- ***Самоорганизующиеся карты*** могут использоваться для решения таких задач, как моделирование, прогнозирование, поиск закономерностей в больших массивах данных, выявление наборов независимых признаков и сжатие информации.
- Наиболее распространенное применение сетей Кохонена - решение задачи классификации без учителя, т.е. **кластеризации**.
- Напомним, что при такой постановке задачи нам дан набор объектов, каждому из которых сопоставлена строка таблицы (вектор значений признаков). Требуется разбить исходное множество на классы, т.е. для каждого объекта найти класс, к которому он принадлежит.
- **Разведочный анализ данных**. Сеть Кохонена способна распознавать кластеры в данных, а также устанавливать близость классов. Таким образом, пользователь может улучшить свое понимание структуры данных, чтобы затем уточнить нейросетевую модель.

Нейронная сеть Кохонена Применение

- Если в данных распознаны классы, то их можно обозначить, после чего сеть сможет решать **задачи классификации.**
- Сети Кохонена можно использовать и в тех задачах классификации, где классы уже заданы, - тогда преимущество будет в том, что сеть сможет выявить сходство между различными классами.
- **Обнаружение новых явлений**. Сеть Кохонена распознает кластеры в обучающих данных и относит все данные к тем или иным кластерам. Если после этого сеть встретится с набором данных, непохожим ни на один из известных образцов, то она не сможет классифицировать такой набор и тем самым выявит его новизну.

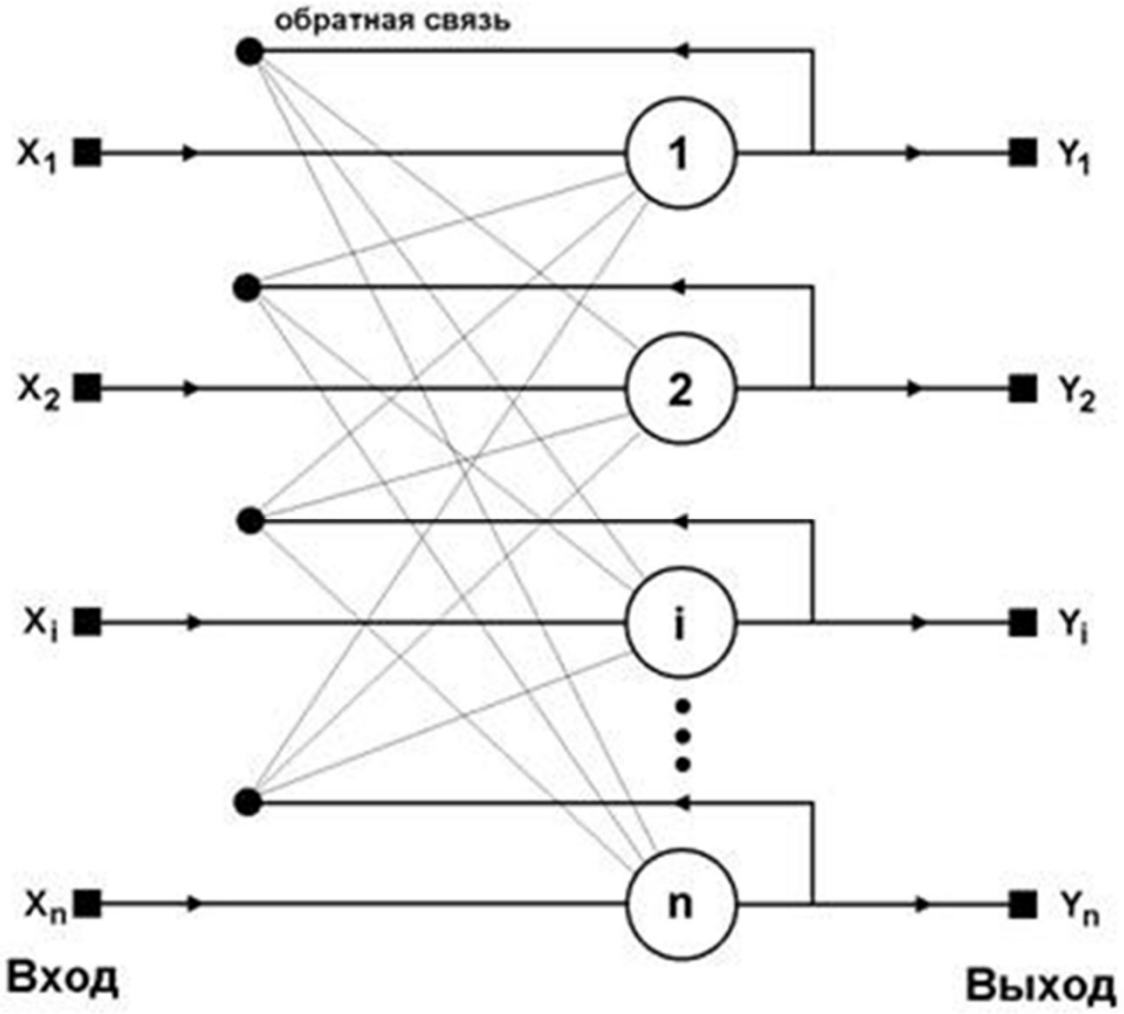
Нейронные сети Хопфилда и Хэмминга

- В сетях данного вида весовые коэффициенты синапсов рассчитываются только однажды перед началом функционирования сети на основе информации об обрабатываемых данных, и все обучение сети сводится именно к этому расчету.
- С одной стороны, предъявление априорной информации можно расценивать, как помощь учителя, но с другой – сеть фактически просто запоминает образцы до того, как на ее вход поступают реальные данные, и не может изменять свое поведение, поэтому говорить о звене обратной связи с "миром" (учителем) не приходится.
- Из сетей с подобной логикой работы наиболее известны сеть Хопфилда и сеть Хэмминга, которые обычно используются для организации ассоциативной памяти.

Нейронные сети Хопфилда и Хэмминга
Задача «ассоциативной памяти»

- Известен некоторый набор двоичных сигналов (изображений, звуковых оцифровок, прочих данных, описывающих некие объекты или характеристики процессов), которые считаются образцовыми.
- Сеть должна уметь из произвольного неидеального сигнала, поданного на ее вход, выделить ("вспомнить" по частичной информации) соответствующий образец (если такой есть) или "дать заключение" о том, что входные данные не соответствуют ни одному из образцов.
- Если, например, сигналы представляют собой некие изображения, то, отобразив в графическом виде данные с выхода сети, можно будет увидеть картинку, полностью совпадающую с одной из образцовых (в случае успеха) или же "вольную импровизацию" сети (в случае неудачи).

Нейронная сеть Хопфилда
Архитектура



Нейронная сеть Хопфилда

- В общем случае, любой сигнал может быть описан вектором

X = (x_1, … x_n), где x_i равен либо +1, либо -1, n – число нейронов в сети и размерность входных и выходных векторов.

- Обозначим вектор, описывающий k-ый образец, через $\mathbf{X}^k$, а его компоненты, соответственно, – x_i^k, k=1…m, m – число образцов.
- Когда сеть распознает (или "вспомнит") какой-либо образец на основе предъявленных ей данных, ее выходы будут содержать именно его, то есть **Y** = $\mathbf{X}^k$, где **Y** – вектор выходных значений сети: **Y** = (y_1, …, y_n).
- В противном случае, выходной вектор не совпадет ни с одним образцовым.

Нейронная сеть Хопфилда
Инициализация

- На стадии инициализации сети весовые коэффициенты синапсов устанавливаются следующим образом:

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, i \neq j \\ 0, \quad i = j \end{cases}$$

- Здесь i и j – индексы, соответственно, предсинаптического и постсинаптического нейронов;
- x_i^k, x_j^k – i-ый и j-ый элементы вектора k-ого образца.

Нейронная сеть Хопфилда

Функционирование

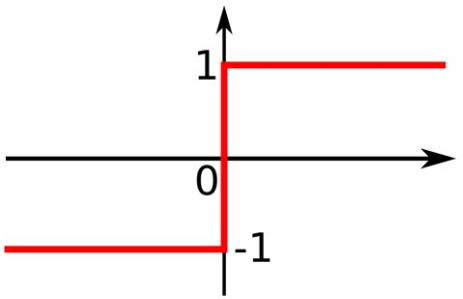
1. На входы сети подается неизвестный сигнал. Фактически его ввод осуществляется непосредственной установкой значений аксонов:

yi(0) = xi , i = 0…n-1,

поэтому обозначение на схеме сети входных синапсов в явном виде носит чисто условный характер. Ноль в скобке справа от yi означает нулевую итерацию в цикле работы сети.

2. Рассчитывается новое состояние нейронов

$$s_j(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p)$$, j=0…n-1



и новые значения аксонов

$$y_j(p+1) = f\left[s_j(p+1)\right]$$

где f – активационная функция в виде скачка.

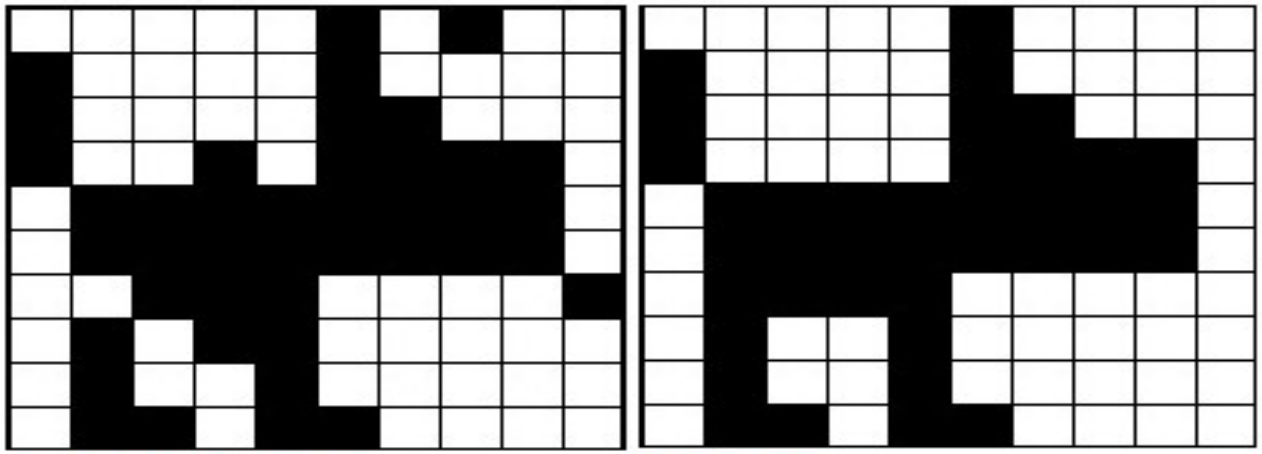
3. Проверка, изменились ли выходные значения аксонов за последнюю итерацию. Если да – переход к пункту 2, иначе (если выходы застабилизировались) – конец. При этом выходной вектор представляет собой образец, наилучшим образом сочетающийся с входными данными.

Нейронная сеть Хопфилда
Функционирование

- Иногда сеть не может провести распознавание и выдает на выходе несуществующий образ.
- Это связано с проблемой ограниченности возможностей сети. Для сети Хопфилда число запоминаемых образов m не должно превышать величины, примерно равной $0.15n$.
- Кроме того, если два образа А и Б сильно похожи, они, возможно, будут вызывать у сети перекрестные ассоциации, то есть предъявление на входы сети вектора А приведет к появлению на ее выходах вектора Б и наоборот.
- Сеть может завершить или исправить образ, но не может ассоциировать полученный образ с другим образом

Нейронная сеть Хопфилда
Функционирование

- Пусть имеется нейронная сеть размерностью N=100, в матрицу связей записан набор чёрно-белых картинок (–1 — чёрный цвет, +1 — белый), среди которых есть изображение собачки.
- Если установить начальное состояние сети близким к этому вектору (рисунок слева, искажённый образ), то в ходе динамики нейронная сеть восстановит исходное изображение (эталон).

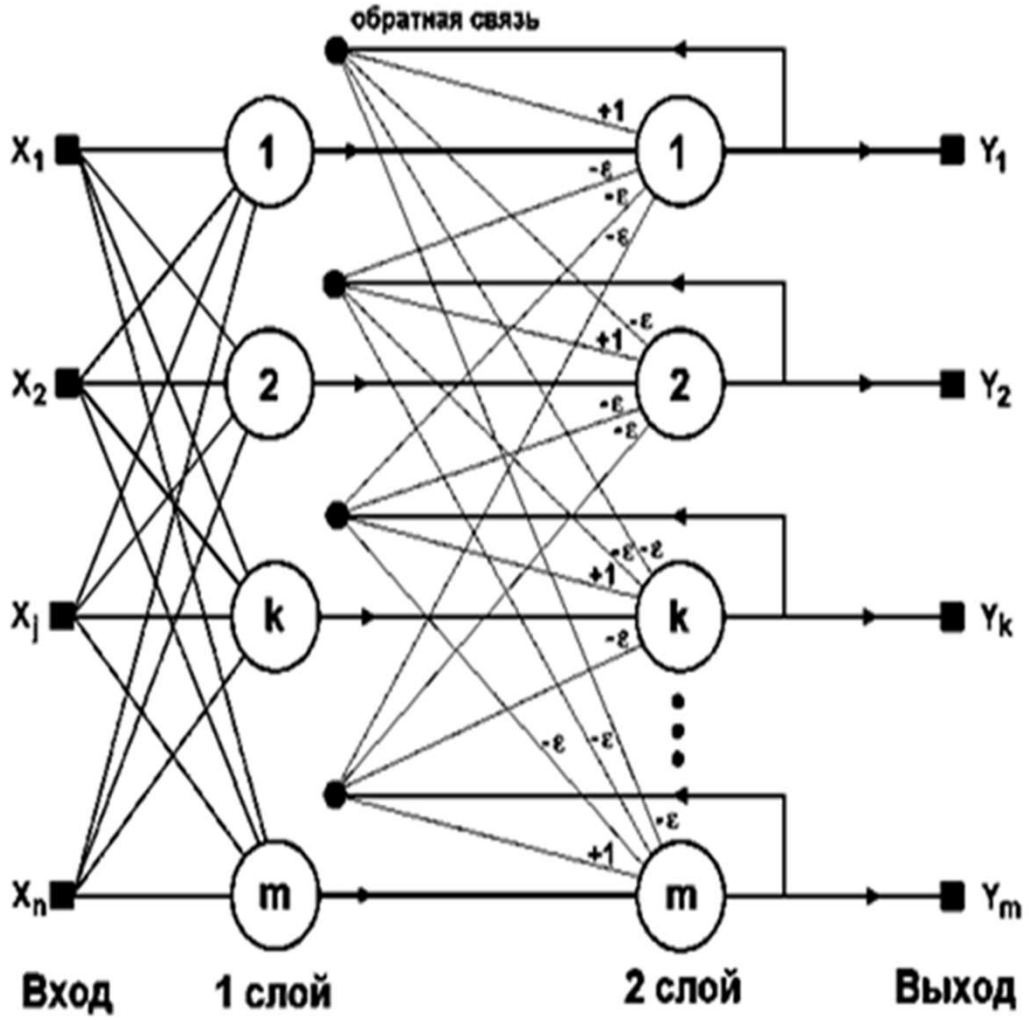


Искажённый образ Эталон

Нейронная сеть Хэмминга

- Когда нет необходимости, чтобы сеть в явном виде выдавала образец, то есть достаточно, скажем, получать номер образца, ассоциативную память успешно реализует сеть Хэмминга. Данная сеть характеризуется, по сравнению с сетью Хопфилда, меньшими затратами на память и объемом вычислений
- Идея работы сети состоит в нахождении расстояния Хэмминга от тестируемого образа до всех образцов.
- Расстоянием Хэмминга называется число отличающихся битов в двух бинарных векторах.
- Сеть должна выбрать образец с минимальным расстоянием Хэмминга до неизвестного входного сигнала, в результате чего будет активизирован только один выход сети, соответствующий этому образцу.

Нейронная сеть Хэмминга
Архитектура



Сеть состоит из двух слоев. Первый и второй слои имеют по m нейронов, где m – число образцов.

Нейроны первого слоя имеют по n синапсов, соединенных со входами сети (образующими фиктивный нулевой слой).

Нейроны второго слоя связаны между собой ингибиторными (отрицательными обратными) синаптическими связями.

Единственный синапс с положительной обратной связью для каждого нейрона соединен с его же аксоном.

Нейронная сеть Хэмминга
Алгоритм функционирования

- На стадии инициализации весовым коэффициентам первого слоя и порогу активационной функции присваиваются следующие значения:

$$w_{ik} = \frac{x_i^k}{2}, \quad i=0\ldots n-1,\ k=0\ldots m-1$$

$$T_k = n / 2,\ k = 0\ldots m-1$$

- Здесь x_i^k – i-ый элемент k-ого образца.
- Весовые коэффициенты тормозящих синапсов во втором слое берут равными некоторой величине $0 < \varepsilon < 1/m$. Синапс нейрона, связанный с его же аксоном имеет вес +1.

Нейронная сеть Хэмминга

Алгоритм функционирования

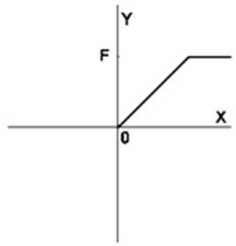
- 1. На входы сети подается неизвестный вектор **X** = {x_i:i=0…n-1}, исходя из которого рассчитываются состояния нейронов первого слоя (верхний индекс в скобках указывает номер слоя):

$$y_j^{(1)} = s_j^{(1)} = \sum_{i=0}^{n-1} w_{ij} x_i + T_j \qquad , j=0\ldots m-1$$

- После этого полученными значениями инициализируются значения аксонов второго слоя:

$$y_j^{(2)} = y_j^{(1)}, j = 0\ldots m-1$$

- 2. Вычислить новые состояния нейронов второго слоя:



$$s_j^{(2)}(p+1) = y_j(p) - \varepsilon \sum_{k=0}^{m-1} y_k^{(2)}(p), k \neq j, j = 0\ldots m-1$$

и значения их аксонов:

$$y_j^{(2)}(p+1) = f\left[s_j^{(2)}(p+1)\right], j = 0\ldots m-1$$

- Активационная функция f имеет вид порога, причем величина F должна быть достаточно большой, чтобы любые возможные значения аргумента не приводили к насыщению.
- 3. Проверить, изменились ли выходы нейронов второго слоя за последнюю итерацию. Если да – перейди к шагу 2. Иначе – конец.

Нейронная сеть Хэмминга

- *Замечание* Роль первого слоя весьма условна: воспользовавшись один раз на шаге 1 значениями его весовых коэффициентов, сеть больше не обращается к нему, поэтому первый слой может быть вообще исключен из сети (заменен на матрицу весовых коэффициентов).