Искусственные и нечеткие нейронные сети

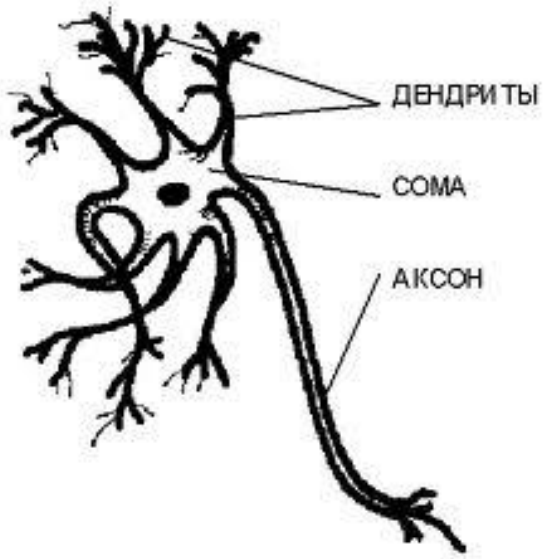
ИНС

- **Иску́сственные нейро́нные се́ти** представляют собой математические модели, а также их программные или аппаратные реализации, построенные по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма.

Применение

- аппроксимация функций на основе ряда данных;
- распознавание образов;
- кластеризация и классификация данных;
- обучение в области статистической обработки данных;
- накопление знаний через обучение на примерах;
- предсказание и прогноз;
- оптимизация;
- ассоциативная память;
- нелинейное моделирование и управление.

Биологический нейрон

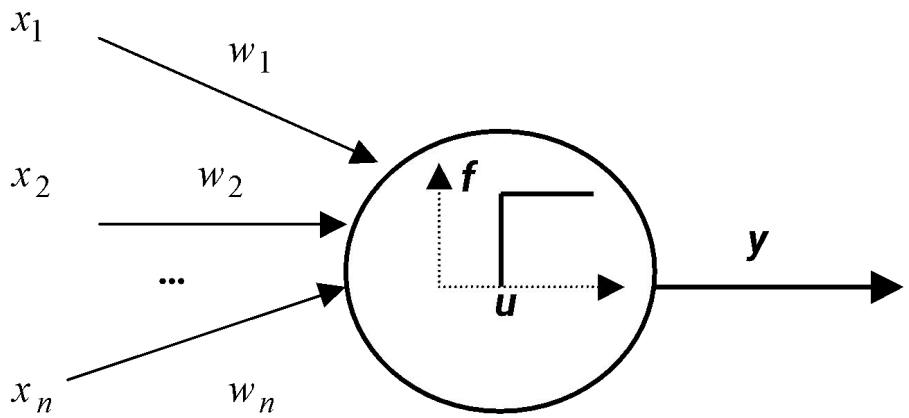


Отростки, проводящие информацию (*стимул*) в тело, называются «*дендритами*» (*dendrites*). Отростки, проводящие информацию (*реацкия*) из тела, называются «*аксонами*» (*axon*).

- Нейрон обладает некоторым *потенциалом активации* (*activaon potential*). Сигнал (называемый a *spike*) передается к другим нейронам посредством аксона и характеризуется частотой, длительностью и амплитудой. Взаимодейтсвие между нейронами реализуется в строго определенных точках, называемых «синапсами» (*synapses*).

Модель искусственного нейрона

- Математическая модель нейрона (Рис.4-2) впервые была предложена Маккалоком и Питтсом (Warren McCulloch and Walter Pitts) в 1943.



$$y = f\left(\sum_{j=1}^{n} w_j x_j - u\right),$$

Итак, иску́сственный нейро́н (или м*атематический нейрон* Маккалока-Питтса, или ф*ормальный нейрон*) — это узел искусственной нейронной сети, являющийся упрощённой моделью естественного нейрона. Математически, искусственный нейрон обычно представляют собой некоторую нелинейную функцию от единственного аргумента — линейной комбинации взвешенных входных сигналов. Полученный результат посылается на единственный выход.

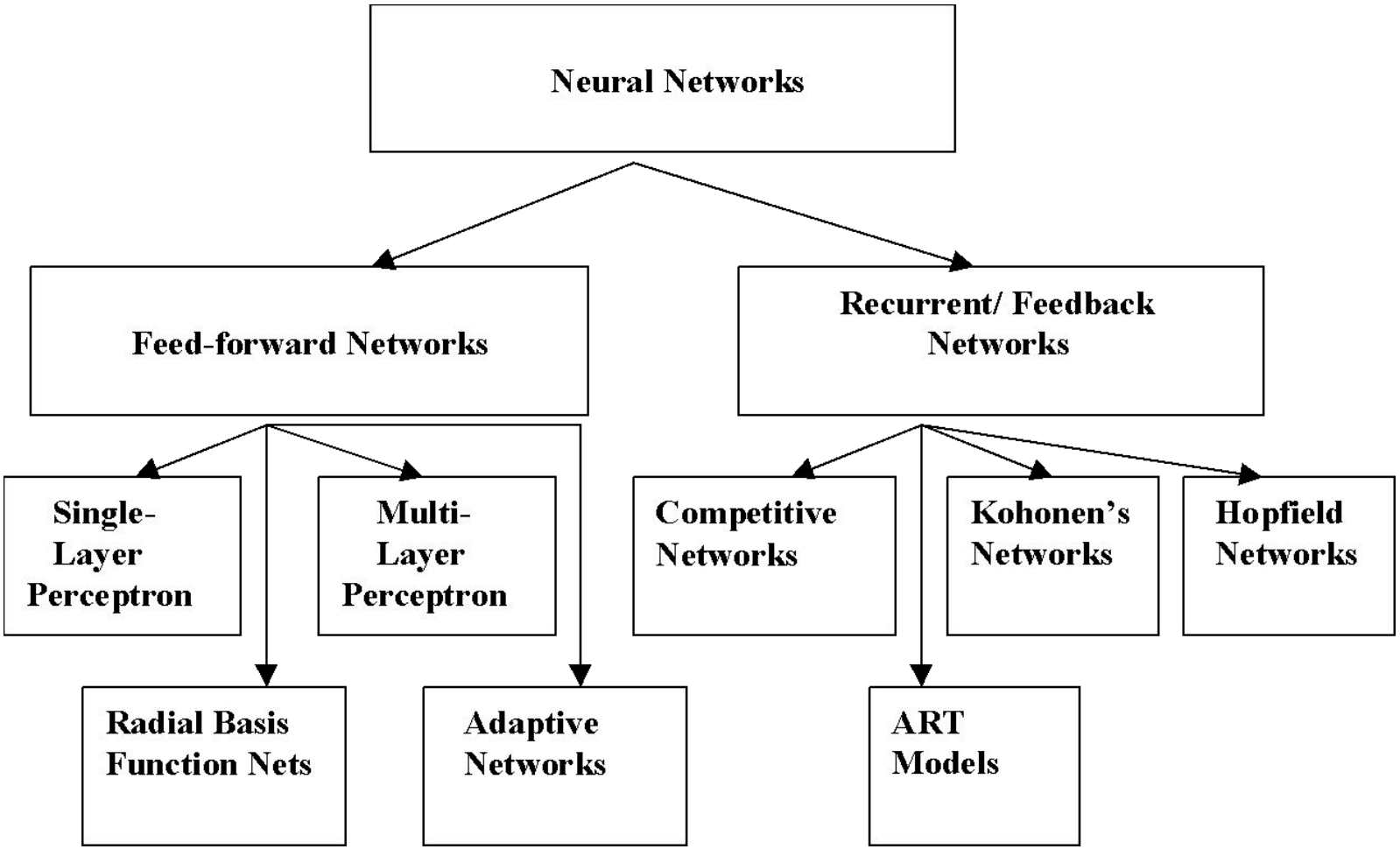
От нейрона к системе нейронов

- ИНС может рассматриваться в виде взвешенного направленного графа (*weighted directed graph*), вершины которого представлены нейронами, а направленные дуги (с весами) описывают связи между нейронами. ИНС характеризуется тремя параметрами:
  - типом нейронов,
  - архитектурой (организацией связей между нейронами)
  - алгоритмом обучения в данной сети.

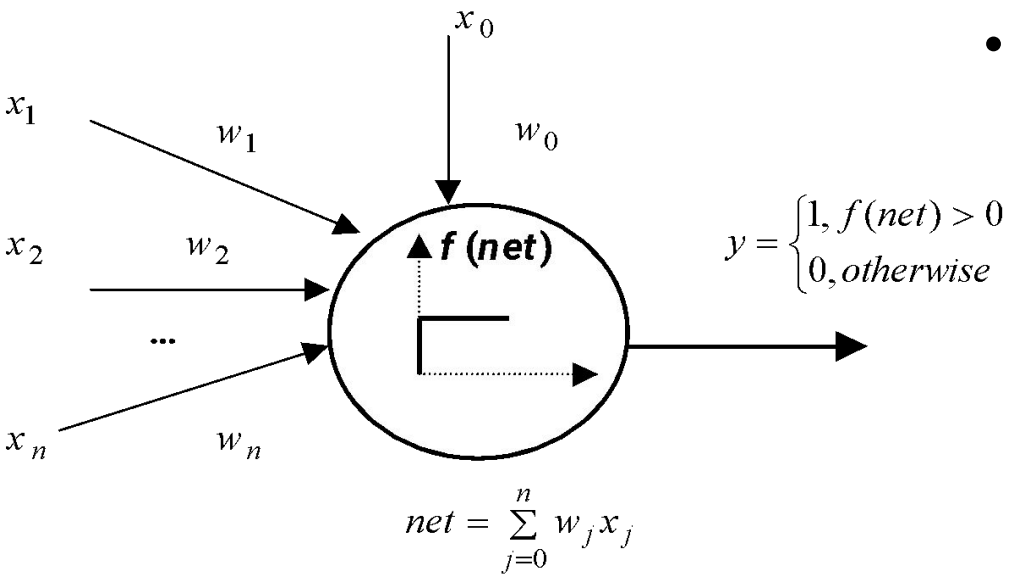
Архитектура

- С учетом архитектуры ИНС могут быть разделены на два базисных класса:
  - сети с прямым распространением сигнала (Feed-forward network), в структуре которых нет петель (циклов) и
  - сети с обратным распространением сигнала или рекуррентные сети (Feedback network or Recurrent), в структуре которых есть циклы благодаря наличию обратных связей.

Типы ИНС


Neural Networks
Feed-forward Networks
Recurrent/ Feedback Networks
Single-Layer Perceptron
Multi-Layer Perceptron
Radial Basis Function Nets
Adaptive Networks
Competitive Networks
Kohonen's Networks
Hopfield Networks
ART Models

Простой «Перцептрон»



- *Перцептро́н* (или *Персептрон*) — [математическая](#) и [компьютерная модель](#), предложенная [Фрэнком Розенблаттом](#) в [1957 году](#) для задачи моделирования [восприятия](#) [информации мозгом](#), и реализованная в виде электронной машины [«Марк-1»](#) в [1960 году](#).

Имея на входе вектор $x = (x_1, x_2, ..., x_n)^T$, перцептрон вычисляет выходное значение следубшим образом:

$$y = f(net), \text{ где } net = (w_0 x_0 + \sum_{j=1}^{n} w_j x_j) = \sum_{j=0}^{n} w_j x_j \text{ и } w_0 = -u, x_0 = 1,$$

где f есть единичная пороговая функция.

Таким образом, выход перцептрона $y = +1$, если $f(net) > 0$ и $y = 0$ в противном случае.

Задача кластеризации

Перцептрон может решать задачу кластеризации. Под кластеризацией понимается разбиение множества входных сигналов на классы, при том, что ни количество, ни признаки классов заранее не известны. После обучения такая сеть способна определять, к какому классу относится входной сигнал.

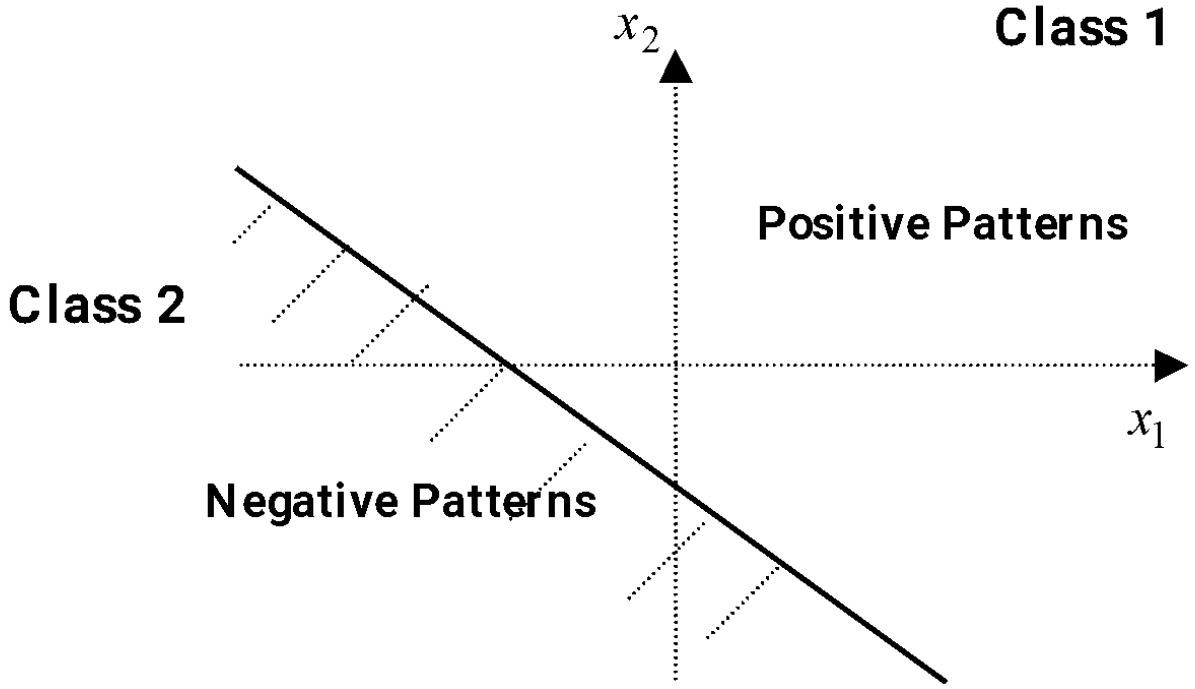
Кроме того, персептрон может распознать принадлежит ли входной вектор значений к данному классу или нет. Например, входной вектор значений $(x_1, x_2, ..., x_n)$ (назовем его также как *входной образ*) может описывать свойства некоторого визуального объекта. Выход перцептрона может соответствовать распознаванию данного объекта (его наличию или отсутствию).

В задаче кластеризации перцептрон приписывает входной образ к классу 1, если $\sum_{j=1}^{n} w_j x_j > u$ и к классу 2, если $\sum_{j=1}^{n} w_j x_j < u$. Линейное уравнение $\sum_{j=1}^{n} w_j x_j - u = 0$ (4.3) описывает границу принятия решений (гиперплоскость в n-мерном входном пространстве), которая разделяет пространство решений на два класса.

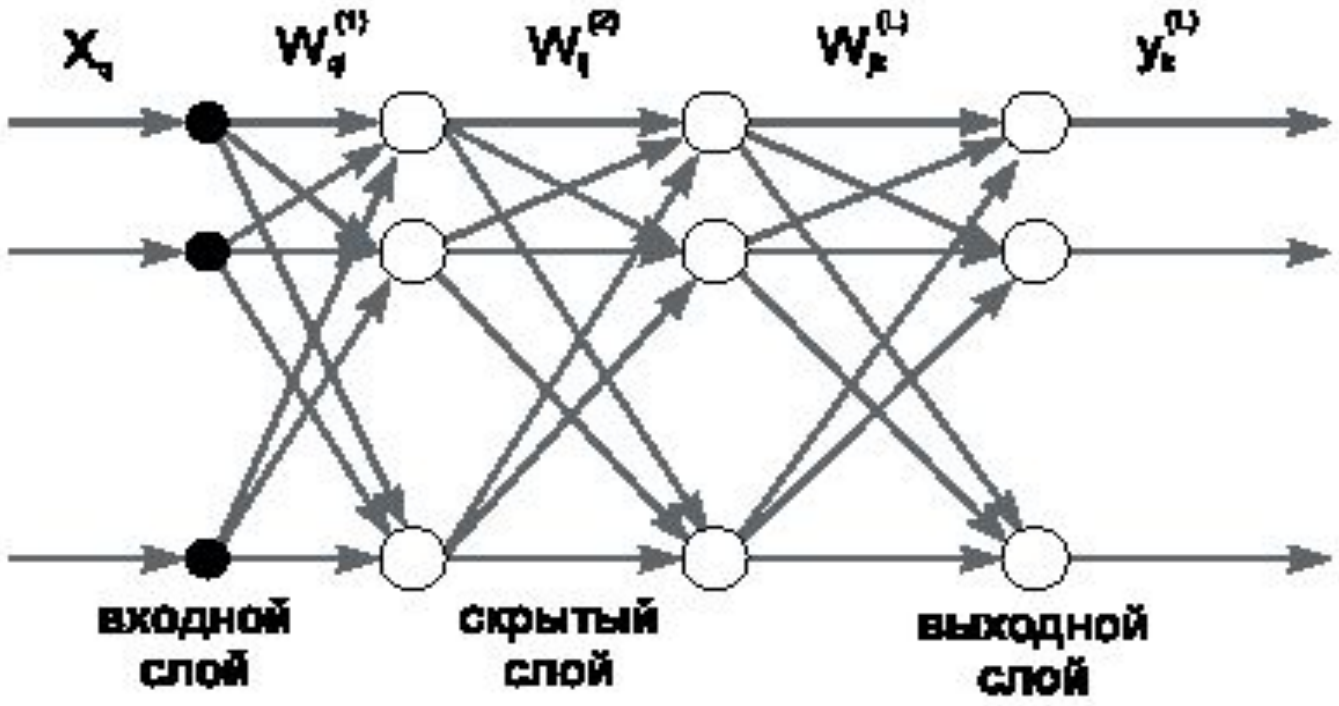
Задача кластеризации

Рассмотрим, например, двумерный случай ($n = 2$). В этом случае формула (4.3) может быть переписана как:

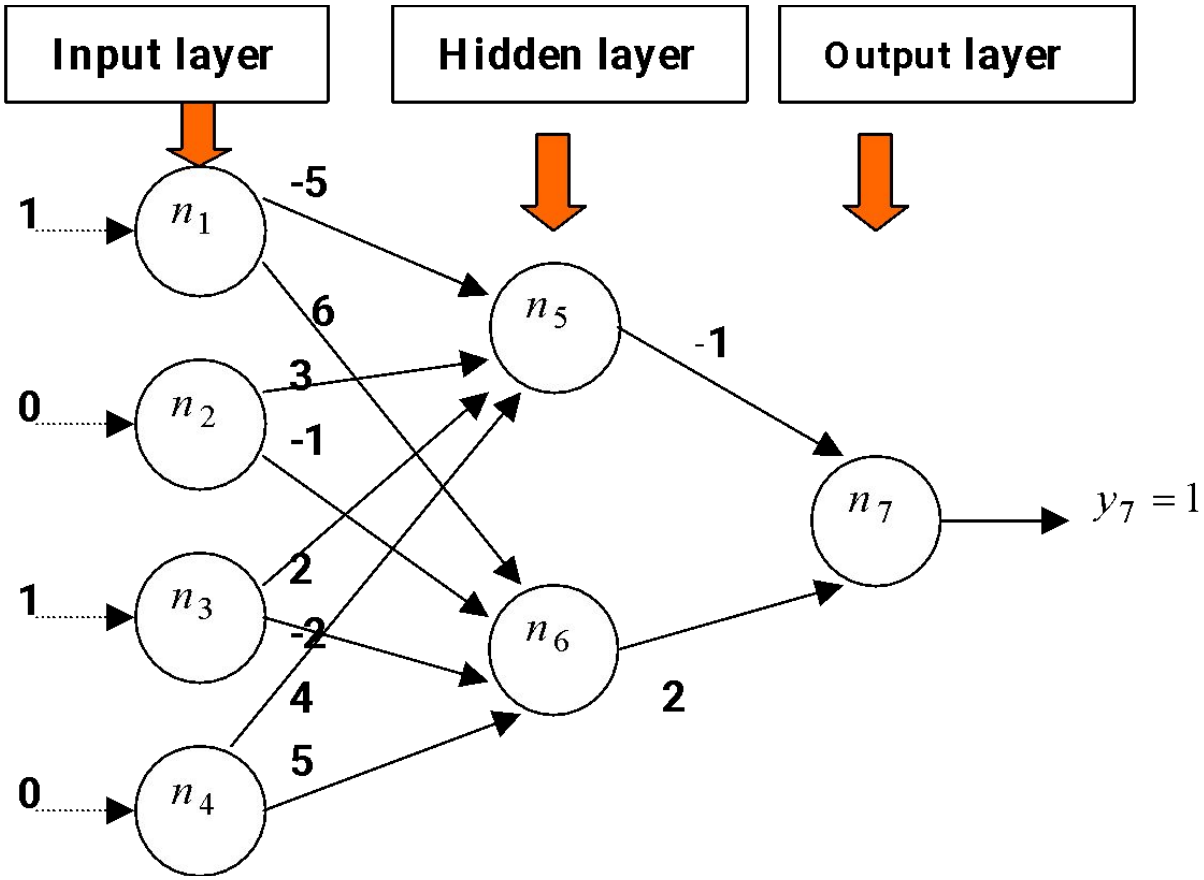
$$w_1 x_1 + w_2 x_2 - u = 0; \quad x_2 = \frac{w_1}{w_2} x_1 + \frac{u}{w_2}.$$



Многослойные сети с прямым распространением



Многослойный «Перцептрон»



Согласно формуле (4.2) выпоним следующие вычисления на сети:

$$f(n_5) = 1(-5) + 0 \cdot 3 + 1 \cdot 2 + 0 \cdot 4 = -3 < 0 \rightarrow y_5 = output(n_5) = 0$$

$$f(n_6) = 1 \cdot 6 + 0 \cdot (-1) + 1 \cdot (-2) + 0 \cdot 5 = 4 > 0 \rightarrow y_6 = output(n_6) = 1$$

$$f(n_7) = y_5 \cdot (-1) + y_6 \cdot 2 = 0 \cdot (-1) + 1 \cdot 2 = 2 > 0 \rightarrow y_7 = output(n_7) = 1$$

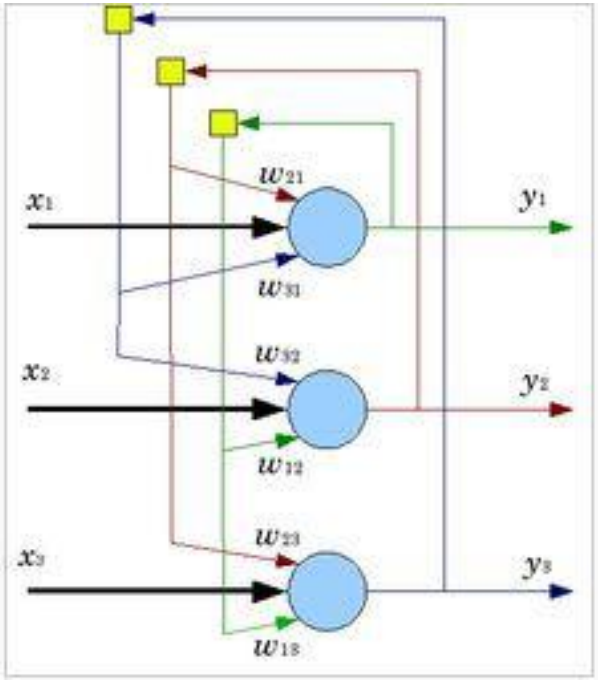
Выходное значение сети $y_7 = 1$.

Многослойный перцептрон как универсальный **аппроксиматор**

- Теорема существования (*Existence Theorem* , Hornik и др,. 1989):
- **MLP может аппроксимировать любую непрерывную функцию с любой заданной степенью точности.**
- Доказательство этой фундаментальной теоремы основывается на теореме Колмогорова (1957), которая гласит
- **Любая непрерывная функция, определенная в *n-мерном* множестве действительных чисел, может быть представлена в виде суммы функций, имеющих своим аргументом суммы непрерывных функций с единственным аргументом.**

Ассоциативная память

- *Ассоциативная память* используется в задачах запоминания, ассоциирования и распознавания образов. Запоминание представляет собой процесс записи входного образа в структуре так называемой автоассоциативной сети (*autoassociative network*) или *сети Хопфильда* (Hopfield network) с целью последующего распознавания входных образов не обязательно точно совпадающих с содержимым памяти.

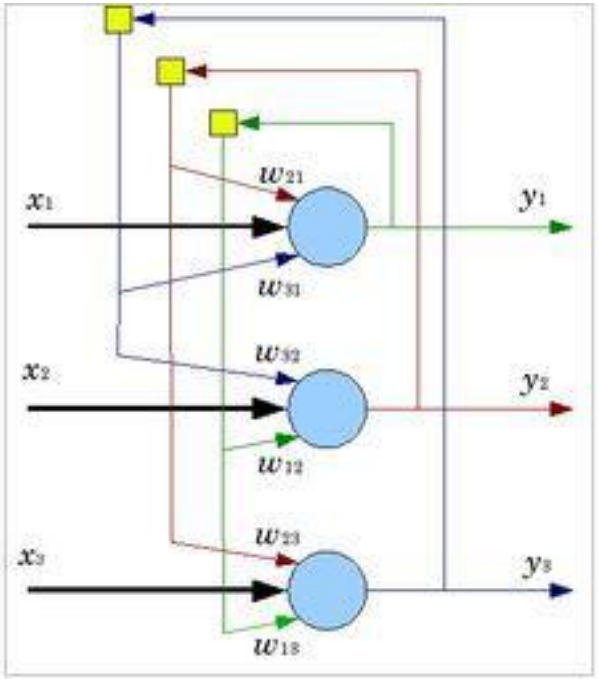


Каждый нейрон в сети связан с каждым другим нейроном. Имеются также внешние входы в нейроны и внешние выходы. Каждый нейрон системы может принимать одно из двух состояний (что аналогично выходу нейрона с пороговой функцией активации). Благодаря своей биполярной природе нейроны сети Хопфилда иногда называют *спинами*.

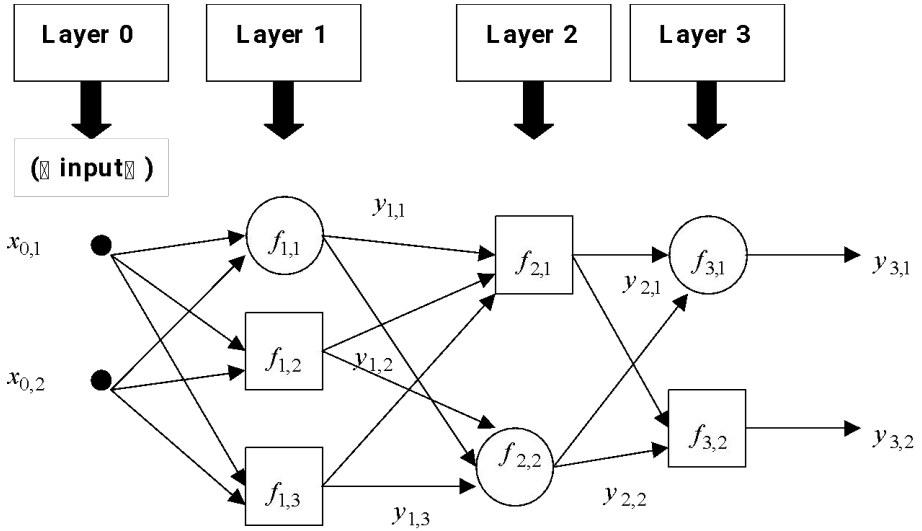
$$y_i = \begin{cases} 1 \\ -1 \end{cases}$$

Сети *Хопфильда*

Взаимодействие спинов сети описывается выражением: $E = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij} x_i x_j$, где w_{ij} - элемент матрицы взаимодействий W, которая состоит из весовых коэффициентов связей между нейронами. В эту матрицу в процессе обучения записывается N «образов» n-мерных бинарных векторов. В сети Хопфилда матрица связей является симметричной $w_{ij} = w_{ji}$, а диагональные элементы матрицы полагаются равными нулю ($w_{ii} = 0$), что исключает эффект воздействия нейрона на самого себя и является необходимым для сети Хопфилда, но не достаточным условием, устойчивости в процессе работы сети.



Адаптивные сети



Сеть имеет L слоев (l =0,1,…,L); l = 0 представляет входной слой. Слой l имеет $N(l)$ вершин. Выходное значение вершины зависит от входных значений и параметров функции активации вершины. Обозначим выход i-той вершины слоя l как $y_{l,i}$.

$$y_{l,i} = f_{l,i}(y_{l-1,1},\ldots, y_{l-1,N(l-1)}, \alpha, \beta, \gamma),$$

где α, β, γ - параметры функции активации $f_{l,i}$ i-той вершины слоя l.

Адаптивными сети называются из-за наличия в них вектора настраиваемых параметров. В ходе адаптивного итерационного процесса сеть обучается устанавливать взаимоотношения между заданной исходной информацией и выходными результатами. Иначе говоря, структура сети по определенным алгоритмам подстраивается таким образом, чтобы минимизировать критерии расхождения входных и выходных параметров. Сети данного вида широко используются в задачах управления.

Нечеткие нейронные сети для задач управления

В рассмотренных выше типах нейронных сетей в качестве модели искусственного нейрона использовалась модель Маккалока и Питтса. Для задач интеллектуального управления была предложена новая модель нейрона - так называемый «*нечеткий нейрон*» (a *fuzzy neuron*).

Типовой нечеткий нейрон (для задач управления) обладает следующими отличительными свойствами:

- его входы являются значения нечетких лингвистических переменных;
- вместо весов используются значения функций принадлежности лингвистических переменных и степени активаций нечетких правил;
- пороговое значение отсутствует.

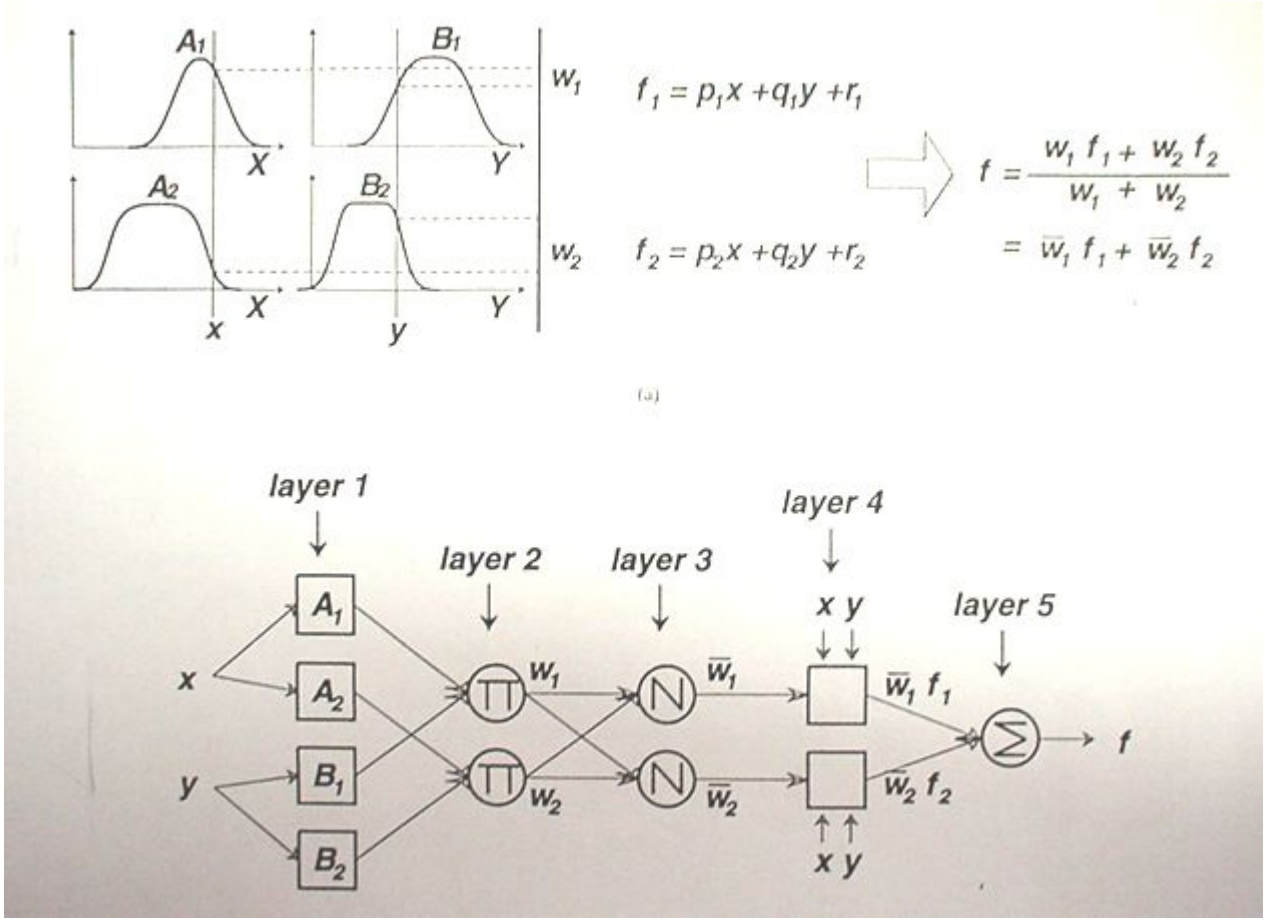
Нечеткие нейронные сети (ННС) являются разновидностью адаптивных сетей. Нечеткий вывод может быть реализован с помощью ННС, включающей в свою структуру нечеткие нейроны. Рассмотрим пример построения ННС для задачи нечеткого вывода в модели Сугено.

Нечеткий контроллер и нейронная сеть

Для простоты предположим, что ННС имеет два входа x и y, и один выход z. Для простоты также возьиеи два правила, например, такие, как:

Правило 1: ЕСЛИ x есть $A_1(x)$ И y есть $B_1(y)$ ТО $z = f_1(x,y) = p_1x + q_1y + r_1$

Правило 1: ЕСЛИ x есть $A_2(x)$ И y есть $B_2(y)$ ТО $z = f_2(x,y) = p_2x + q_2y + r_2$



Слой 1

Слой 1

Выходное значение каждой вершины данного слоя вычисляется следующим образом:

$y_{1,i} = \mu_{A_i}(x)$, для $i =$ 1,2, и $y_{1,i} = \mu_{B_{i-2}}(y)$ для $i = 3,4$,

где x и y - входные значения, $\mu_{A_i}(x)$ и $\mu_{B_{i-2}}(y)$ - функции принадлежности входных переменных.

Например, функции принадлежности входных переменных описываются так называемыми «bell functions» следеющего вида:

$$\mu_{A_i}(x) = \frac{1}{\left[1 + \left(\frac{x - c_i}{a_i}\right)^2\right]^{b_i}},$$

где $\{a_i, b_i, c_i\}$ - множество параметров функций принадлежности. Параметры данного слоя будем называть *параметрами посылок* (*premise parameters*).

Слой 2 и 3

Слой 2

Выходное значение каждой вершины данного слоя вычисляет значение нечеткой конъюнкции двух нечетких величин:

$$y_{2,i} = w_i = \mu_{A_i}(x) \times \mu_{B_i}(y),\ i = 1,2.$$

Таким образом, каждая вершина второго слоя выдает значение степени активации нечеткого правила.

Примечание.В приведенной выше формуле в качестве операции нечеткой конъюнкции выбрано умножение, однако могут использоваться и другие интерпретации этой операции.

Слой 3

Выходное значение каждой вершины данного слоя вычисляет как:

$$y_{3,i} = \overline{w}_i = \frac{w_i}{w_1 + w_2},\ i = 1,2.$$

Слой 4 и 5

Слой 4

$$y_{4,i} = \overline{w}_i f_i = \overline{w}_i (p_i x + q_i y + r_i),$$

где $\overline{w}_i$ - выход i-той вершины слоя 3 и $\{p_i, q_i, r_i\}$ - параметры, называемые *параметрами заключения* (*consequent parameters*).

Слой 5

В данном слое формируется выходное значение сети:

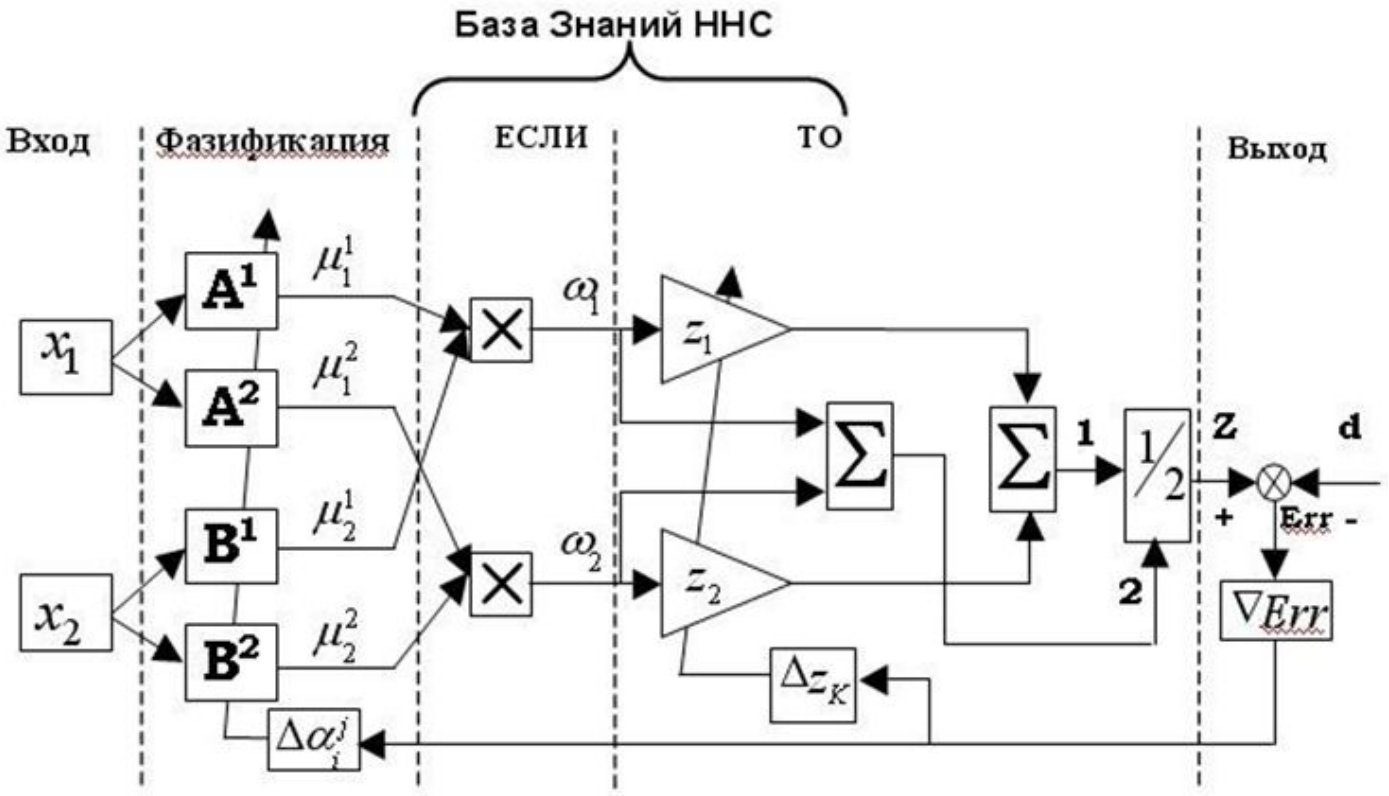
$$y_{5,1} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}. \qquad (4.5)$$

Перепишем формулу (4.5) как:

$$f = \frac{w_1}{w_1 + w_2} f_1 + \frac{w_1}{w_1 + w_2} f_2 = \overline{w}_1 f_1 + \overline{w}_2 f_2 =$$
$$= (\overline{w}_1 x) p_1 + (\overline{w}_1 y) q_1 + (\overline{w}_1 x) r_1 + (\overline{w}_2 x) p_2 + (\overline{w}_2 y) q_2 (\overline{w}_2) r_2.$$

Итак, мы построили ННС, реализующую нечеткий вывод Сугено первого порядка. Далее, с использованием специального алгоритма обучения, можно настраивать параметры сети $\{a_i, b_i, c_i\}$ и $\{p_i, q_i, r_i\}$ на решение определенной задачи.

Структура ННС Суггено



Заключение

1.ИНС рассматривается как взвешенный (или нет) направленный граф (*weighted directed graph*), вершины которого представлены нейронами, а направленные дуги описывают связи между нейронами. ИНС характеризуется тремя параметрами: типом нейронов, архитектурой (организацией связей между нейронами) и алгоритмом обучения в данной сети.

2.Функционирование ИНС может быть рассмотрено как некоторое отображение $F: X \to Y$, где X - пространство входных значений, а Y - пространство выходных значений. Таким образом, ИНС отображает входной вектор $x \in X$ в выходной вектор $y \in Y$ через «фильтр» синаптических весов. Можно представить выходной вектор как $y = F(W, x)$, где W - матрица весов. Матрица W представляет «знания» сети, ее «долговременную память», в то время как, значения активации нейронов представляют текущее состояние сети или «кратковременную память» сети.

Заключение

3. Характерные черты ИНС:

- способность к обучению и адаптации;
- способность к обобщению;
- свойство массивного параллелизма;
- свойство ассоциативной памяти;
- надежность (*robustness*);
- пространственно-временная обработка информации.

Способность к обучению и адаптации является фундаментальным свойством естественного интеллекта. Существует множество определений этих свойств. В контексте ИНС, процесс обучения рассматривается, как процесс обновления весов сети так, что сеть могла оптимальным образом решать поставленную задачу.
Способность к обобщению позволяет на сходные, похожие, стимулы (входные значения) получать сходные, похожие, реакции сети (выходнык значения). Например, сеть может распознать образ по входным данным,содержащим шум.
Надежность ИНС означает, что ИНС может продолжать успешно функционировать даже если сеть или входные данные повреждены.
Ассоциативная память обладает свойством запоминать образы и распознавать их даже если на входе присутствует только часть образа.

Обучение сети

Достоинством ИНС является способность к обучению, в процессе которого синаптические веса сети (или параметры сети) настраиваются с помощью того или иного адаптивного алгоритма с целью наиболее эффективного решения поставленной проблемы.

Чтобы понять или разработать обучающий процесс, прежде всего, необходимо иметь некоторые знания о среде функционирования ИНС. Другими словами, нужно знать, какая информация доступна в сети. Это так называемая *парадигма обучения*.

Существует три парадигмы обучения:

- супервизорное (с учителем),
- несупервизорное (без учителя),
- гибридное.

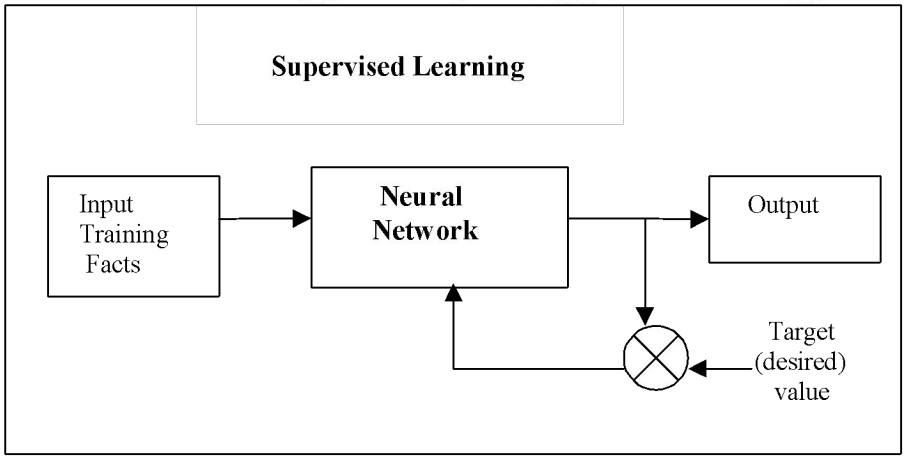
Супервизорное

В супервизорном обучении, нейросеть обеспечивается правильным (корректным) выходом для заданного множества входных образов (значений).

Выбор данных для обучения сети и их обработка является самым сложным этапом решения задачи. Набор данных для обучения должен удовлетворять нескольким критериям:

- Репрезентативность — данные должны иллюстрировать истинное положение вещей в предметной области;
- Непротиворечивость — противоречивые данные в обучающей выборке приведут к плохому качеству обучения сети.

Исходные данные преобразуются к виду, в котором их можно подать на входы сети. Пара «вход-выход» называется *обучающей парой* или *обучающим вектором*.



Супервизорное

В процессе обучения используется множество обучающих пар «вход- желаемый (или целевой) выход». По обучающему входу сеть вычисляет результат, получаемый в данной структуре сети, и сравнивает его с обучающим выходом. После чего вычисляется *ошибка* между обучающим выходом и реальным. Просматривая обучающую выборку множество раз, сеть оптимизирует свои параметры так, чтобы ошибка была меньше заданной величины. После чего обучение заканчивается и обученную сеть можно протестировать на множестве тестовых (проверочных) данных.

Если на проверочных данных ошибка не увеличивается, то сеть действительно выполняет поставленнйю задачу. Если ошибка на обучающих данных продолжает уменьшаться, а ошибка на тестовых данных увеличивается, значит, сеть перестала выполнять свою задачу и просто «запоминает» обучающие данные. Это явление называется переобучением сети или оверфиттингом. В таких случаях обучение обычно прекращают. В процессе обучения могут проявиться другие проблемы, такие как паралич или попадание сети в локальный минимум поверхности ошибок. Невозможно заранее предсказать проявление той или иной проблемы, равно как и дать однозначные рекомендации к их разрешению

Несупервизорное обучение (обучение без учителя)

В обучении без учителя знание «обучающего (желаемого) выхода» не требуется.

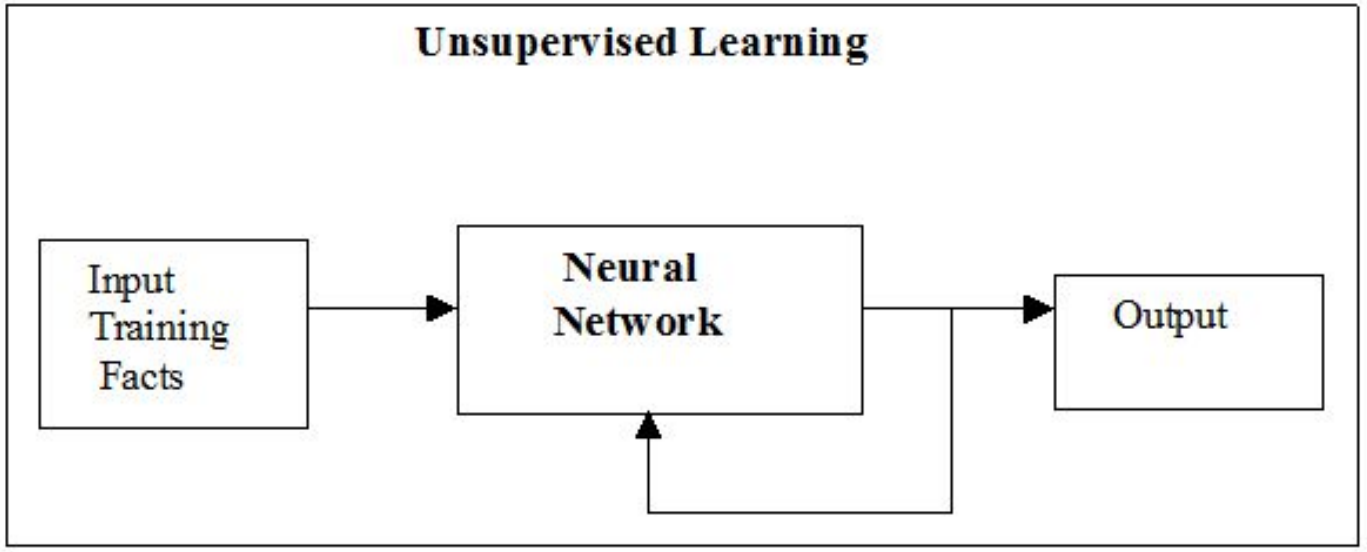


Рисунок 4-13. Unsupervised learning scheme in a neural network.

Несупервизорное обучение

Сеть исследует структуру входных данных и организует их в гоуппы. В процессе обучения сеть в определенном порядке просматривает обучающую выборку. Порядок просмотра может быть последовательным, случайным и т. д. Некоторые сети, обучающиеся без учителя, например, сети Хопфилда просматривают выборку только один раз.

В отличие от супервизорного обучения, в несупервизорном обучении нет знаний о «желаемом выходе». Сеть обучается на входных данных до тех пор, пока не придет в некоторое «стабильное» состояние. В этом смысле, такая форма обучения более близка к естественному механизму в биологических нейросетях.

Гибридное обучение соединяет в себе черты двух приведенных выше схем обучения.

Для задач интеллектуального управления будем рассматривать супервизорное обучение сети. Рассмотрим типовой метод обучения, называемый *методом обратного распространения ошибки*.

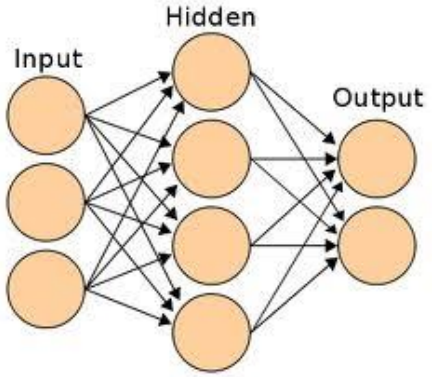
Правило обучения, основанное на коррекции ошибки (Error-correction rules of learning)

Вычисление ошибки может происходить несколькими способами:

$Err = |(d - y)|$ (текущая ошибка);

$Err = \frac{(d - y)^2}{2}$ (средне-квадратичная ошибка);

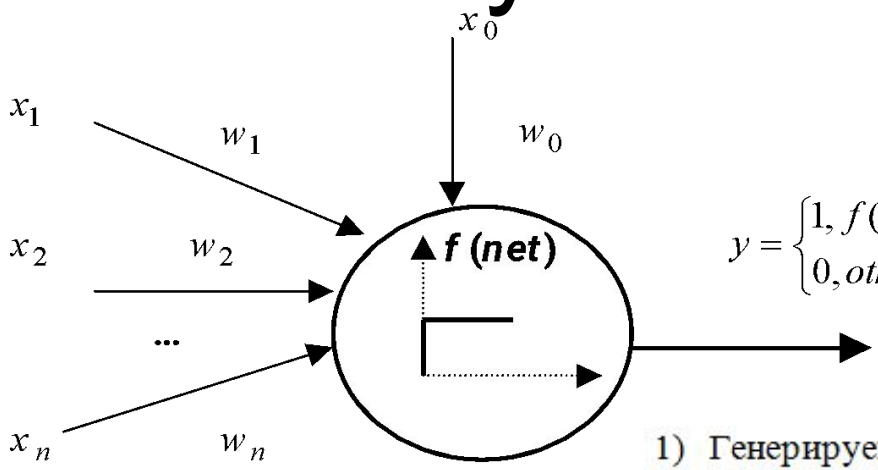
$Err = \sum_{(p)} \sum_{(j)} Err_j^p$ (общая ошибка),



где p - число обучающих пар; j - число выходов нейросети, и Err_j^p - ошибка для p -той обучающей пары.

В нейросетях существует два наиболее популярных алгоритма супервизорного обучения: обучение перцептрона (*perceptron learning*) и алгоритм обратного распространения ошибки (*error back propagation-based learning*)

Обучение перцептрона



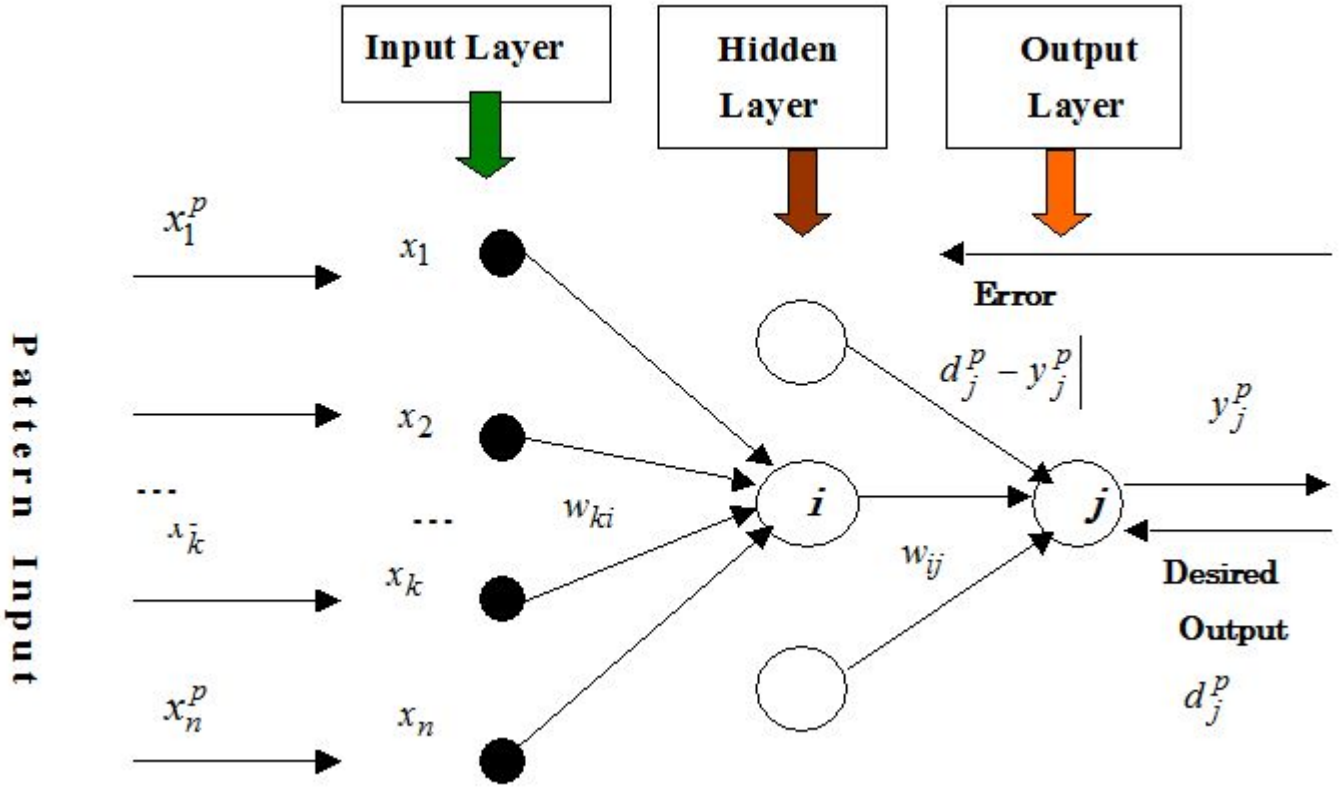
1) Генерируем начальные веса w_j $(j = 0,1.2,...n)$ с помощью генератора случайных (малых) чисел.
2) Поставляем на вход персептрона входной образ $(x_1, x_2,..., x_n)^t$, где t - порядковый номер текущей итерации процесса обучения.
3) Вычисляем реальный выход персептрона: $y = \begin{cases} 1, & \text{if } \sum_{j=0}^{n} w_j x_j > 0, \\ 0, & \text{в противном случае} \end{cases}$.
4) Вычисляем ошибку $Err = (d - y)$, где d - желаемый выход для входного образа $(x_1, x_2,..., x_n)^t$.
5) Обновляем значения весов следующим образом:
   $w_j(t+1) = w_j(t) + \eta \cdot Err \cdot x_j$, где η - заданный коэффициент обучения (число между 0 и 1).
6) Повторяем шаги 2-5 до тех пор, пока ошибка не станет меньше заданного уровня толерантности $Err < Err_{\max}$ (заданный уровень).

Теорема сходимости Перцептрона

- **Теорема сходимости перцептрона**, описанная и доказанная Ф. Розенблаттом показывает, что элементарный перцептрон, обучаемый по такому алгоритму, независимо от начального состояния весовых коэффициентов и последовательности появления стимулов *всегда* приведет к достижению решения за конечный промежуток времени.

Алгоритм обратного распространения ошибки (*error back propagation-based learning*)

- Алгоритм обратного распространения ошибки применяется для многослойного перцептрона и сетей с прямым распространением сигнала



Для возможности применения этого метода функция активации нейронов должна быть дифференцируема.

Алгоритм обратного распространения ошибки

- Необходимо настроить синоптические веса, что бы ошибка была минимальной

Для этого будем реализовывать **градиентный спуск**, то есть будем подправлять веса после каждого обучающего примера и, таким образом, "двигаться" в многомерном пространстве весов. Таким образом, будем использовать следующее правило изменения весов, которое в общем виде записывается как:

на $(t+1)$-ой итерации алгоритма добавлять к каждому весу w_{ij} его поправку

$$\Delta w_{ij}(t+1) = -\eta \frac{\partial E}{\partial w_{ij}(t)}, \qquad (4.6)$$

где где $0 < \eta < 1$ - множитель, задающий скорость "движения".

Алгоритм обратного распространения ошибки

Рассмотрим многослойный перцептрон с множеством выходных вершин (рис.4-14), обозначим через j - число его выходов. В качестве общей (глобальной) ошибки возьмем общую средне-квадратичную ошибку для заданного множества обучающей выборки:

$$E = \sum_{(p)} \sum_{(j)} Err_j^p \text{ , и } Err_j^p = \frac{(d_j^p - y_j^p)^2}{2},$$

где d_j^p есть желаемый результат j-той выходной вершины для p-ой обучающей выборки, а y_j^p - реальный результат j-той выходной вершины для p-ой обучающей выборки.

Вернемся к формуле (4.6). Как считать представленную в ней производную?

Правило Дельты (Delta rule):

$$\Delta w_{ij}(t+1) = \eta \cdot Err_j \cdot y_i \qquad (4.7)$$

Алгоритм обратного распространения ошибки

Обсудим теперь, как вычислять ошибку Err_j **?**

Если j-тая вершина – выходная, то мы можем ее вычислить как $\left|d_j^p - y_j^p\right|$, но как быть с ошибкой в скрытых слоях сети. Рассмотрим две версии алгоритма обратного распространения ошибки.

Алгоритм обучения состоит из двух фаз: (1) фаза прямого распространения сигнала (a *forward pass*) и (2) фаза обратного распространения сигнала (*a backward pass*). В первой фазе входные обучающие сигналы подаются в сеть и вычисляются выходные значения и выходная ошибка. Во второй фазе процесс вычисления идет в обратную сторону – от выходной ошибки к внутренним слоям сети, в результате чего настраиваются новые значения синаптических узлов.

Первая фаза

Фаза прямого распространения сигнала:

1) Генерируем начальные веса w_j $(j = 0,1.\ 2,...n)$ с помощью генератора случайных (малых) чисел.
2) Случайным образом выбираем входной образ $\vec{x} = (x_1, x_2, ...x_n)^p$ из заданного множества обучающих пар.
3) Распространяем сигнал $\vec{x}$ через сеть и вычисляем результат.
4) Для каждого j -го выходного нейрона (слой L) вычисляем ошибку $Err_j = \left| d_j^p - y_j^p \right|$.

Вторая фаза

Фаза обратного распространения сигнала:

5) Обновляем веса между i-тыми нейронами слоя $L-1$ и j-м выходным нейроном (слоя L) следующей поправкой:
$$\Delta w_{ij}(t+1) = \eta Err_j y_j (1-y_j) y_i + \alpha \Delta w_{ij}(t),$$

- где α - параметр, называемый «моментом» (*momentum*), $t+1$ - текущая итерация процесса обучения, а t - его предыдущая итерация.

6) Вычисляем ошибку Err_i для i-тых нейронов слоя $L-1$ как:

$$Err_i = \sum_j Err_j w_{ij} .$$

7) Распространяем процесс вычисления назад по сети к k-тым нейронам промежуточных слоев $(l = L-2,\ldots,0)$:

$$\Delta w_{ki}(t+1) = \eta Err_i y_i (1-y_i) y_k + \alpha \Delta w_{ki}(t) \text{ и } Err_k = \sum_i Err_i w_{ki} .$$

8) Переходим к шагу 2 и повторяем процесс до тех пор, пока пока ошибка не станет меньше заданного уровня толерантности $Err < Err_{\max}$.