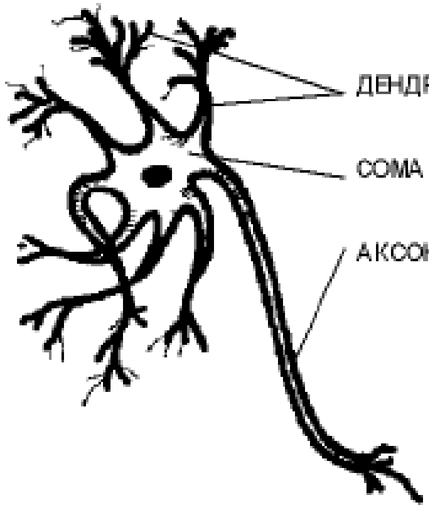
Раздел 2. Нейронные сети

- Модель искусственного нейрона
- Полносвязные нейронные сети
- Сверточные нейронные сети
- Реккурентные нейронные сети

Идея:

•моделирование процессов, происходящих в нервной системе живых организмов при обработке поступающей информации.

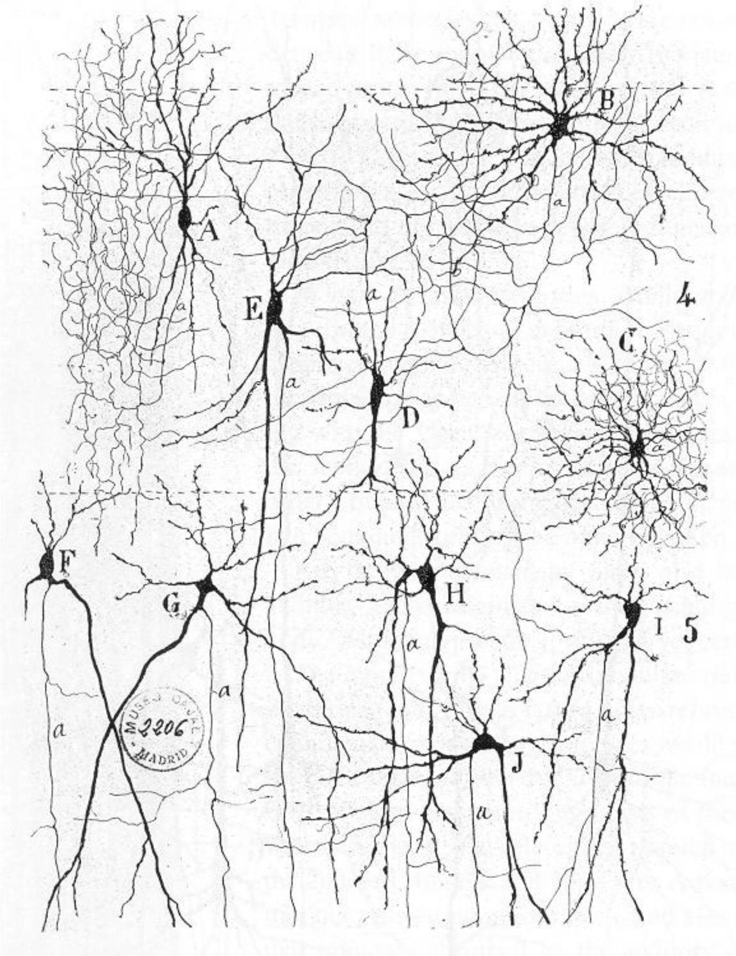
•мозг гораздо быстрее и точнее компьютера может обрабатывать аналоговую информацию: распознавать изображения, вкус, звуки, читать чужой почерк, оперировать качественными понятиями.



- *Дендриты* отвечают за получение нервных импульсов от других нейронов. Далее под их воздействием *сома* испытывает специфическое возбуждение, которое затем распространяется по аксону.

- *Нейронная сеть - м*ножество нейронов, взаимодействующих между собой.
- *Синапс* – область контакта аксона одного нейрона с дендритами других нейронов.

Мозг человека состоит из белого и серого веществ: белое – это тела нейронов, а серое – это соединительная ткань между нейронами, или *аксоны и дендриты.* Мозг состоит примерно из 10^{11} нейронов, связанных между собой.



Нейрон может иметь до 10000 дендритов. Таким образом, мозг содержит примерно 10^{15} взаимосвязей.

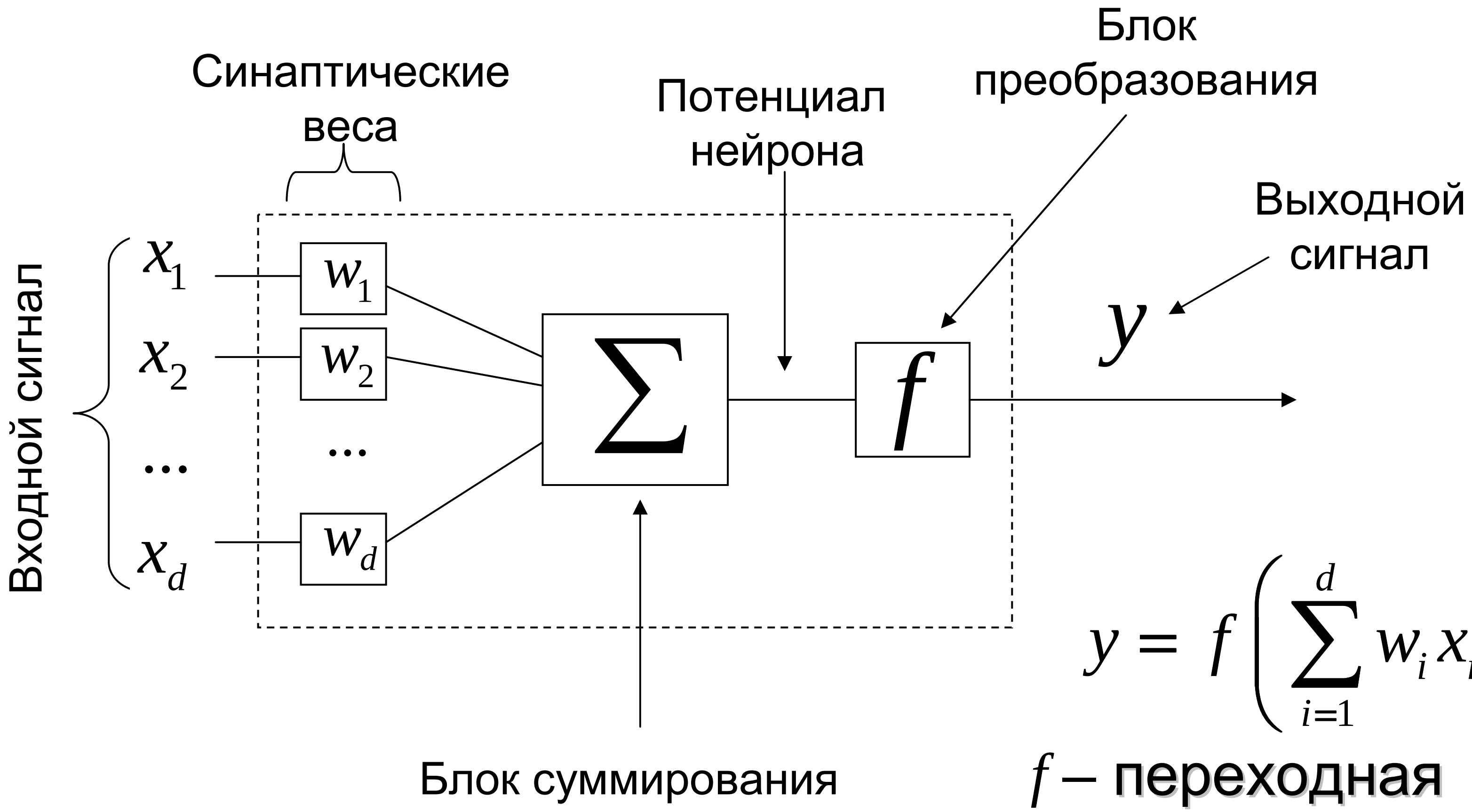
Нейроны взаимодействуют посредством серий импульсов, длящихся несколько миллисекунд, каждый импульс - сигнал с частотой 1-100 герц.

Нейронная сеть

- Совокупность соединенных между собой нейронов;
- Сеть осуществляет преобразование входного сигнала с рецепторов в выходной, являющейся реакцией организма на внешнюю среду.

- Мозг – система из параллельных «процессоров».
- Большое число параллельно функционирующих простых устройств – работает гораздо эффективнее, чем сложные последовательные устройства.

2.1. Модель кибернетического нейрона

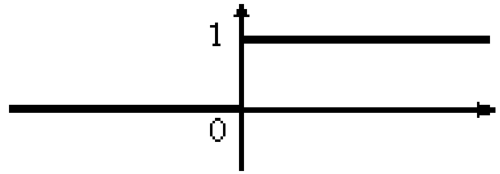
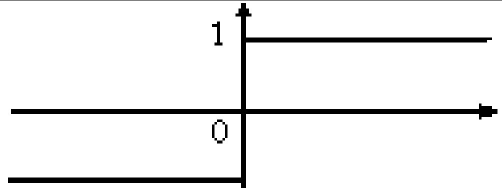
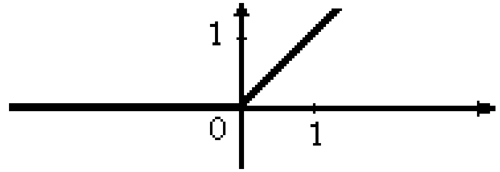
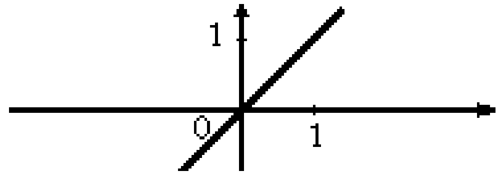


$$y = f\left(\sum_{i=1}^{d} w_i x_i\right)$$

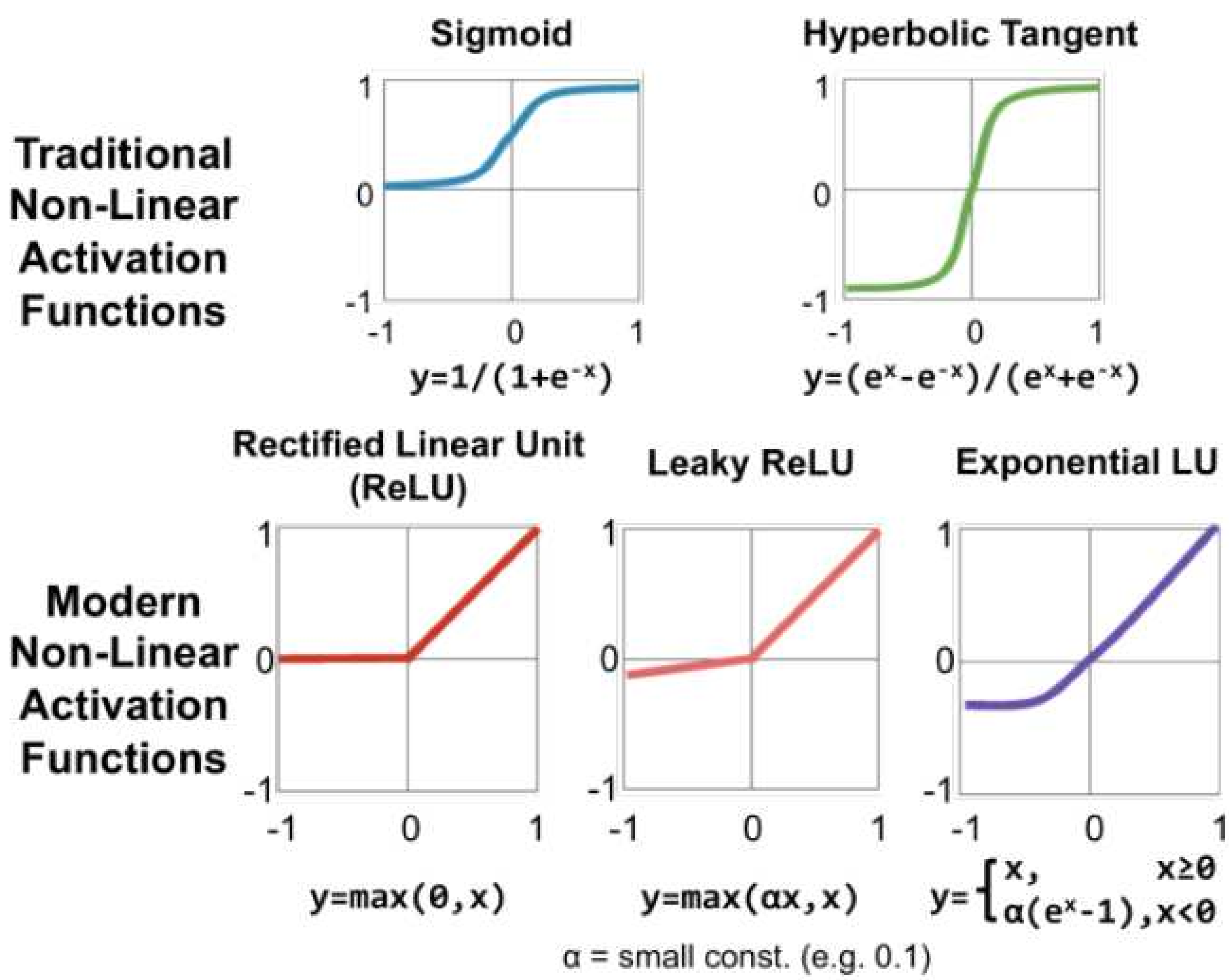
f – переходная функция, функция активации

McCulloch, W. and Pitts, W. (1943)

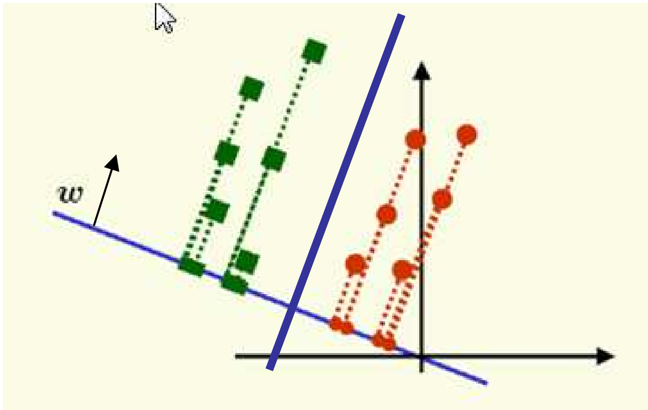
Функции активации нейронов

Название	Формула	График
Пороговая	$f(u)=\begin{cases}0 & u<0\\ 1 & u\geq 0\end{cases}$	
Знаковая (сигнатурная)	$f(u)=\begin{cases}1 & u>0\\ -1 & u\leq 0\end{cases}$	
Полулинейная rectifier linear unit (ReLU)	$f(u)=\begin{cases}u & u>0\\ 0 & u\leq 0\end{cases}$	
Линейная	$f(u)=u$	

Функции активации нейронов



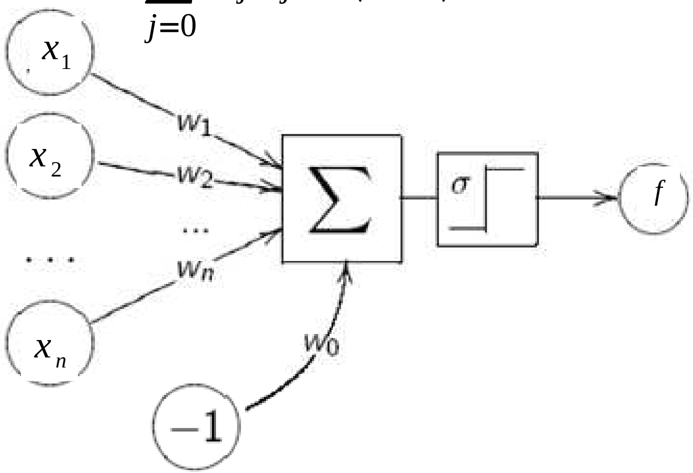
Линейный дискриминант Фишера: $f(x) = sign\left(\sum_j w_j x_j - w_0\right)$



Модель линейного персептрона (персептрон Розенблатта):

$$f(x, w) = \sigma(\sum_j w_j x_j - w_0),$$

где $\sigma(\cdot)$ функция активации (в частности, sign). Можно ввести $x_0 \equiv -1$ $\Rightarrow \sum_{j=1}^{n} w_j x_j - w_0 = \sum_{j=0}^{n} w_j x_j = \langle w, x \rangle$



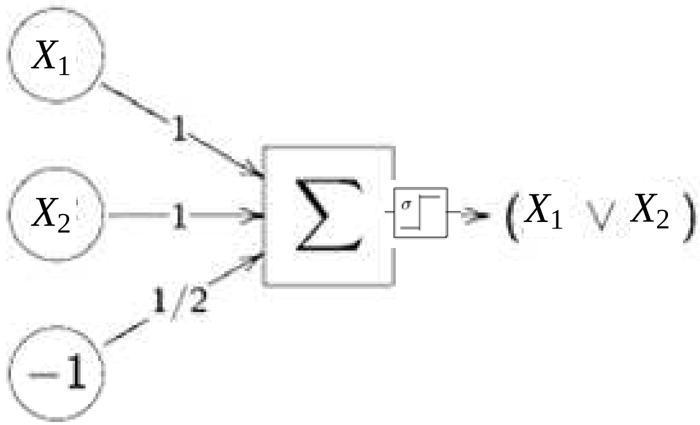
Пример. Функции И, ИЛИ, НЕ от булевых переменных X_1 и X_2:

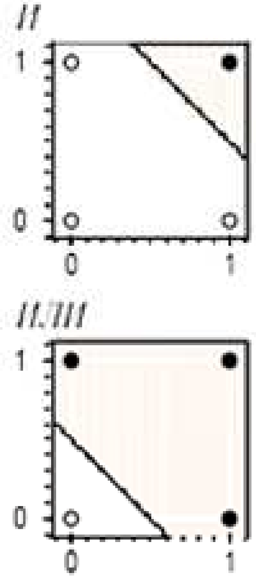
$$X_1 \wedge X_2 = \left[X_1 + X_2 - \frac{3}{2} > 0 \right];$$

$$X_1 \vee X_2 = \left[X_1 + X_2 - \frac{1}{2} > 0 \right];$$

$$\neg X_1 = \left[-X_1 + \frac{1}{2} > 0 \right];$$

$([\cdot])$ – индикаторная функция.





Задача обучения персептрона:
- подобрать веса *w* так, чтобы ошибка на обучающей выборке была минимальной.

Алгоритм Розенблатта (коррекции ошибок):

Пусть $t = 1, 2, \ldots$ - номер шага (эпохи);

1. Случайным образом задаются веса $w(0)$;
2. По очереди предъявляются объекты выборки;
   для каждого объекта $x^{(i)}$ вычисляется выход (расстояние до разделяющей гиперплоскости с учетом знака)
   $$\tilde{y}^{(i)} = \left\langle w, x^{(i)} \right\rangle;$$
3. Если $\tilde{y}^{(i)} \cdot y^{(i)} > 0$, то веса не изменяются;
4. Если $\tilde{y}^{(i)} \cdot y^{(i)} \le 0$, то проводится коррекция (правило Хебба):
   $$w(t+1) = w(t) + \eta\, x^{(i)} \cdot y^{(i)},$$
   где $\eta > 0$ - параметр (темп обучения).

При этом будет выполняться:

$$\tilde{y}^{(i)}(t+1) \cdot y^{(i)} = \left\langle w(t+1), x^{(i)} \right\rangle \cdot y^{(i)} = \left\langle w(t) + \eta\, x^{(i)} \cdot y^{(i)},\ x^{(i)} \right\rangle \cdot y^{(i)} =$$

$$\left\langle w(t),\ x^{(i)} \right\rangle \cdot y^{(i)} + \eta \left\langle x^{(i)} \cdot y^{(i)},\ x^{(i)} \cdot y^{(i)} \right\rangle = \tilde{y}^{(i)} \cdot y^{(i)} \underbrace{+ \eta \| x^{(i)} \|}_{\ge 0};$$

сдвиг в «правильном» направлении

5. Повторяются 2)-4) пока не выполнится правило останова:

   а) веса $w(t)$ перестали изменяться; или

   б) ошибка распознавания стала меньше заданного параметра.

Теорема (Новиков, 1962).

Пусть выборка $\left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$ линейно разделима, т.е. $\exists \tilde{w}, \exists \delta > 0 : \left\langle \tilde{w}, x^{(i)} \right\rangle y^{(i)} > \delta$ для всех $i = 1, \ldots, N$.

Тогда алгоритм Розенблатта находит вектор весов w:

- разделяющий обучающую выборку без ошибок;
- при любом $w(0)$;
- при любом темпе обучения $\eta > 0$;
- независимо от порядка объектов;
- за конечное число итераций;
- если $w(0) = 0$, то число итераций $t_{\max} \le \dfrac{1}{\delta^2} \max \| x^{(i)} \|$.

Правило Розенблатта (“delta-rule”, “error-correcting learning”)

$$w(t+1) = w(t) + \eta\,\delta^{(i)} x^{(i)},$$

где $\delta^{(i)}(t) = y^{(i)} - \tilde{y}^{(i)}(t)$.

Веса изменяются так, чтобы уменьшить разницу между выходом ($\tilde{y}^{(i)}$) и целевым значением ($y^{(i)}$).

Качество персептрона (суммарная ошибка):

$$Q(w) = \frac{1}{2}\sum_i Q_i = \frac{1}{2}\sum_i (y^{(i)} - \tilde{y}^{(i)})^2 \to \min$$

чтобы локально минимизировать Q, на каждом шаге нужно корректировать веса: $w_j(t+1) = w_j(t) - \eta \dfrac{\partial Q_i}{\partial w_j}$;

$$\frac{\partial Q_i}{\partial w_j} = \frac{\partial Q_i}{\partial \tilde{y}^{(i)}} \frac{\partial \tilde{y}^{(i)}}{w_j} = \underbrace{(y^{(i)} - \tilde{y}^{(i)})}_{\delta^{(i)}}(-x^{(i)}) \Rightarrow w(t+1) = w(t) + \eta\,\delta^{(i)} x^{(i)}$$

Направления дальнейшего развития алгоритма Розенблатта:

• использование произвольной функции потерь (частота ошибки => риск);

• поиск наилучшего способа инициализации весов;

• изменить порядок рассмотрения объектов:

•а) случайный;

•б) оказывать большее внимание объектам, которые дают ошибку;

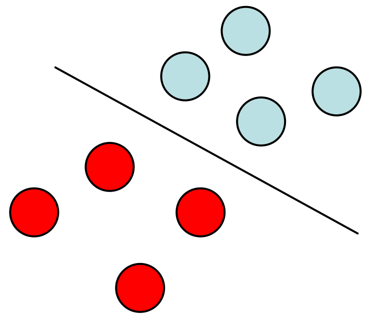
•в) не рассматривать объекты – выбросы;

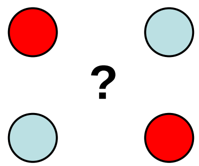
• изменение темпа обучения ($\eta_t=1/t$); скорейший градиентный спуск; пробные случайные шаги;

2.2. Полносвязные нейронные сети

Основной недостаток линейного персептрона:

алгоритм работает только для линейно разделимых образов





Выход: использовать многослойный персептрон

Пример: функция XOR

Функция $X_1 \oplus X_2 = [X_1 \neq X_2]$ не реализуется одним нейроном.

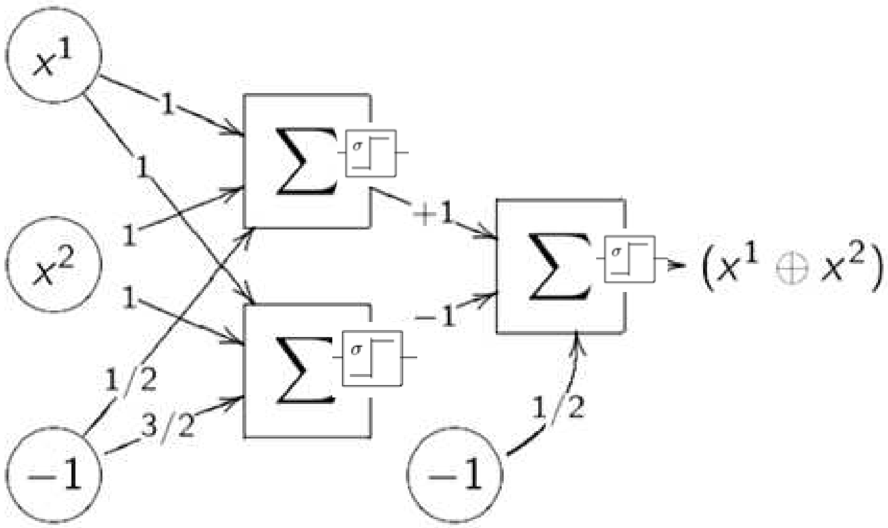
Два способа реализации

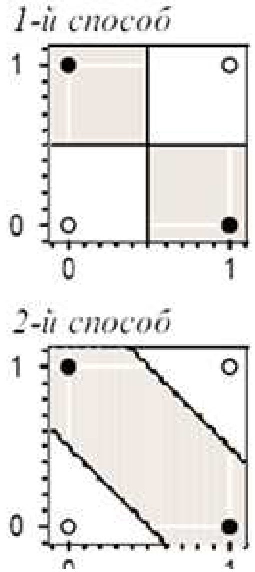
- добавление нелинейного признака:

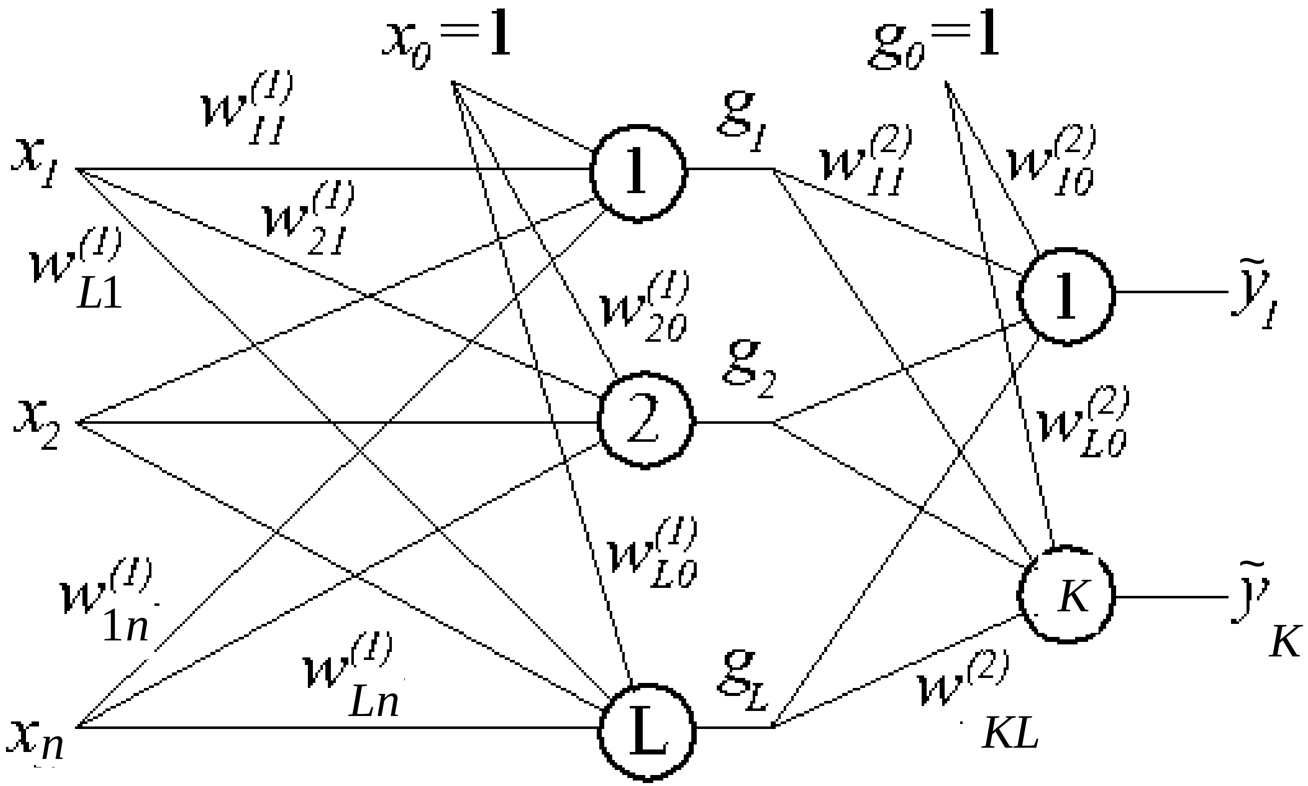
$$X_1 \oplus X_2 = [X_1 + X_2 - 2X_1X_2 - 1/2 > 0];$$

- Сеть (суперпозиция) функций И, ИЛИ, НЕ:

$$X_1 \oplus X_2 = [(X_1 \vee X_2) - (X_1 \wedge X_2) - 1/2 > 0]$$





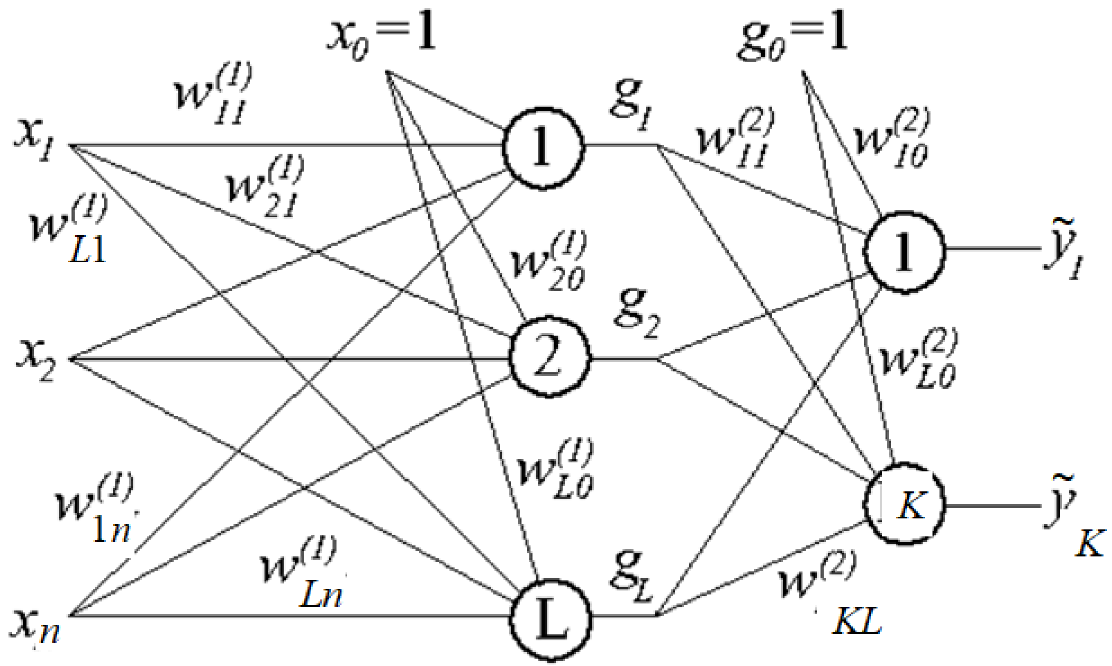


Структурная схема двухслойного персептрона.

Верхние индексы в скобках (*m*), *m*=1,2 - номер слоя нейрона. Для обозначения структуры сети используется кодировка "*n-L-K*".

K= число образов.

Выходные сигналы нейронных слоев:



$$g_l = f\left(\sum_{j=0}^{n} w_{lj}^{(1)} \cdot x_j\right), \; l = 1,\ldots,L$$

$$\tilde{y}_q = f\left(\sum_{l=0}^{L} w_{ql}^{(2)} \cdot g_l\right) = f\left(\sum_{l=0}^{L} w_{ql}^{(2)} \cdot f\left(\sum_{j=0}^{n} w_{lj}^{(1)} \cdot x_j\right)\right), \; q = 1,\ldots,K$$

Функции активации всех нейронов сети одинаковы.

Можно ли произвольную функцию представить нейросетью?

Решение тринадцатой проблемы Гильберта:

Теорема (Колмогоров, 1957)

Любая непрерывная функция n аргументов на единичном кубе $[0,1]^n$ представима в виде суперпозиции непрерывных функций одного аргумента и операции сложения:

$$f(x_1,\ldots,x_n) = \sum_{k=1}^{2n+1} h_k \left(\sum_{i=1}^{n} \varphi_{i,k}(x_i) \right),$$

где $h_k, \varphi_{i,k}$ - непрерывные функции одного аргумента.

Андрей Колмогоров, «О представлении непрерывных функций нескольких переменных суперпозициями непрерывных функций меньшего числа переменных», *Известия АН СССР*, 108 (1956), с. 179—182; английский перевод: Amer. Math. Soc. Transl., 17 (1961), p. 369—373.

Теорема (Цыбенко, 1989)

Любая непрерывная функция n аргументов на единичном кубе $[0,1]^n$ сколь угодно близко представима искусственной нейронной сетью прямой связи (feed-forward; в которых связи не образуют циклов) с одним скрытым слоем:

$\exists \alpha_k, w_k, b_k$:

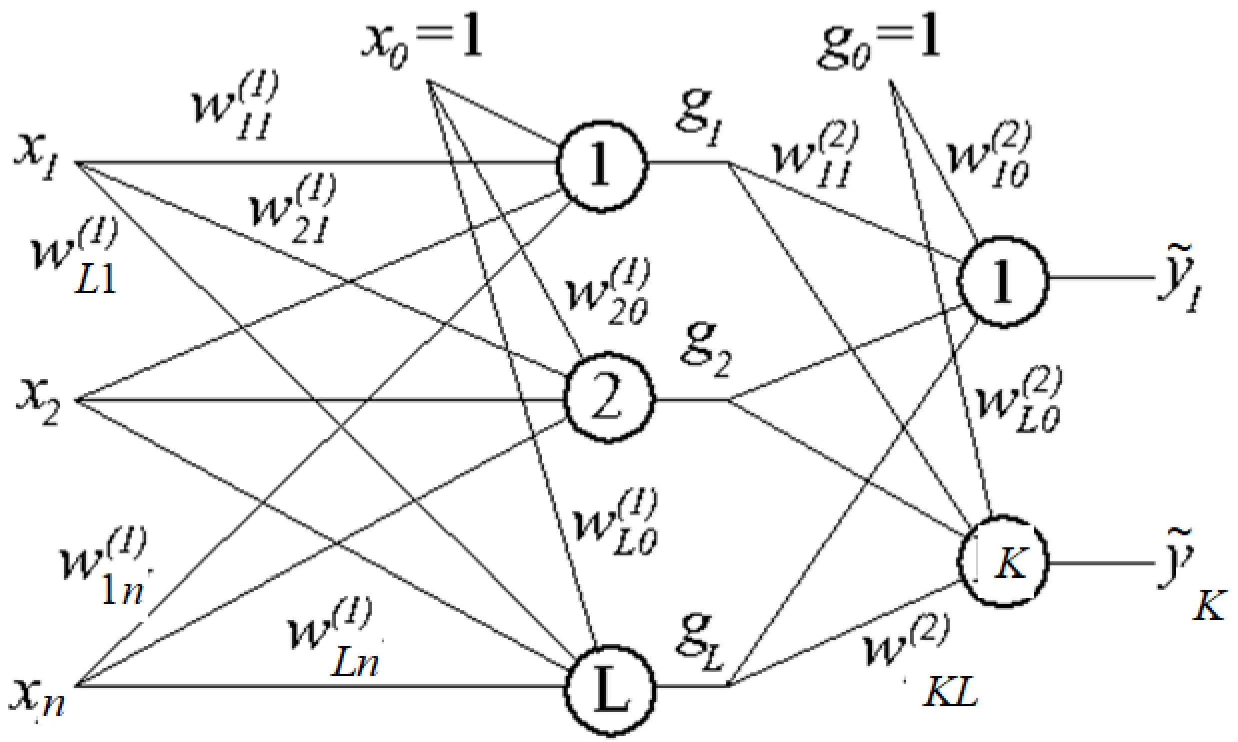
$$f(x_1,\dots,x_n) = \sum_{k=1}^{N} \alpha_k \sigma\left(w_k{}^T x + b_k \right),$$

где $\sigma(t) = \dfrac{1}{1+e^{-t}}$ - функция активации сигмоида.

Cybenko, G. V. Approximation by Superpositions of a Sigmoidal function // Mathematics of Control Signals and Systems. — 1989. — Т. 2, № 4. — С. 303—314.

С помощью суперпозиции линейных функций и нелинейной функции активации можно аппроксимировать любую непрерывную функцию с любой заданной точностью

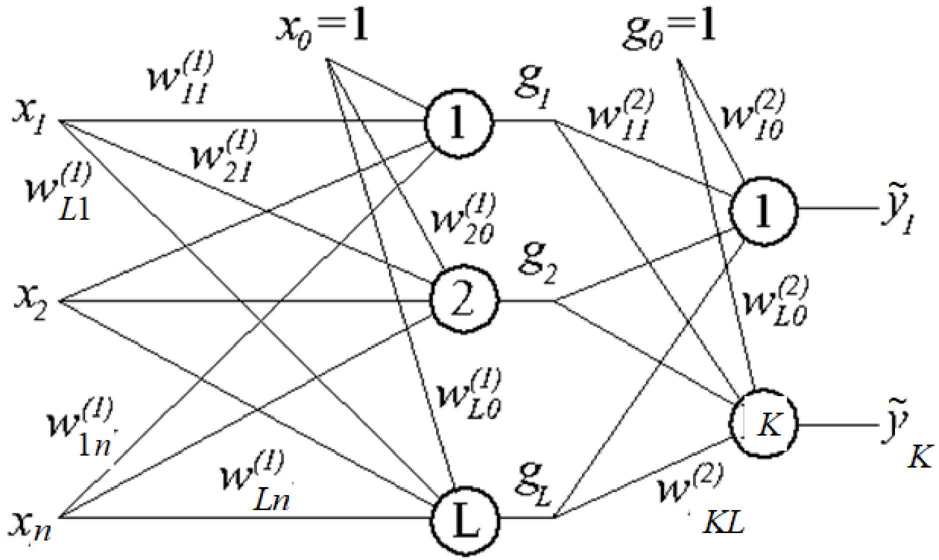
Обучение многослойного персептрона



Задача: найти такие веса, чтобы суммарная ошибка обучения была минимальной

$$Q_\Sigma(w) = \frac{1}{2}\sum_{i=1}^{N}\sum_{q=1}^{K}\left(\tilde{y}_q(x^{(i)}) - y_q^{(i)}\right)^2 \to \min_w .$$

Алгоритм стохастического градиента:



1.Задать случайные веса $(w_{lj}^{(1)}(0), w_{ql}^{(2)}(0)),\ l=1,...,L,\ q=1,...,K$;

2.Повторять для $t=1,2,...$:

3.Вычислить критерий $Q_\Sigma\left(w(t)\right)$;

4.Для каждого i-го объекта ($i=1,...,N$): положим

$$w_{mk}^{(h)}(t+1)=w_{mk}^{(h)}(t)-\eta\frac{\partial Q(w(t))}{\partial w_{mk}^{(h)}(t)};$$

5.Продолжать 2)-4), пока либо критерий, либо веса не стабилизируются.

Нахождение градиента – трудоемкая операция.

Метод обратного распространения ошибок – позволяет эффективно его вычислять.

Суммарная ошибка для объекта $x^{(i)}$: $Q(\tilde{y}) = \frac{1}{2}\sum_{q=1}^{K}\left(\tilde{y}_q(x^{(i)}) - y_q^{(i)}\right)^2$.

Частная производная:

$\frac{\partial Q(\tilde{y})}{\partial \tilde{y}_q} = \tilde{y}_q - y_q^{(i)} = \tilde{\varepsilon}_q^{(i)}$ - ошибка для объекта $x^{(i)}$ на выходе сети.

Частные производные по выходам скрытого слоя:

$$\frac{\partial Q(g)}{\partial g_l} = \frac{1}{2}\frac{\partial}{\partial g_l}\sum_{q=1}^{K}\left(\overbrace{f\left(\sum_{l=0}^{L} w_{ql}^{(2)} \cdot g_l\right)}^{\tilde{y}_q} - y_q^{(i)}\right)^2 = \sum_{q=1}^{K}\left(\tilde{y}_q - y_q^{(i)}\right) f_q' \, w_{ql}^{(2)} =$$

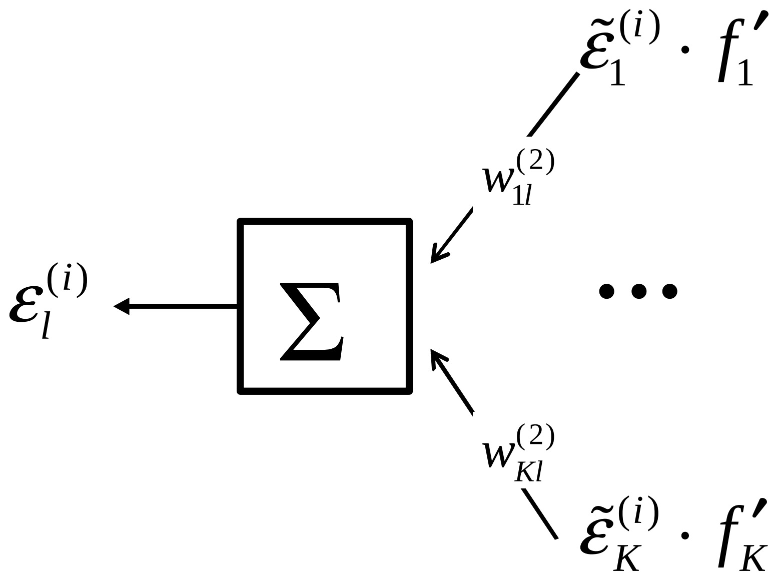
$= \sum_{q=1}^{K} \tilde{\varepsilon}_q^{(i)} f_q' \, w_{ql}^{(2)} \underset{\text{def}}{=} \varepsilon_l^{(i)}$ - ошибка на l-м выходе скрытого слоя,

$l = 1, \ldots, L$.

Таким образом, $\varepsilon_l^{(i)}$ можно вычислить по $\tilde{\varepsilon}_q^{(i)}$, $q = 1, \ldots, K$:

$$\varepsilon_l^{(i)} = \sum_{q=1}^{K} \tilde{\varepsilon}_q^{(i)} f_q' \; w_{ql}^{(2)}.$$

То есть как бы в обратном направлении:



Нахождение производной f':

например, для сигмоидальной переходной функции:

$$f'(u) = \left(\frac{1}{1+e^{-u}}\right)' = -\frac{-e^{-u}}{\left(1+e^{-u}\right)^2} = \frac{1}{1+e^{-u}}\left(1 - \frac{1}{1+e^{-u}}\right) = f(u)\left(1 - f(u)\right).$$

Найдем частные производные по весам (как производные сложной функции):

$$\frac{\partial Q(\tilde{y}(w))}{\partial w_{ql}^{(2)}} = \frac{\partial Q(\tilde{y})}{\partial \tilde{y}_q} \frac{\partial \tilde{y}_q}{\partial w_{ql}^{(2)}} = \tilde{\varepsilon}_q^{(i)} f_q' g_l, \; q = 1,\dots,K, \; l = 1,\dots,L,$$

$$\frac{\partial Q(g(w))}{\partial w_{lj}^{(1)}} = \frac{\partial Q(g)}{\partial g_l} \frac{\partial g_l}{\partial w_{lj}^{(1)}} = \varepsilon_l^{(i)} f_l' x_j, \; l = 1,\dots,L, \; j = 0,1,\dots,n,$$

где

$$f_q' = f'\left(\sum_{l=0}^{L} w_{ql}^{(2)} \cdot g_l \right),$$

$$f_l' = f'\left(\sum_{j=0}^{n} w_{lj}^{(1)} \cdot x_j \right).$$

Алгоритм обратного распространения

1. Задать случайным образом начальные веса $w_{mk}^{h}(0)$;

2. Для $t = 1, 2, \ldots$ повторять:

3. Для всех объектов $x^{(i)}$, $i = 1, \ldots, N$ повторять:

4. Прямой ход: $g_l = f\left(\sum_{j=0}^{n} w_{l\,j}^{(1)} \cdot x_j^{(i)}\right)$, $l = 1, \ldots, L$;

$$\tilde{y}_q = f\left(\sum_{l=0}^{L} w_{ql}^{(2)} \cdot g_l\right), \; \tilde{\varepsilon}_q^{(i)} = \tilde{y}_q^{(i)} - y_q^{(i)}, \; q = 1, \ldots, K;$$

$$Q\left(w(t)\right) = \sum_{q=1}^{K} \left(\tilde{\varepsilon}_q^{(i)}\right)^2;$$

5. Обратный ход: $\varepsilon_l^{(i)} = \sum_{q=1}^{K} \tilde{\varepsilon}_q^{(i)} f_q' \, w_{ql}^{(2)}$, $l = 1, \ldots, L$;

$$w_{ql}^{(2)}(t+1) = w_{ql}^{(2)}(t) - \eta\, \tilde{\varepsilon}_q^{(i)} f_q' g_l, \; q = 1, \ldots, K, \; l = 1, \ldots, L;$$

$$w_{lj}^{(1)}(t+1) = w_{lj}^{(1)}(t) - \eta\, \varepsilon_l^{(i)} f_l' x_j^{(i)}, \quad l = 1, \ldots, L, \;\; j = 0, 1, \ldots, n.$$

6. Повторять 2)-5) пока либо критерий Q, либо веса w не стабилизируются.

Достоинства алгоритма

- градиент вычисляется за небольшое число шагов;
- алгоритм можно обобщить на произвольную функцию потерь и произвольную функцию активации;
- можно обучаться в динамике;
- можно распараллелить процесс обучения.

Недостатки алгоритма

- возможна медленная сходимость; «застревание» в локальном минимуме;
- зависимость от порядка объектов;
- проблема переобучения;
- непонятно, как задавать архитектуру сети, параметры алгоритма обучения;
- персептрон – «черный ящик»!