

ФГБОУ ВО «Липецкий государственный  
технический университет»

Кафедра АСУ

## **Системы искусственного интеллекта**

### *Лекция 5* **НЕЙРОННЫЕ СЕТИ ПРЯМОГО РАСПРОСТРАНЕНИЯ**

**д.т.н. Сараев Павел Викторович**

Липецк - 2024

# ПЛАН ЛЕКЦИИ

## Нейронные сети

1. Основы теории нейронных сетей
2. Нейронные сети прямого распространения
  - 2.1. Структура
  - 2.2. Методика применения
  - 2.3. Постановка задачи обучения
  - 2.4. Процедура обратного распространения ошибки
3. Самоорганизующиеся карты Кохонена
  - 3.1. Кластеризация данных
  - 3.2. Обучение без учителя

# Нейронные сети

## 1. Основы теории нейронных сетей

**1943** г. - МакКаллок (McCulloch) и Питтс (Pitts) «Логическое исчисление идей, относящихся к нервной деятельности»

**1949** г. - Хебб (Hebb) «Организация поведения» - соединение нейронов усиливает их возможность решать логические задачи

**1957** г. — Розеблатт — первый компьютер-нейронная сеть — перцептрон — обработка логических задач — линейная классификация

**1969** г. - Минский (Minsky) и Пейперт (Papert) «Перцептроны»: показана ограниченность возможностей НС (XOR-problem)

**1982** г. - Хопфилд (Hopfield): работы по математическим основам динамики НС

**1984** г. - Кохонен (Kohonen): сети, обучающиеся без учителя

**1986 г.** - Румельхарт (Rumelhart) и МакКлеланд (McClelland): алгоритм обратного распространения ошибки для обучения многослойных НС

**1989 г.** – ЛеКун ( LeCun) - сверточные нейронные сети, распознавание рукописных индексов на конвертах (zip-кодов)

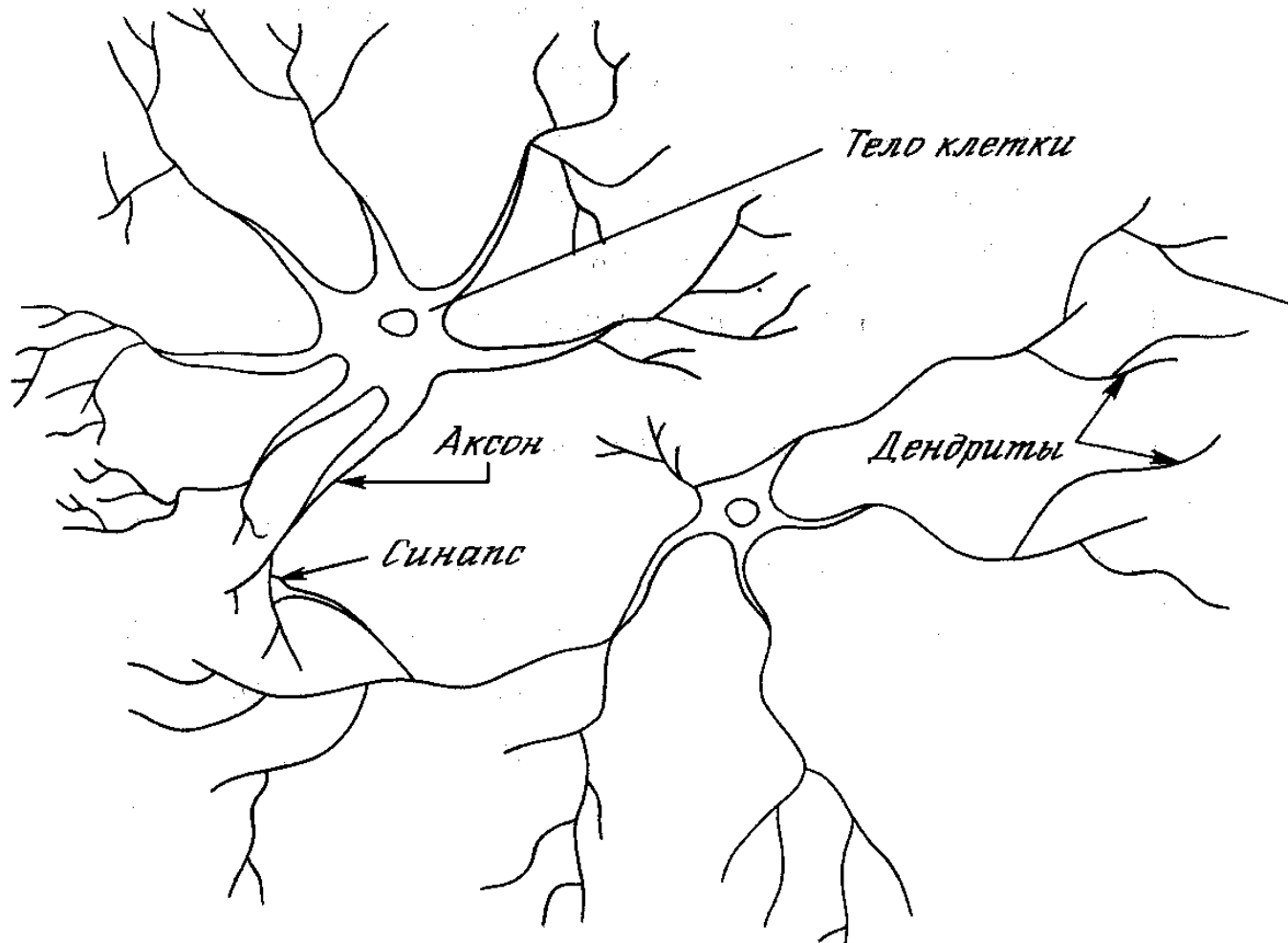
**1998 г.** – ЛеКун ( LeCun) – распознавание чеков

**2006 г.** – Хинтон (Hinton) – алгоритм обучения глубоких слоев в сетях – «умный» выбор начальных значений весов

**2010 г.** – Глорот (Glorot) и Бенджио (Bengio) – анализ сложности обучения глубоких НС, критика дифференцируемых функций активации

**2012 г.** – Хинтон (Hinton) – глубокие сверточные НС (deep convolutional NN) - отличные результаты в распознавании образов (ImageNet) – 15,3 % ошибки (26,2 % - ближайший конкурент)

## Биологический нейрон (электрохимическая модель)



## Искусственный нейрон

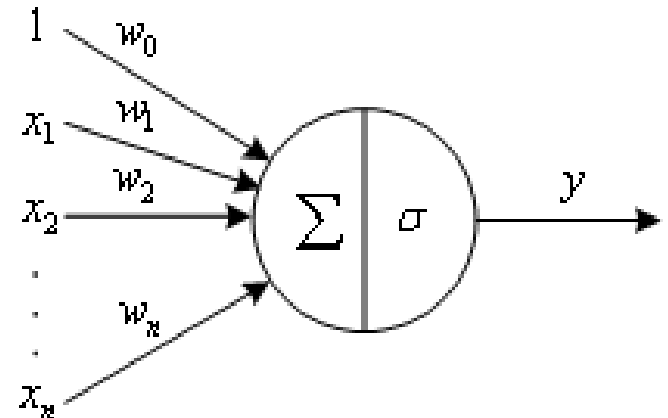
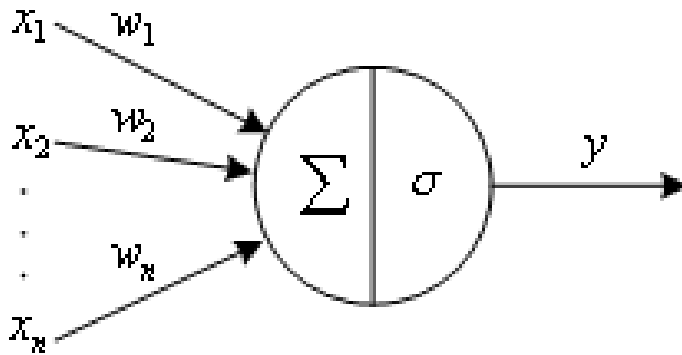
вход  $x$  – вектор, выход  $y$  – скаляр

1) уровень активности  $net$

$$net = \sum_{i=0}^n w_i x_i = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

2) функция активации  $\sigma$

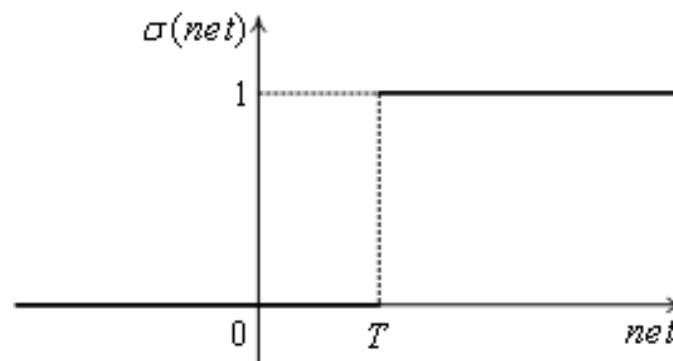
$$y = \sigma(net)$$



Функции активации:

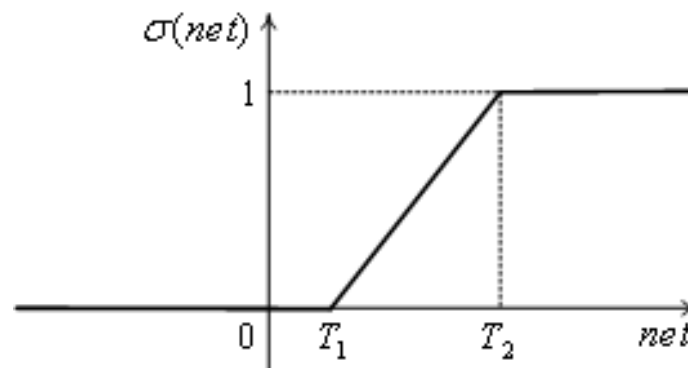
1) пороговая

$$\sigma(net) = \begin{cases} 1, & net \geq T \\ 0, & net < T \end{cases}$$



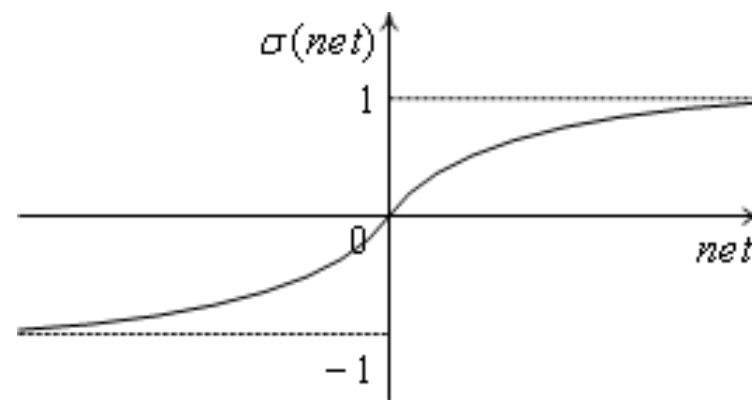
2) линейная пороговая

$$\sigma(net) = \begin{cases} 1, & net \geq T_2 \\ \frac{net - T_1}{T_2 - T_1} & T_1 < net < T_2 \\ 0, & net \leq T_1 \end{cases}$$



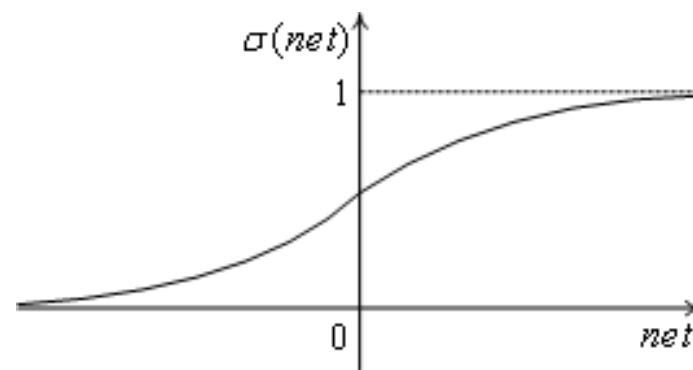
### 3) гиперболический тангенс

$$\sigma(net) = \text{th}(net) = \frac{e^{net} - e^{-net}}{e^{net} + e^{-net}}$$



### 4) сигмоидная логистическая

$$\sigma(net) = \frac{1}{1 + e^{-net}}$$



## Перцептрон: функция активации – пороговая

$$\sigma(net) = \begin{cases} 1, & w_1x_1 + w_2x_2 + \dots + w_nx_n \geq T \\ 0, & w_1x_1 + w_2x_2 + \dots + w_nx_n < T \end{cases}$$

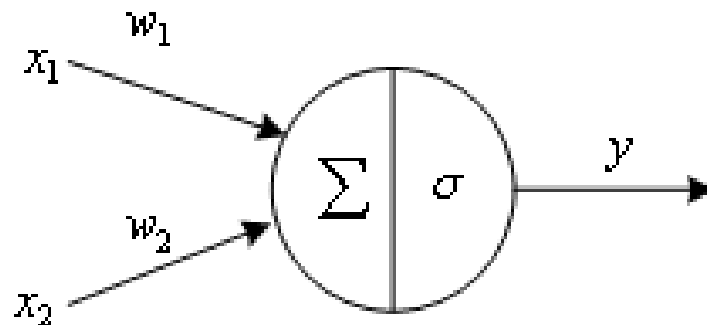
*Пример.*

Булева функция

$$y = f(x_1, x_2) = x_1 \vee x_2$$

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	1

**Задача:** определить веса перцептрона  $w_1$ ,  $w_2$ , порог  $T$ , чтобы выдавать правильный ответ для всех входных значений



$$\sigma(net) = \begin{cases} 1, & w_1x_1 + w_2x_2 \geq T \\ 0, & w_1x_1 + w_2x_2 < T \end{cases}$$

Бесконечное множество решений.

$$w_1 = w_2 = 0,5; \quad T = 0,4$$

Пусть  $x_1=1, x_2=0$ :

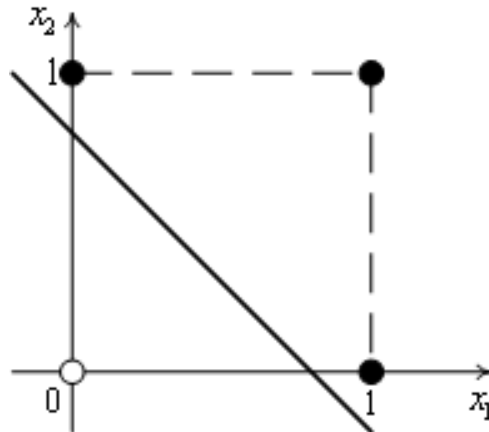
$$net=0,5*1 + 0,5*0=0,5$$

$$net=0,5 \geq T=0,4$$

$$y = 1$$

Две полуплоскости – граница:

$$0,5x_1 + 0,5x_2 = 0,4$$



Пример



Унификация записи:

$$w_1x_1 + w_2x_2 + \dots + w_nx_n \geq T$$

$$-1 \cdot T + w_1x_1 + w_2x_2 + \dots + w_nx_n = \sum_{i=0}^n w_ix_i = \langle w, x \rangle \geq 0$$

$$x_0 = 1 \quad w_0 = -T$$

Задача – **линейно разделимая**, если существуют веса  $w_i$ , т.ч.

$$\langle w, x \rangle \geq 0 \quad \forall x \in C_1$$

$$\langle w, x \rangle < 0 \quad \forall x \in C_2$$

Для линейно разделимых задач применим алгоритм обучения.

**Алгоритм** (обучение перцептрона):

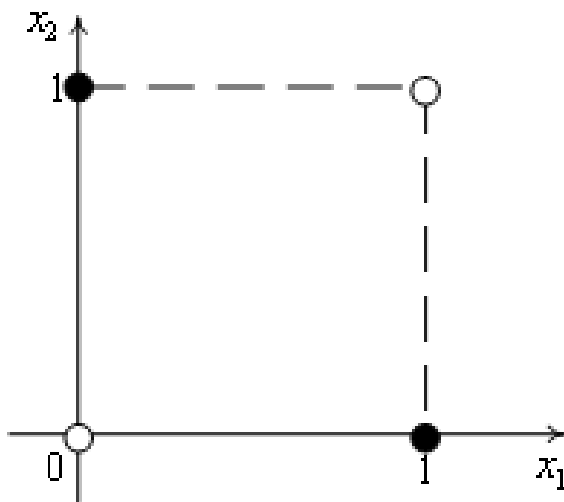
1. Задание шага обучения  $\eta > 0$ ,  $k$  – количество строк в таблице
2. Инициализация весов  $w_i$  небольшими случайными значениями.  
Ошибка  $E := 0$   
Номер строки таблицы:  $m := 1$
3. Начало обучения.  $x^* := x_m, y^* := y_m$   
Выход перцептрона:  $y = y(x^*)$
4. Обновление весов  $w := w - \eta(y^* - y)x^*$
5. Накопление ошибки  $E := E + \|y^* - y\|^2$
6. Если  $m < k$ , то  $m := m + 1$   
и переход на шаг 3.
7. Если  $E = 0$ , то прекращение вычислений (решение найдено),  
иначе  $E := 0, m := 1$   
и переход на шаг 3.

Существуют линейно неразделимые задачи

Проблема «исключающего или» (*XOR-problem*) (Минский и Пейперт):

$$y = f(x_1, x_2) = x_1 \oplus x_2$$

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0



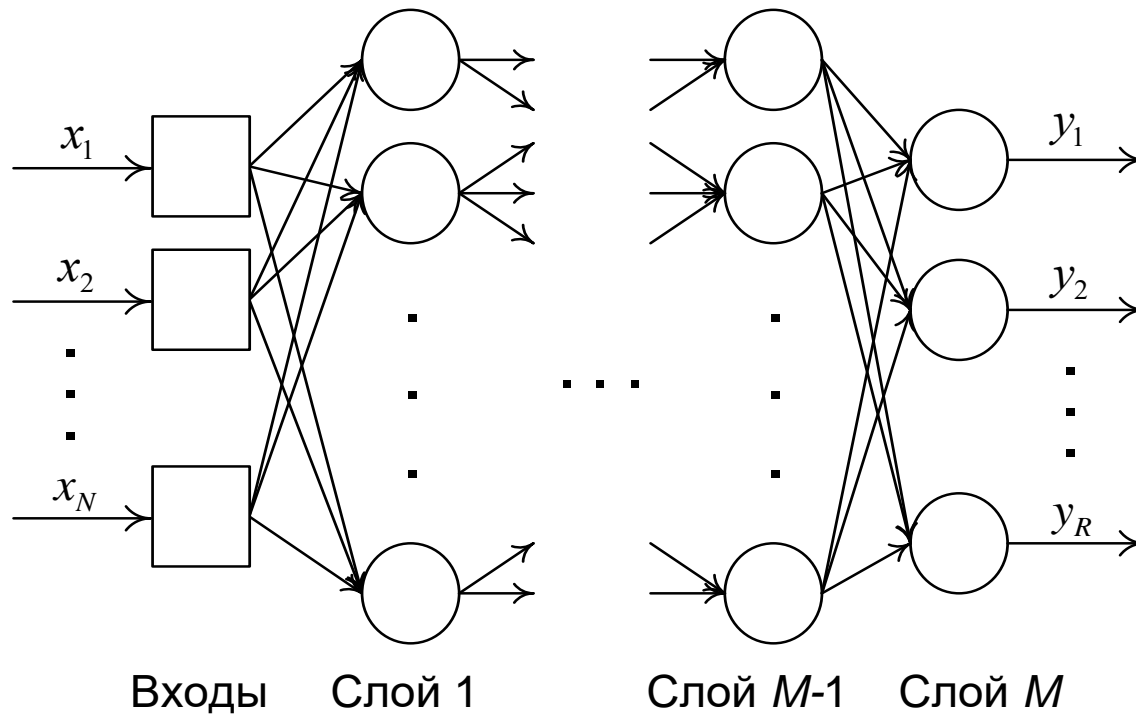
## 2. Нейронные сети прямого распространения (НС ПР)

### 2.1. Структура

Входной слой

Скрытые (промежуточные) слои

Выходной слой



Выход  $q$ -го нейрона  $m$ -го слоя:

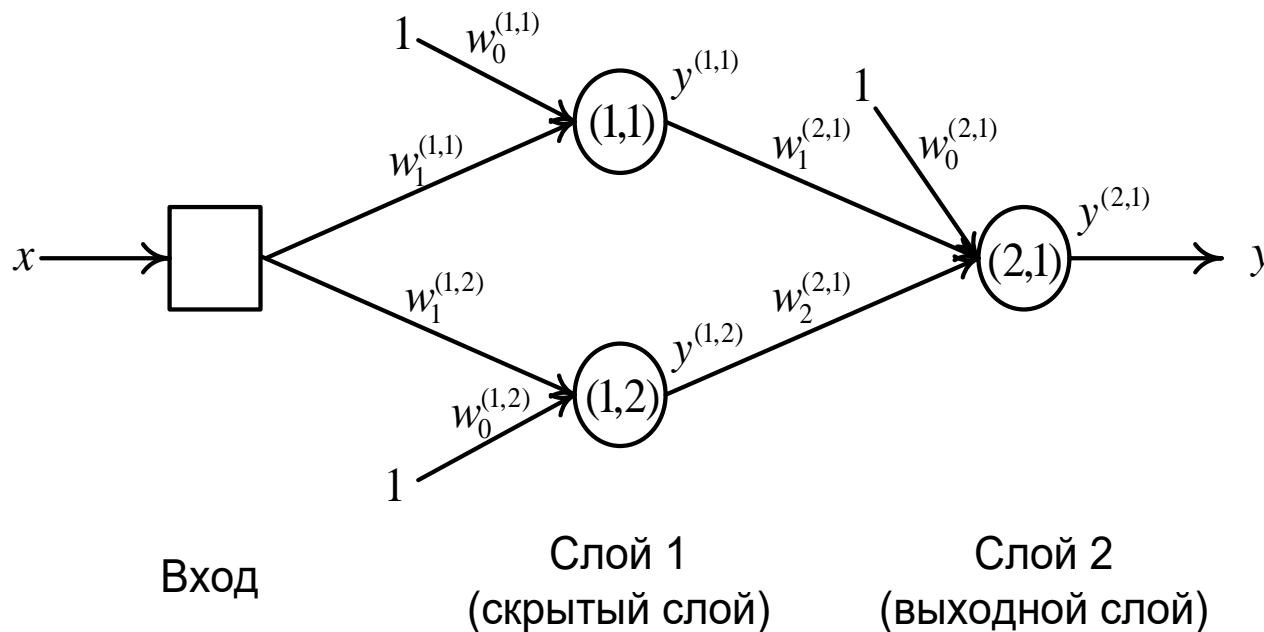
$$y^{(m,q)} = \sigma(\text{net}^{(m,q)}) = \sigma\left(\sum_{i=0}^n w_i^{(m,q)} y^{(m-1,i)}\right)$$

Выход нейронной сети:

$$y(x; w) = y^{(M)} = W^{(M)} \Phi^{(M-1)}(W^{(M-1)} \Phi^{(M-2)}(\dots \Phi^{(1)}(W^{(1)} x) \dots))$$

суперпозиционная линейно-нелинейная структура

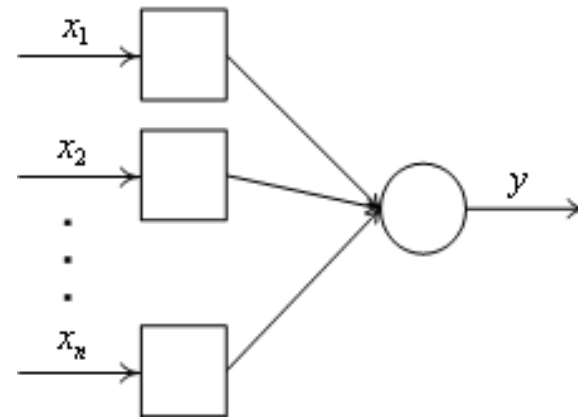
*Пример.*



*Пример*

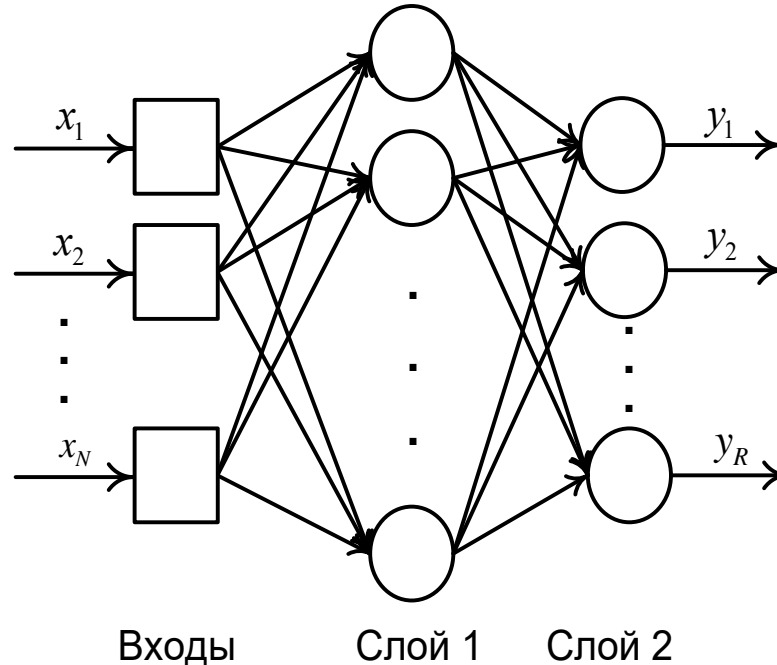
Простейшая НС ПР:

$$y = w_0 + \sum_{j=1}^N w_j x_j$$



Двухслойная (стандартная) НС ПР:

$$y = \sum_{i=1}^q w_i \sigma \left( \sum_{j=1}^N w_{ij} x_j \right)$$



### Теорема (Фунахаши) об универсальных аппроксимационных способностях НС ПР.

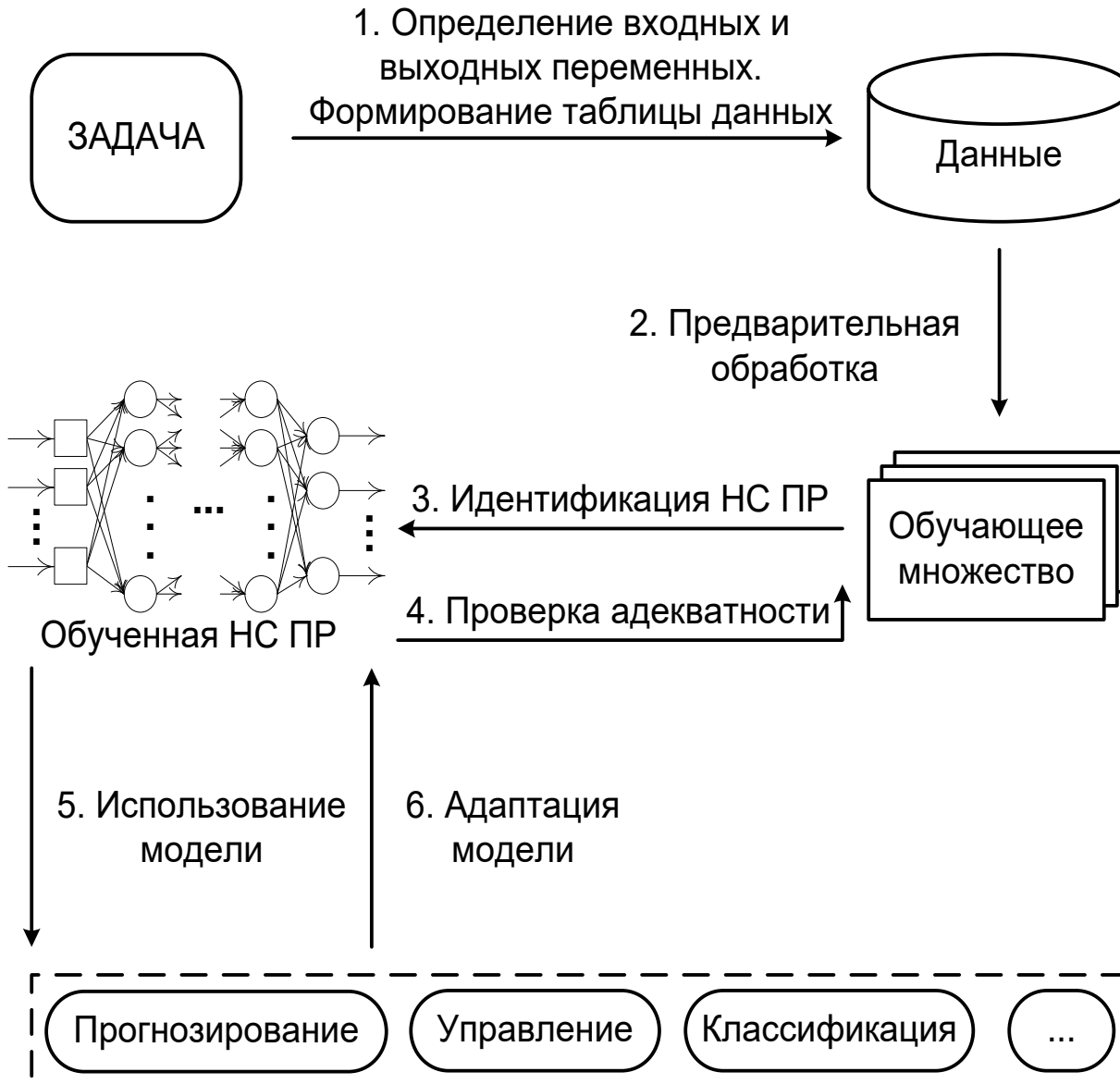
Пусть  $f(x_1, x_2, \dots, x_N)$  - непрерывная функция, определенная на компактном множестве, и  $\varepsilon > 0$  – точность аппроксимации. Существует такое натуральное число  $q$  и набор действительных чисел  $w_{ij}, w_i$ , что функция

$$y = \sum_{i=1}^q w_i \sigma \left( \sum_{j=1}^N w_{ij} x_j \right)$$

где  $\sigma$  - непостоянная ограниченная монотонно возрастающая непрерывная функция (например, сигмоидная логистическая), приближает исходную функцию с погрешностью, не превышающей  $\varepsilon$  на всей области определения, т.е.

$$\sup_{(x_1, x_2, \dots, x_N) \in R^n} \left| f(x_1, x_2, \dots, x_N) - f^*(x_1, x_2, \dots, x_N) \right| \leq \varepsilon$$

## 2.2. Методика применения



Этап 1. Информативное множество входных  $x$  и выходных  $y$  величин выбирается, исходя из поставленной задачи.

Этап 2.

- Преобразование данных, заданных в качественном, к числовому виду.
- Восстановление отсутствующих данных в обучающем множестве (восстановление пробелов).
- Удаление неестественных, ошибочных, сильно искаженных данных, не отражающих реального поведения объекта.
- Переход от абсолютных значений к относительным.
- Нормировка значений переменных в диапазон  $[0;1]$  или  $[-1;1]$ .

Этап 3. Идентификация (построение) нейросетевой модели - центральный этап.

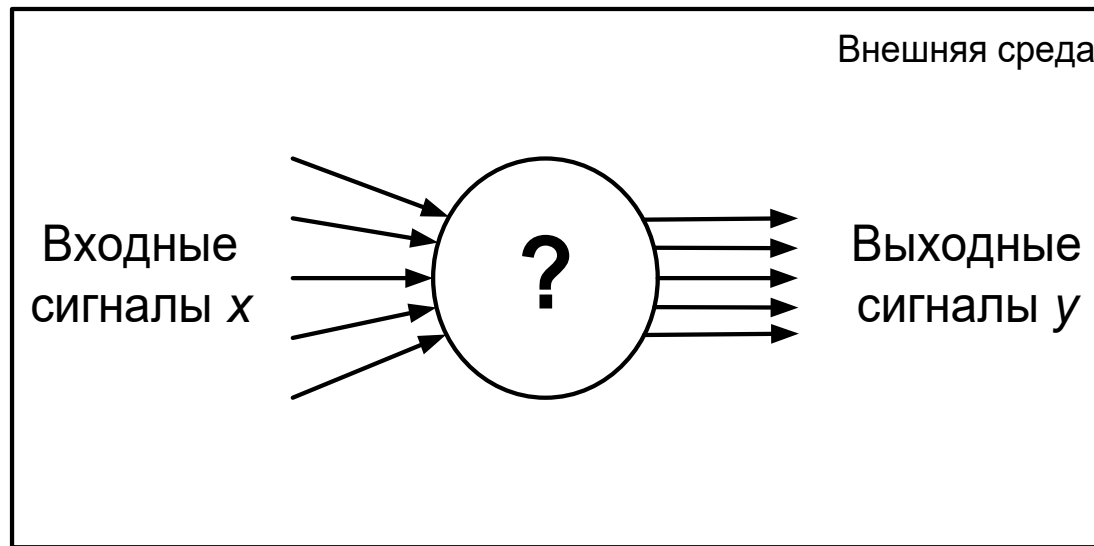
Этап 4. Основной подход: выделение из исходной таблицы вход-выходных данных тестового множества (около 20%). Остальная часть - обучающее множество. Тестовое множество состоит из данных, не использующихся в процессе обучения. Более адекватна модель, которая показывает меньшую ошибку на тестовом множестве.

Этап 5.

- Прогнозирование - получение выходных величин для заданных входных значений.
- Управление - формирование управляющих сигналов с целью получения желаемых выходов управляемой системы.
- Классификация - соотнесение входного вектора, описывающего объект, к определенному классу объектов.

Этап 6. При использовании модели с течением времени встает необходимость учета новой информации о предметной области, рассматриваемой в задаче, например, связанная с поступлением новой вход-выходной информации.

## 2.3. Постановка задачи обучения



Обучающее множество:

$$\begin{bmatrix}
 \tilde{x}_{11} & \tilde{x}_{12} & \dots & \tilde{x}_{1N} & \tilde{y}_{11} & \tilde{y}_{12} & \dots & \tilde{y}_{1R} \\
 \tilde{x}_{21} & \tilde{x}_{22} & \dots & \tilde{x}_{2N} & \tilde{y}_{21} & \tilde{y}_{22} & \dots & \tilde{y}_{2R} \\
 \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
 \tilde{x}_{K1} & \tilde{x}_{K2} & \dots & \tilde{x}_{KN} & \tilde{y}_{K1} & \tilde{y}_{K2} & \dots & \tilde{y}_{KR}
 \end{bmatrix}
 \begin{matrix}
 \text{1-й пример} \\
 \text{2-й пример} \\
 \\
 \text{K-й пример}
 \end{matrix}$$

указания учителя

Построение НС ПР  $y=f(w, x)$ :

- **определение структуры сети** (количества скрытых слоев и нейронов в каждом слое);
- **обучение** (определение для выбранной фиксированной структуры НС ПР значений весов  $w$ ).

Критерий соответствия НС ПР обучающему множеству - квадратичный функционал качества обучения:

$$Q(w) = \sum_{k=1}^K \sum_{r=1}^R Q_{kr}(w) = \sum_{k=1}^K \sum_{r=1}^R (y_r(w, x^{(k)}) - \tilde{y}_{kr})^2$$

нелинейная задача о наименьших квадратах (НЗНК).

Цель – оптимизация (минимизация) функционала по вектору весов НС ПР  $w$ .

Обучение НС ПР – обучение с **УЧИТЕЛЕМ**

Оптимизация (минимизация) функции одной переменной

$$y = f(x)$$

Необходимое условие оптимума (НУО):

$$y' = f'(x) = 0$$

Достаточное условие оптимума (ДУО):

$$y'' = f''(x) > 0 \text{ (т. } \min)$$

$$y'' = f''(x) < 0 \text{ (т. } \max)$$

*Пример.*

$$y = x^2$$

НУО:

$$y' = 2x = 0 \Rightarrow x^* = 0$$

ДУО:

$$y'' = 2 > 0 \Rightarrow x^* = 0 - \text{т. } \min$$

*Пример*



Оптимизация (минимизация) функции нескольких переменных

$$y = f(x_1, x_2, \dots, x_n)$$

Необходимое условие оптимума (НУО):

$$\nabla f(x) = \text{grad } f(x) = \begin{bmatrix} f'_{x_1} \\ f'_{x_2} \\ \vdots \\ f'_{x_n} \end{bmatrix} = \begin{bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \\ \vdots \\ \partial f / \partial x_n \end{bmatrix} = 0$$

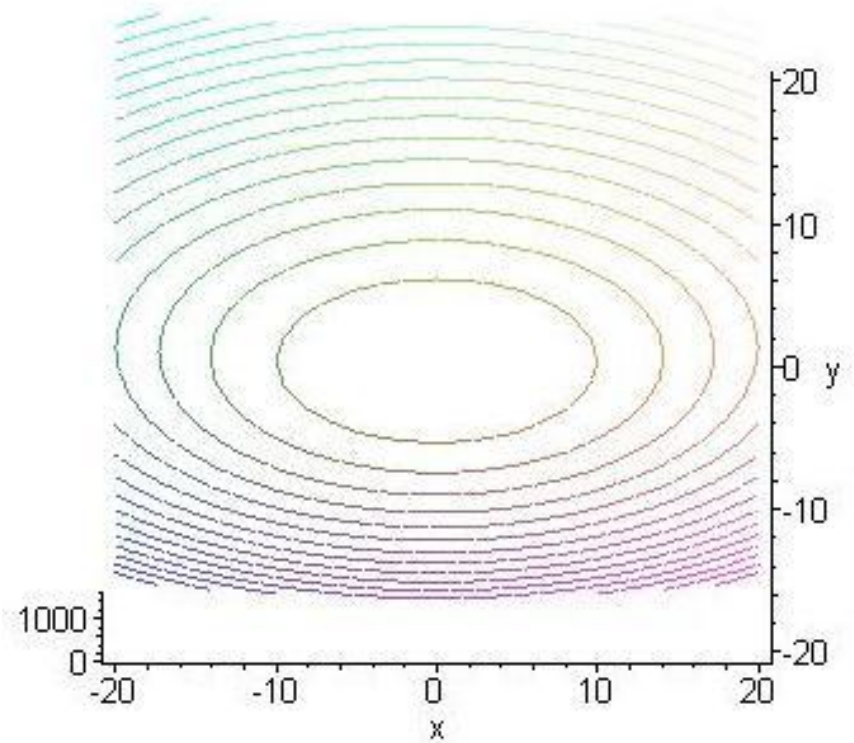
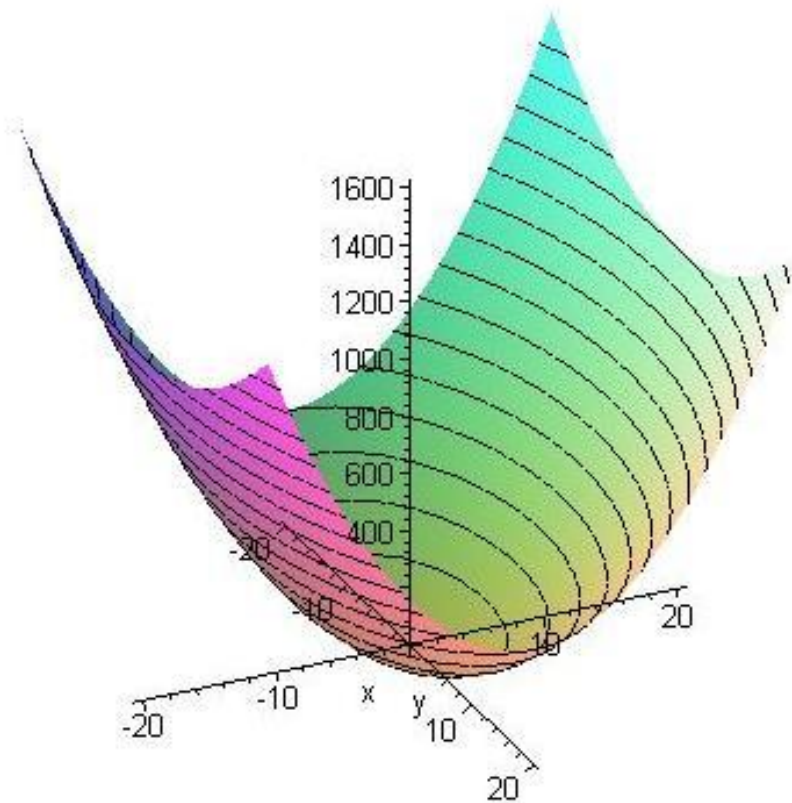
Достаточное условие оптимума (ДУО):

$$\nabla^2 f(x) = \text{Hessian } f(x) = \begin{bmatrix} \partial^2 f / \partial x_1^2 & \partial^2 f / \partial x_1 \partial x_2 & \cdots & \partial^2 f / \partial x_1 \partial x_n \\ \partial^2 f / \partial x_2 \partial x_1 & \partial^2 f / \partial x_2^2 & \cdots & \partial^2 f / \partial x_2 \partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial^2 f / \partial x_n \partial x_1 & \partial^2 f / \partial x_n \partial x_2 & \cdots & \partial^2 f / \partial x_n^2 \end{bmatrix}$$

- положительно определен – т. min;
- отрицательно определен – т. max

Пример.

$$y = x_1^2 + 3x_2^2$$



$$y = x_1^2 + 3x_2^2$$

НУО:

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 6x_2 \end{bmatrix} = 0$$

$$x_1^* = 0, x_2^* = 0$$

ДУО:

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} \bigg|_{x=0, y=0} = \begin{bmatrix} 2 & 0 \\ 0 & 6 \end{bmatrix} \bigg|_{x=0, y=0} = \begin{bmatrix} 2 & 0 \\ 0 & 6 \end{bmatrix}$$

Матрица положительно определена (по критерию Сильвестра:  
 $2 > 0, 2 * 6 - 0 * 0 = 12 > 0$ ) =>

$$x_1^* = 0, x_2^* = 0$$

- т. min

Пример

$$f(x) \rightarrow \min$$

< - >

$$f'(x) = g(x) = 0$$

Численные методы решения уравнений (метод Ньютона, ...).

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}, k = 1, 2, \dots$$

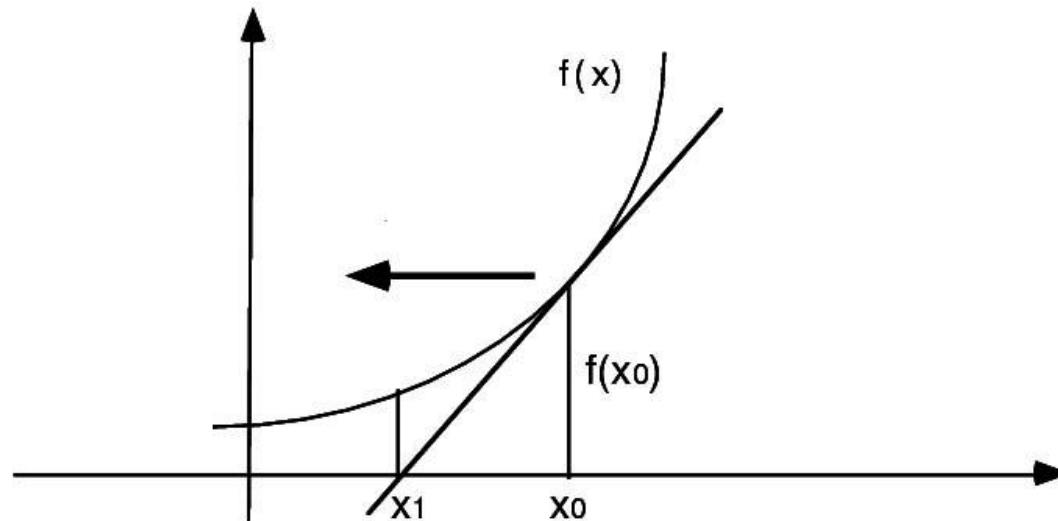

---

$$x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)}, k = 1, 2, \dots$$



Типичный метод минимизации, идея которого: переход к  $x_{k+1}$ :

$$f(x_{k+1}) < f(x_k), k = 1, 2, \dots$$



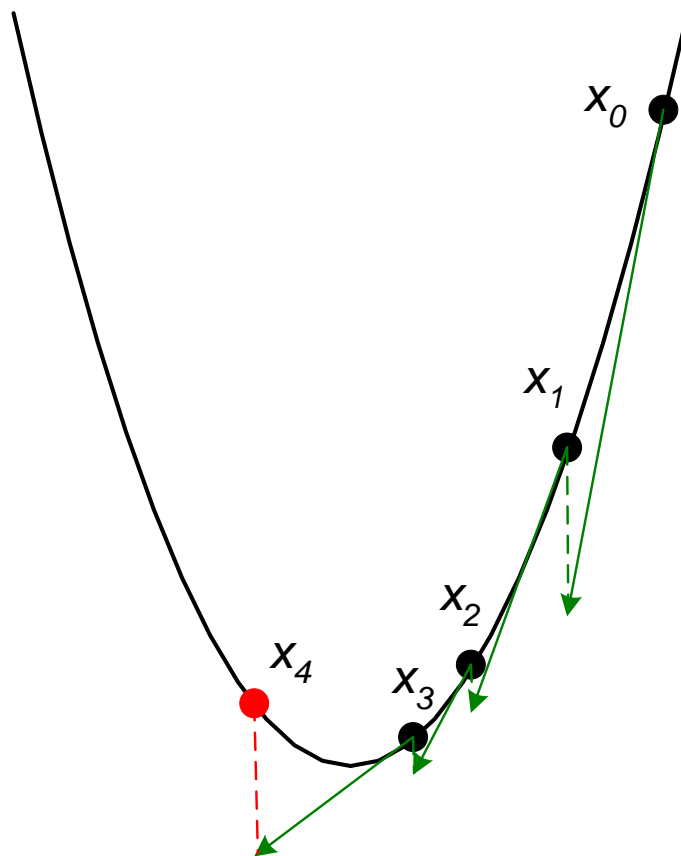
Метод наискорейшего спуска (производная – направление возрастания функции):

$$x_{k+1} = x_k - \eta f'(x_k), \eta > 0$$

где  $\eta$  – шаг оптимизации – достаточно мал.

В частности:

$$\eta = \frac{1}{f''(x_k)}, f''(x_k) > 0$$



Функция нескольких переменных: направление возрастания функции - **градиент**. Метод наискорейшего спуска:

$$x_{k+1} = x_k - \eta \nabla f(x_k), \eta > 0$$

**Критерий останова:**

$$|f(x_{k+1}) - f(x_k)| < \varepsilon$$

где  $\varepsilon > 0$  — малое число — точность.

## 2.4. Процедура обратного распространения ошибки (ОРО)

Процедура обратного распространения ошибки (ОРО, *error backpropagation*) – эффективный метод вычисления градиента функционала качества обучения НС ПР по вектору весов, учитывающий суперпозиционную структуру.

$$Q(w) = \sum_{k=1}^K \sum_{r=1}^R Q_{kr}(w) = \sum_{k=1}^K \sum_{r=1}^R \left( y_r(w, x^{(k)}) - \tilde{y}_{kr} \right)^2$$

Одновыходная НС ПР ( $R = 1$ ):

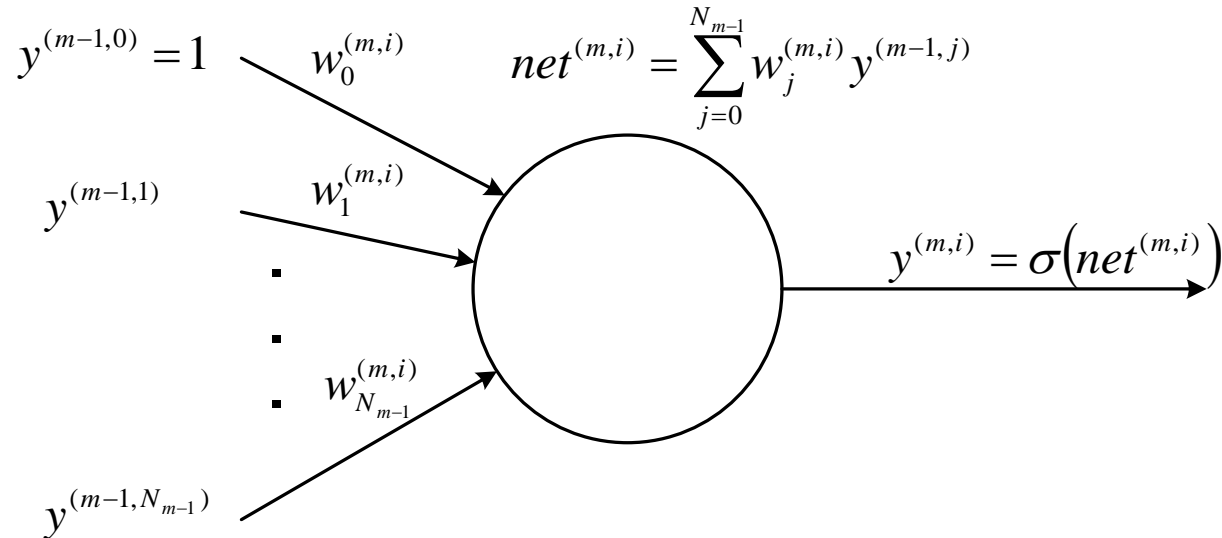
$$Q(w) = \sum_{k=1}^K Q_k(w) = \sum_{k=1}^K \left( y(w, x^{(k)}) - \tilde{y}_k \right)^2$$
$$\nabla_w Q(w) = \nabla_w \left( \sum_{k=1}^K Q_k(w) \right) = \sum_{k=1}^K \nabla_w Q_k(w)$$

Производная суперпозиции функций:

$$[f(g(x))]' = f'_g(g(x)) \cdot g'(x)$$
$$[f(g_1(x), g_2(x), \dots, g_n(x))]' = \sum_{i=1}^n \frac{\partial f(\cdot)}{\partial g_i(x)} \cdot g'_i(x)$$

$$\nabla_w Q_k(w) = \begin{bmatrix} \vdots \\ \partial Q_k(w) / \partial w_j^{(m,i)} \\ \vdots \end{bmatrix}$$

$i$ -й нейрон  $m$ -го слоя:



Используем правило дифференцирования суперпозиции функций:

$$\frac{\partial Q_k(w)}{\partial w_j^{(m,i)}} = \frac{\partial Q_k(w)}{\partial y^{(m,i)}} \cdot \frac{\partial y^{(m,i)}}{\partial net^{(m,i)}} \cdot \frac{\partial net^{(m,i)}}{\partial w_j^{(m,i)}}$$

Второй множитель:

$$\frac{\partial y^{(m,i)}}{\partial net^{(m,i)}} = \sigma'_{net}(net^{(m,i)})$$
$$\sigma(net) = \frac{1}{1 + e^{-net}} \quad \rightarrow \quad \underline{\sigma'_{net}(net) = \sigma(net) \cdot (1 - \sigma(net))}$$

Третий множитель:

$$\frac{\partial net^{(m,i)}}{\partial w_j^{(m,i)}} = y^{(m-1,j)}$$

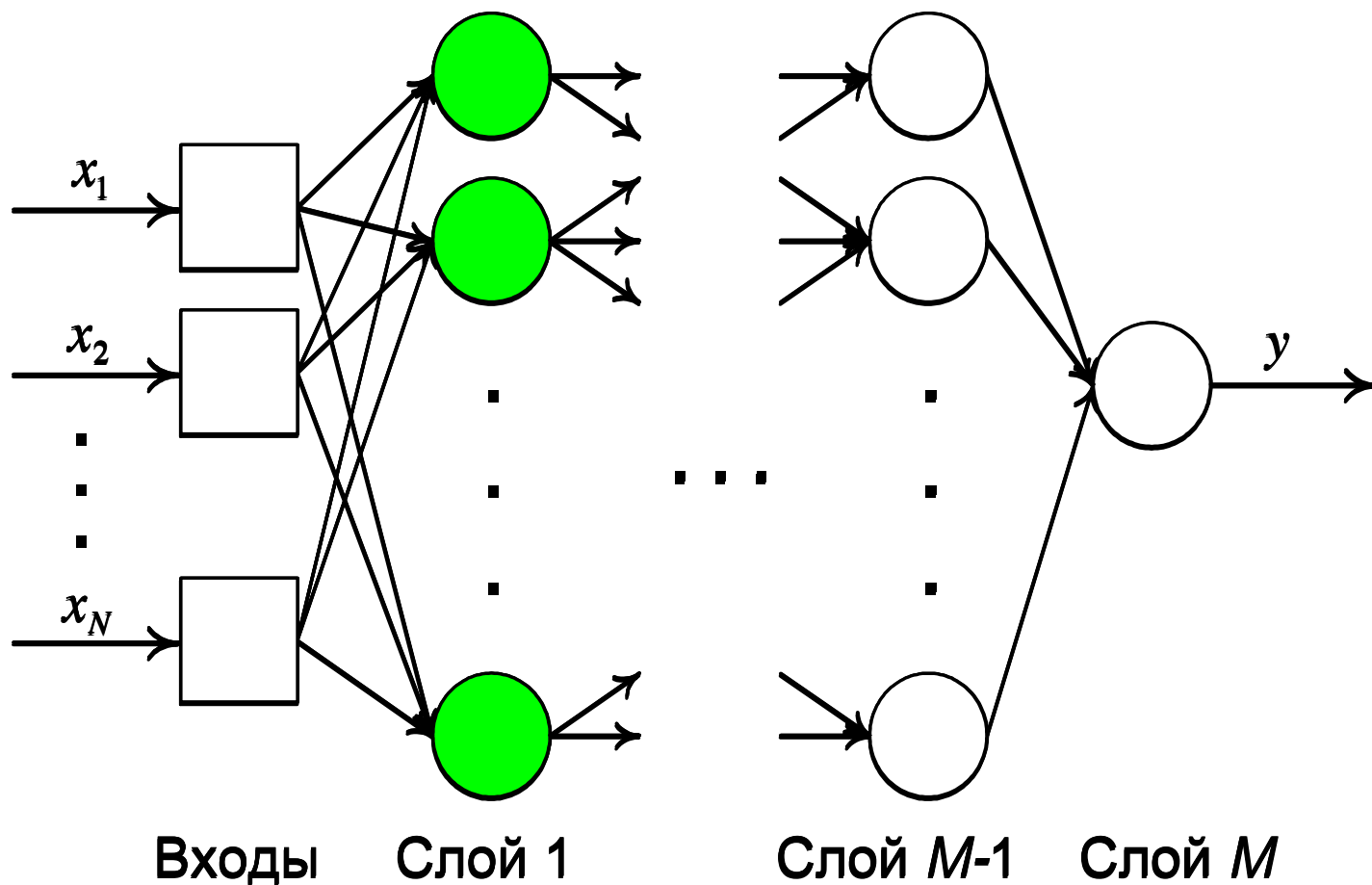
Первый множитель - рекуррентная процедура:

$$s^{(m,i)} = \frac{\partial Q_k(w)}{\partial y^{(m,i)}} = \sum_{j=1}^{N_{m+1}} \frac{\partial Q_k(w)}{\partial y^{(m+1,j)}} \cdot \frac{\partial y^{(m+1,j)}}{\partial y^{(m,i)}} = \sum_{j=1}^{N_{m+1}} s^{(m+1,j)} \cdot \sigma'_{net}(net^{(m+1,j)}) \cdot w_i^{(m+1,j)}$$

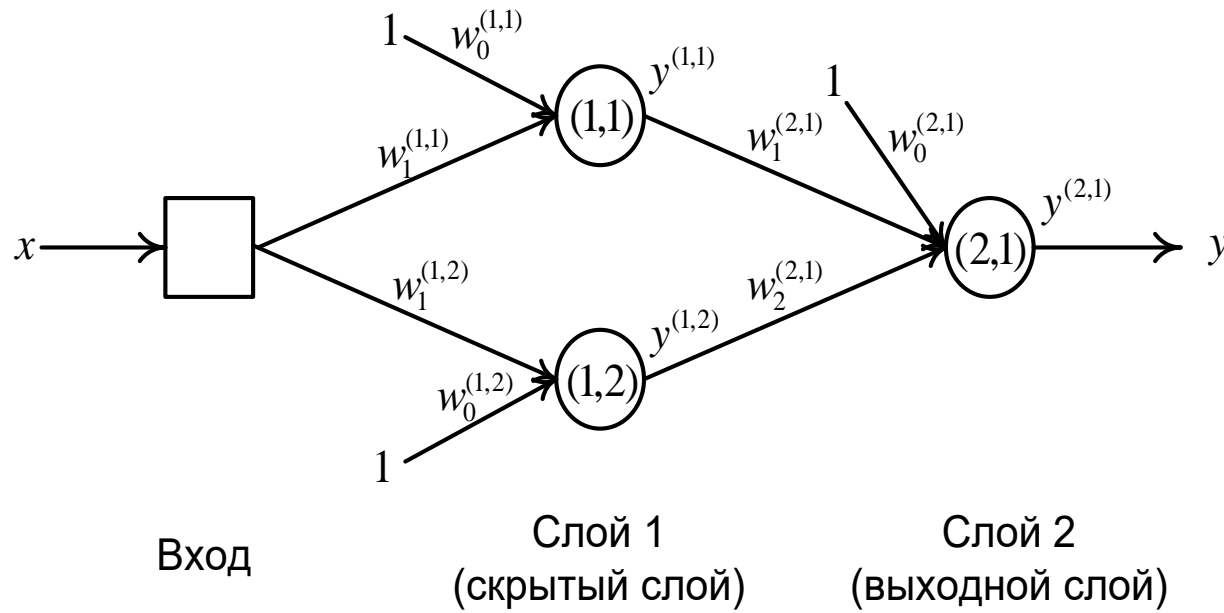
начальное условие:

$$s^{(M,1)} = \frac{\partial Q_k(w)}{\partial y^{(M,1)}} = \varepsilon(w) = y(w) - \tilde{y}$$

Послойное вычисление градиента (в обратном направлении):

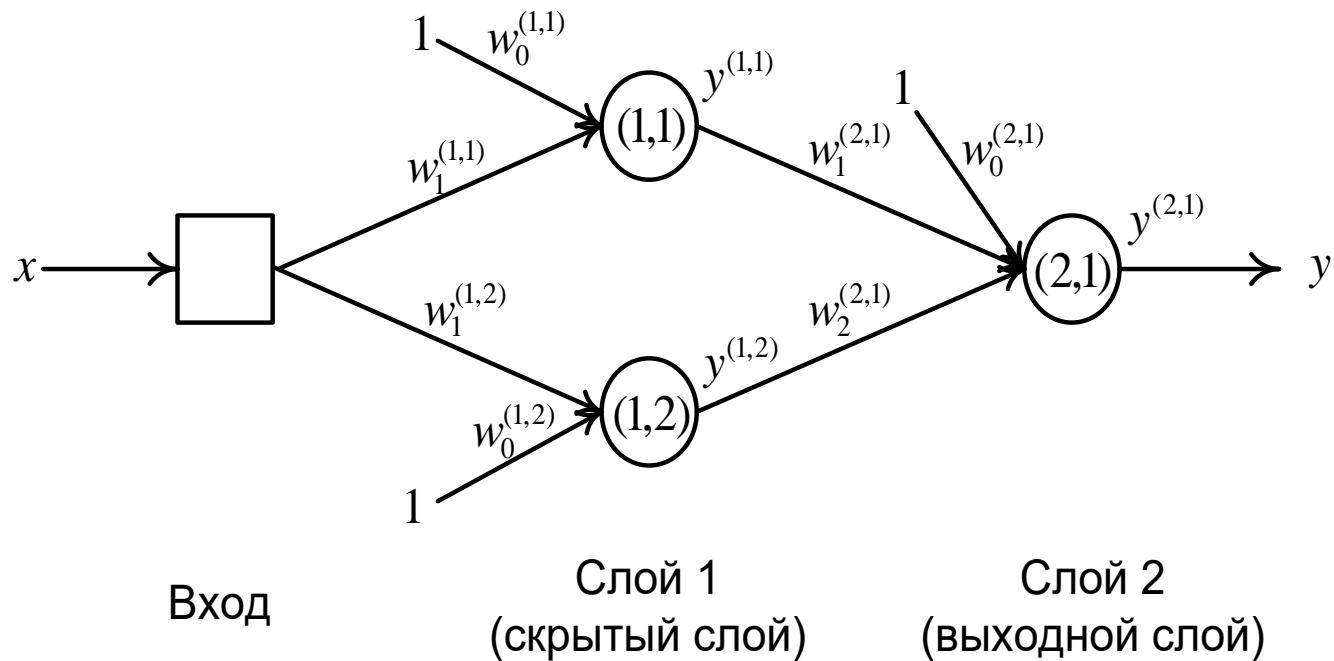


Пример.



$$w = \begin{bmatrix} w_0^{(1,1)} \\ w_1^{(1,1)} \\ w_0^{(1,2)} \\ w_1^{(1,2)} \\ w_0^{(2,1)} \\ w_1^{(2,1)} \\ w_2^{(2,1)} \end{bmatrix}$$

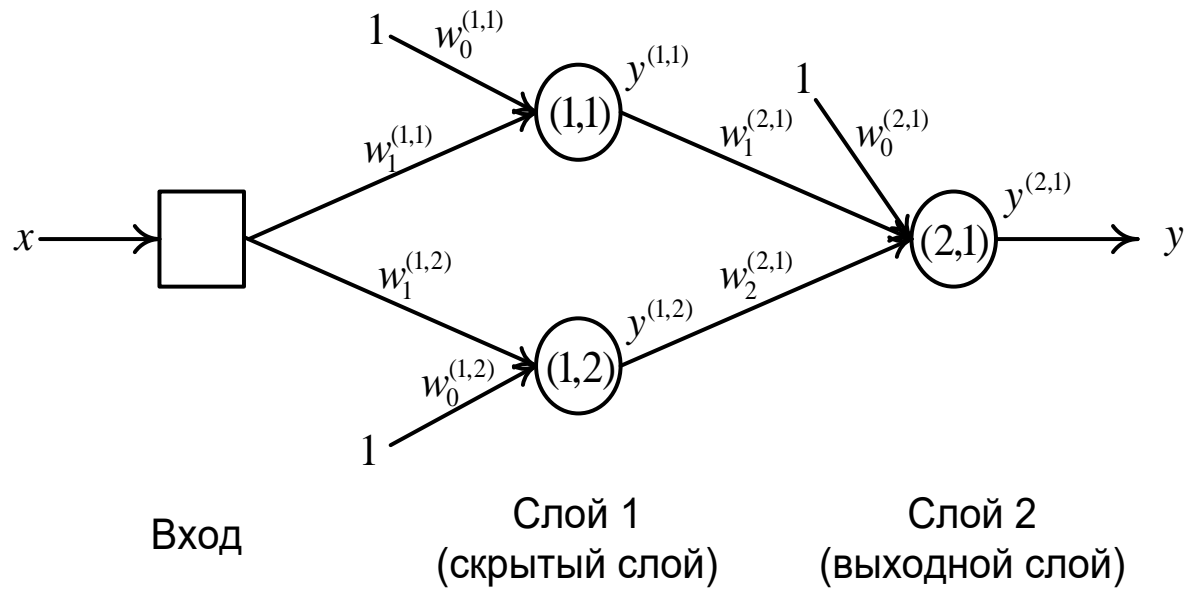
$$y^{(2,1)} = w_0^{(2,1)} + w_1^{(2,1)} y^{(1,1)} + w_2^{(2,1)} y^{(1,2)}$$



$$s^{(1,1)} = s^{(2,1)} \frac{\partial y^{(2,1)}}{\partial y^{(1,1)}} = s^{(2,1)} w_1^{(2,1)}$$

$$s^{(1,2)} = s^{(2,1)} \frac{\partial y^{(2,1)}}{\partial y^{(1,2)}} = s^{(2,1)} w_2^{(2,1)}$$

$$s^{(2,1)} = \frac{\partial Q(w)}{\partial y^{(2,1)}} = y(w) - \tilde{y}$$



$$\frac{\partial Q(w)}{\partial w_0^{(1,1)}} = s^{(1,1)} \cdot \frac{\partial y^{(1,i)}}{\partial net^{(1,i)}} \cdot \frac{\partial net^{(1,1)}}{\partial w_0^{(1,1)}} =$$

$$\nabla_w Q(w) = \begin{bmatrix} s^{(1,1)} y^{(1,1)} (1 - y^{(1,1)}) \\ s^{(1,1)} y^{(1,1)} (1 - y^{(1,1)}) x \\ s^{(1,2)} y^{(1,2)} (1 - y^{(1,2)}) \\ s^{(1,2)} y^{(1,2)} (1 - y^{(1,2)}) x \\ s^{(2,1)} y^{(2,1)} (1 - y^{(2,1)}) \\ s^{(2,1)} y^{(2,1)} (1 - y^{(2,1)}) y^{(1,1)} \\ s^{(2,1)} y^{(2,1)} (1 - y^{(2,1)}) y^{(1,2)} \end{bmatrix}$$

### 3. Самоорганизующиеся карты Кохонена

#### 3.1. Кластеризация

Самоорганизующиеся карты (СОК, *self-organizing maps*, *SOM*) Кохонена (1984) - специфический класс НС, применяемых в задачах **кластеризации (таксономии)**.

#### **Исходные данные:**

множество объектов, представленных своими векторами признаков

$$\{x^p \in R^N, p=1, \dots, P\}$$

$$\begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^P \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_N^1 \\ x_1^2 & x_2^2 & \dots & x_N^2 \\ \vdots & \vdots & & \vdots \\ x_1^P & x_2^P & \dots & x_N^P \end{bmatrix}$$

#### **Цель:**

компактно описать информацию, заложенную в исходных данных

#### **Задача:**

разбить на классы - **кластеры (таксоны)** – чтобы объекты в пределах одного кластера были эквивалентными.

В каждом кластере из  $M$  типичный представитель -  $w^m$  - **ядро** кластера.

Оценка степени близости объектов: расстояние (его квадрат)

$$d(x, y) = \|x - y\|^2 = \sum_{i=1}^N (x_i - y_i)^2$$

Иногда:

$$d(x, y) = \sum_{i=1}^N |x_i - y_i|$$

При заданном числе кластеров  $M$  ставится задача кластеризации: найти  $M$  ядер и разбить объекты на кластеры так, чтобы

$$D = \sum_{p=1}^P d(x^p, w^m(x^p)) \rightarrow \min$$

где

$$w^m(x^p)$$

- ядро кластера, к которому принадлежит данный вектор

Близость объекта  $x$  ядру кластера  $w^m$ :

$$d(x, w) = \|x - w^m\|^2 = \langle x, x \rangle - 2\langle x, w^m \rangle + \langle w^m, w^m \rangle$$

$$d(x, w) = \|x - w^m\|^2 = \|x\|^2 - 2\langle x, w^m \rangle + \|w^m\|^2$$

Пусть  $\|w^m\| = 1$

$$d(x, w) = \|x\|^2 - 2\langle x, w^m \rangle + 1$$

---

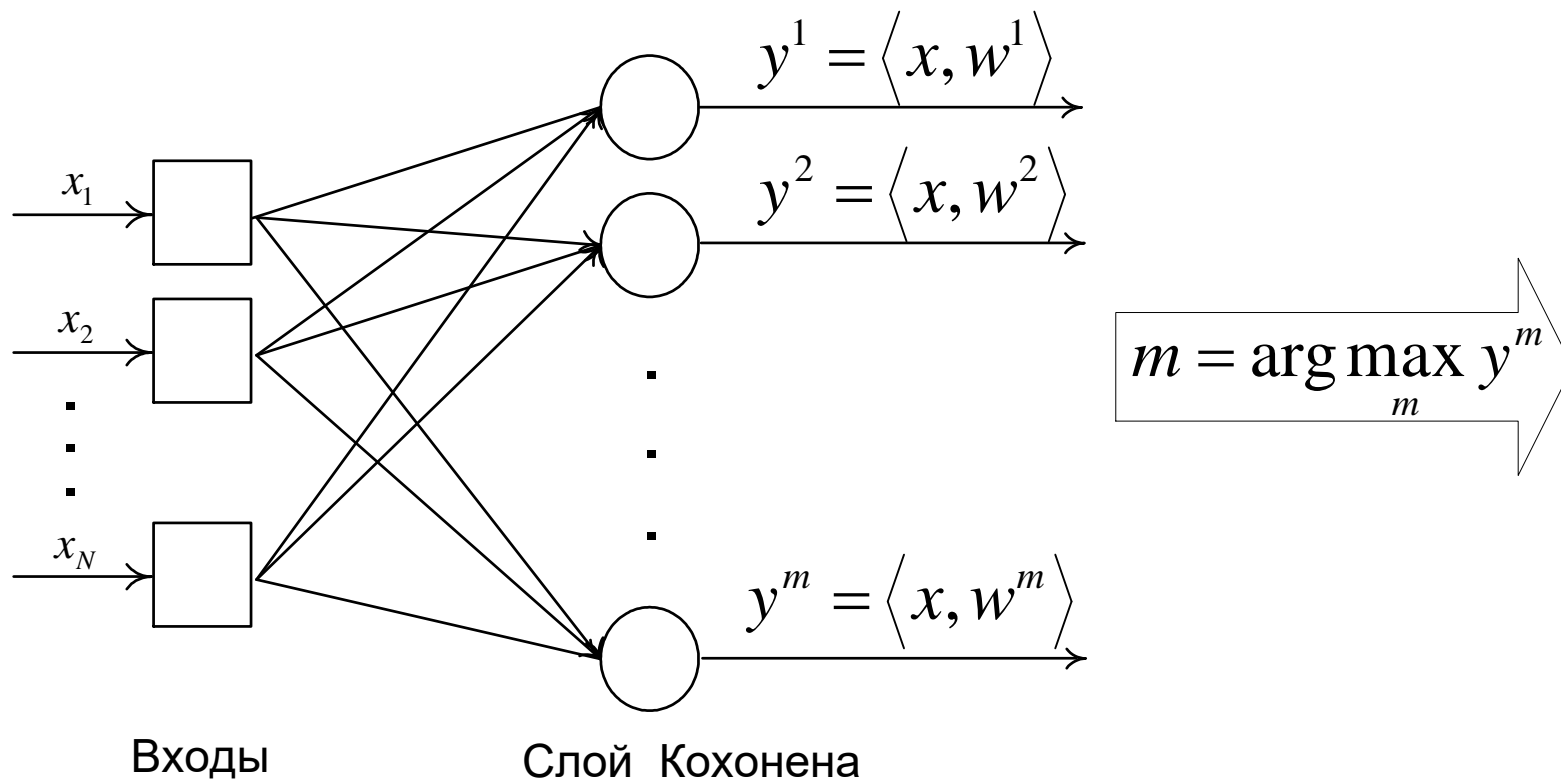
Объект  $x$  наиболее близок ядру  $m$ , для которого

$$d(x, w) = \min_{m=1}^M d(x, w^m)$$

или

$$d(x, w) = \max_{m=1}^M \langle x, w^m \rangle$$

НС ПР из входного слоя и выходного слоя, у которого функция активации - тождественная единица. Выход – номер нейрона с максимальным выходом.



- Весовые векторы нейронов слоя Кохонена - ядра.
- Количество кластеров совпадает с числом нейронов Кохонена.
- Весовые вектора нормированы.

### 3.2. Обучение без учителя

Задача обучения СОК - обучение без учителя (реакция сети на входной вектор неизвестна).

*Алгоритм («Победитель забирает всё», Winner-Takes-All).*

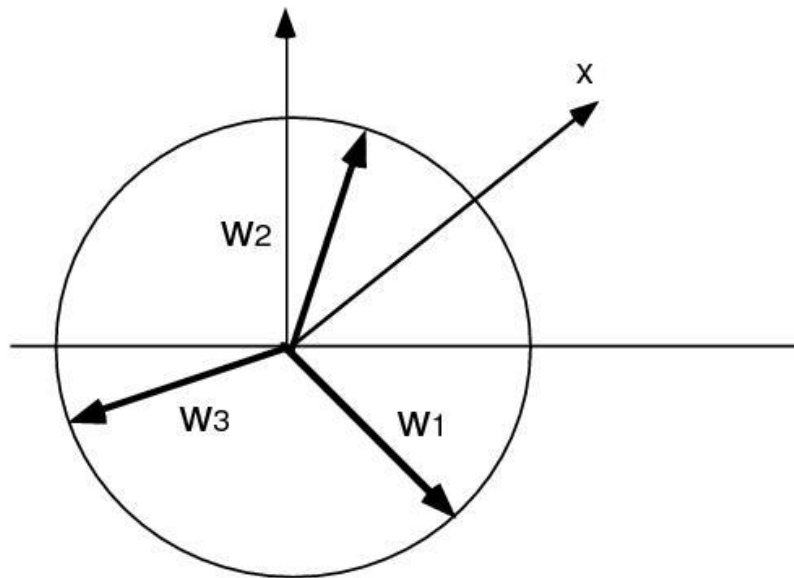
1. Выбор шага обучения  $\eta > 0$  ( $0,1 - 0,7$ ).  
Иногда  $\eta = \eta(t)$  – убывающая до 0 функция.
2. Инициализация весов сети небольшими случайными числами.
3. Настройка весов. Для каждого входного вектора:
  - 2.1. Расчет выходов нейронов слоя Кохонена. Выбирается нейрон-победитель  $r$  с наибольшим значением на выходе.
  - 2.2. Корректировка весов победившего нейрона:

$$w_r := w_r + \eta(x - w_r)$$

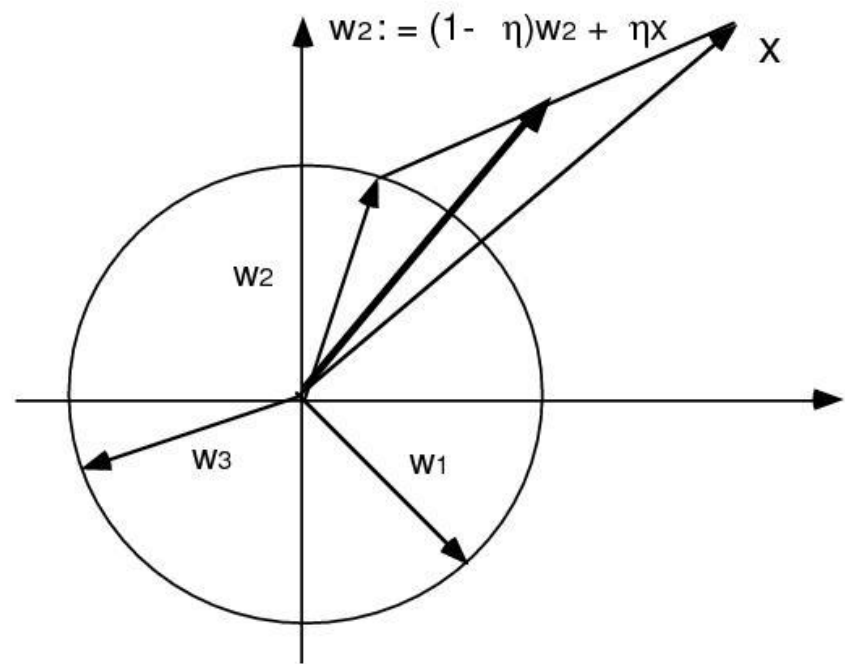
- 2.3. Нормировка вектора весов нейрона-победителя:

$$w_r := \frac{w_r}{\|w_r\|}$$

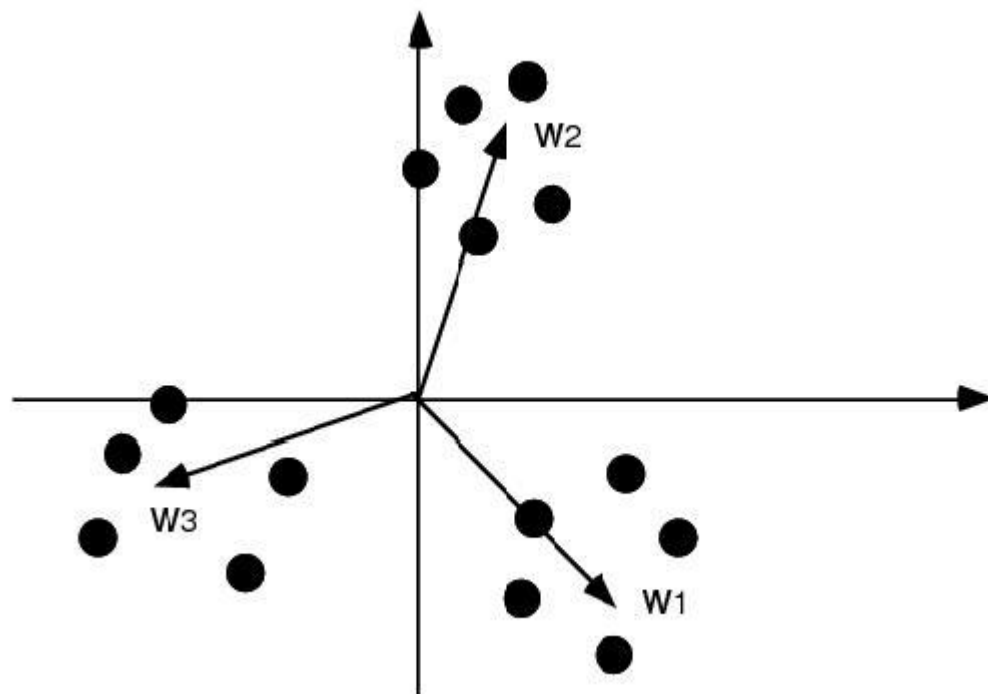
Нейрон-победитель:  
с вектором весов  $w^2$



Корректировка весов  
нейрона-победителя:



Результат обучения:



Модификация алгоритма «Победитель забирает всё»:

- обучение нейронов в некоторой окрестности;
- обучение пропорционально выходу;
- чувство справедливости (у часто выигрывающего нейрона уменьшают значение функции). «Механизма утомления» - потенциал нейрона  $p_i$  ( $k$  – номер шага обучения) :

$$p_i(k+1) = \begin{cases} p_i(k) + \frac{1}{n}, i \neq w \\ p_i(k) - p_{\min}, i = w \end{cases}$$

$p_{\min} = 0 \Rightarrow$  стандартный алгоритм

$p_{\min} = 1 \Rightarrow$  нейроны побеждают по очереди

$p_{\min} = 0,75 \Rightarrow$  хорошие результаты.

В определении победителя используются нейроны, у которых  $p_i \geq p_{\min}$ .

# ВЫВОДЫ (ОСНОВНЫЕ МОМЕНТЫ ЛЕКЦИИ)

1. Искусственный нейрон – **аналог** биологического нейрона.
2. Преобразование нейрона осуществляется в два этапа: вычисляется **уровень активности**, затем применяется **функция активации**.
3. Нейронные сети прямого распространения (НС ПР) обладают характерной **суперпозиционной структурой**.
4. Основа успешного применения НС ПР – их **универсальная аппроксимационная способность**.
5. Центральный этап применения НС ПР – **обучение** (настройка весов сети фиксированной структуры).
6. Для обучения применяются методы оптимизации, в основе которых – вычисление **градиента функционала качества** НС ПР по вектору весов. Эффективный метод вычисления градиента – процедура **обратного распространения ошибки**.

## СПИСОК ЛИТЕРАТУРЫ

1. Сараев П.В. Нейросетевые методы искусственного интеллекта: Учебное пособие. - Липецк: ЛГТУ, 2007.- 64 с.
2. Блюмин С.Л., Шуйкова И.А., Сараев П.В., Черпаков И.В. Нечеткая логика: алгебраические основы и приложения. Липецк: ЛЭГИ, 2002 – 107с.
3. Fuller R. Introduction to Neuro-Fuzzy Systems.– Berlin/Heidelberg, Springer-Verlag, 2000.– 289 p.
4. Осовский С. Нейронные сети для обработки информации.– М.: Финансы и статистика, 2004. – 344 с.
5. Круглов В.В., Борисов В., Искусственный нейронные сети: теория и практика. М.: Горячая линия – Телеком, 2002.– 382 с.

**Спасибо за внимание!**