

# ЛЕКЦИЯ #9

## Корлякова М.О.

# МОДЕЛЬ НЕЙРОНА МАККАЛОКА-ПИТТСА

```
#selection at the end -add back the deselected mirror modifier ob
mirror_ob.select=1
#deselect other modifiers
bpy.context.scene.objects.active = modifier_ob
print("Selected" + str(modifier_ob)) # modifier ob is the active ob
# Mirror ob.select = 0
#ob = bpy.context.scene.objects.active
#ob.modifiers.remove(modifier)
```



# ИСТОРИЯ И ПРЕДПОСЫЛКИ

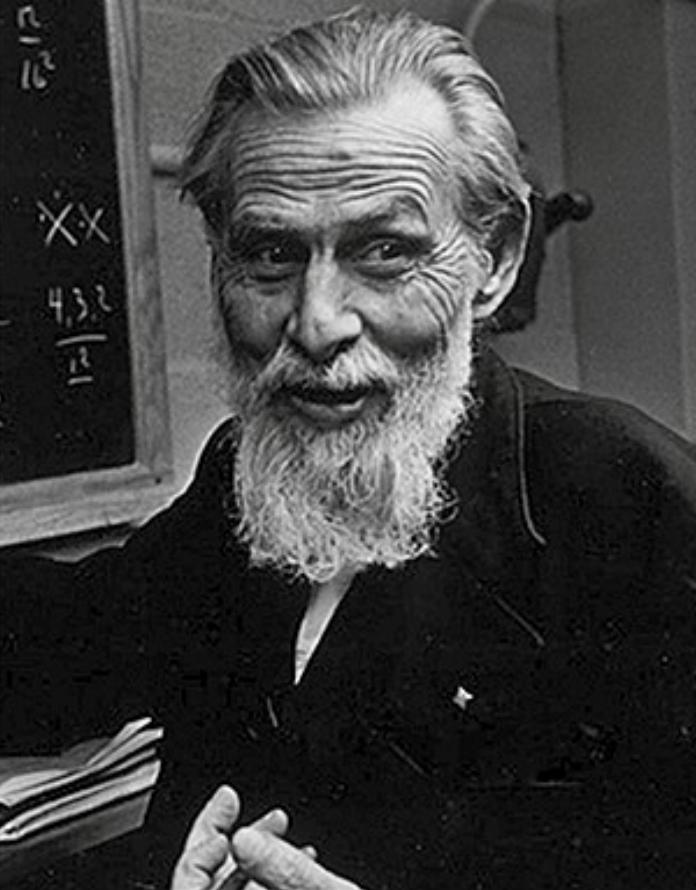
---

# История и предпосылки

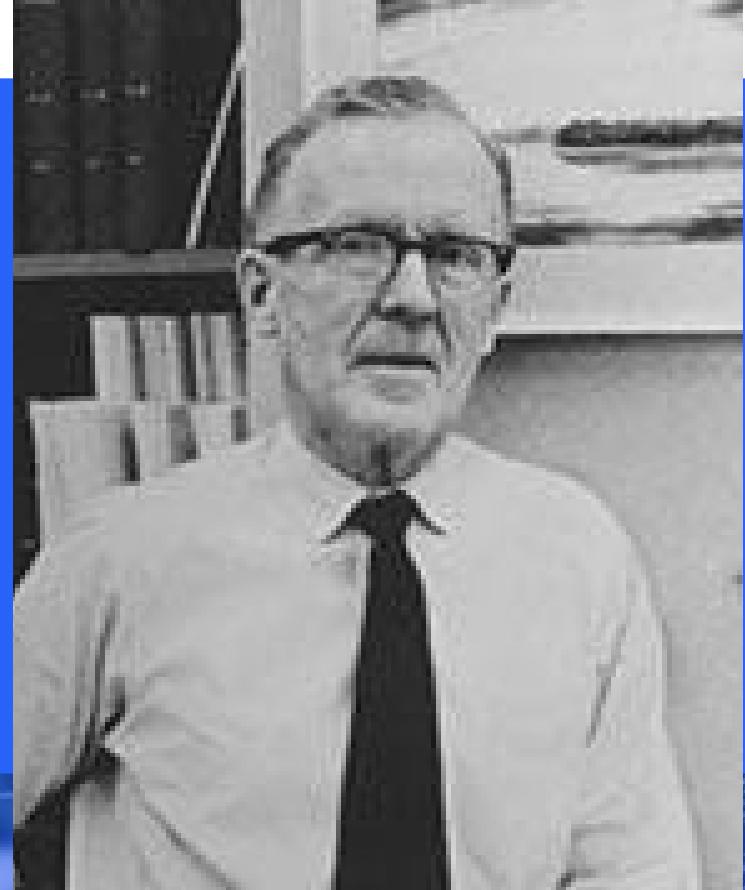
---



**Уолтер  
Питтс,**  
нейролингвист



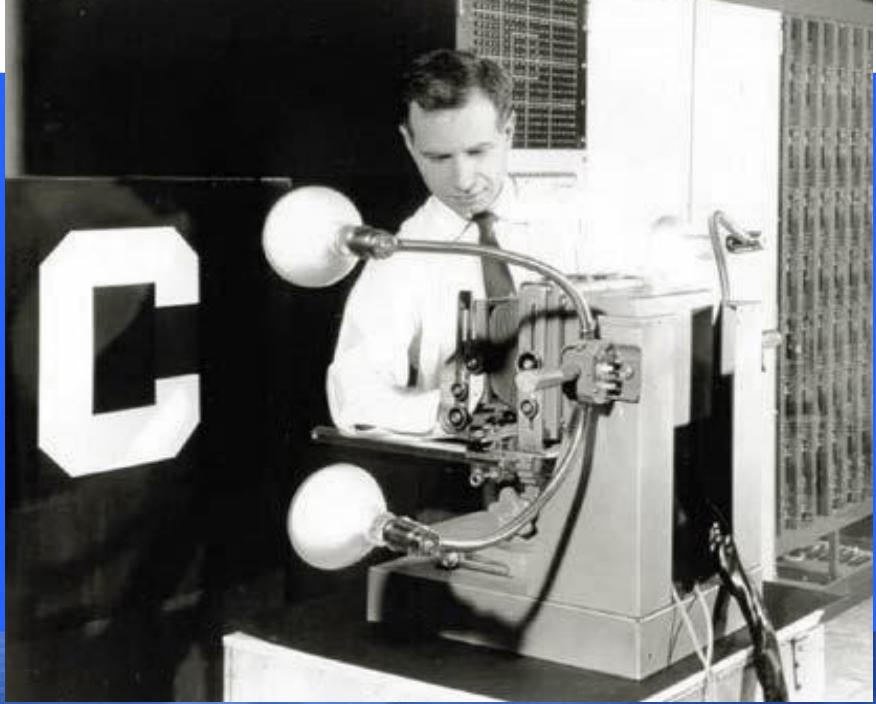
**Уоррен Мак-  
Каллок,**  
нейролингвист



**Дональд  
Хебб,**  
нейрофизиоло

# История и предпосылки

---



Ф.  
Розентлат



М. Минский, С.  
Пайперт

# ИСТОРИЯ И ПРЕДПОСЫЛК И



- Сеть Хопфилда
- Сеть Кохонена
- Обратное распространение ошибки
- Когнитрон

# ИСТОРИЯ И ПРЕДПОСЫЛКИ

- Глубокое обучение
- Свёрточные сети



Фукушима



Р.Дехтер

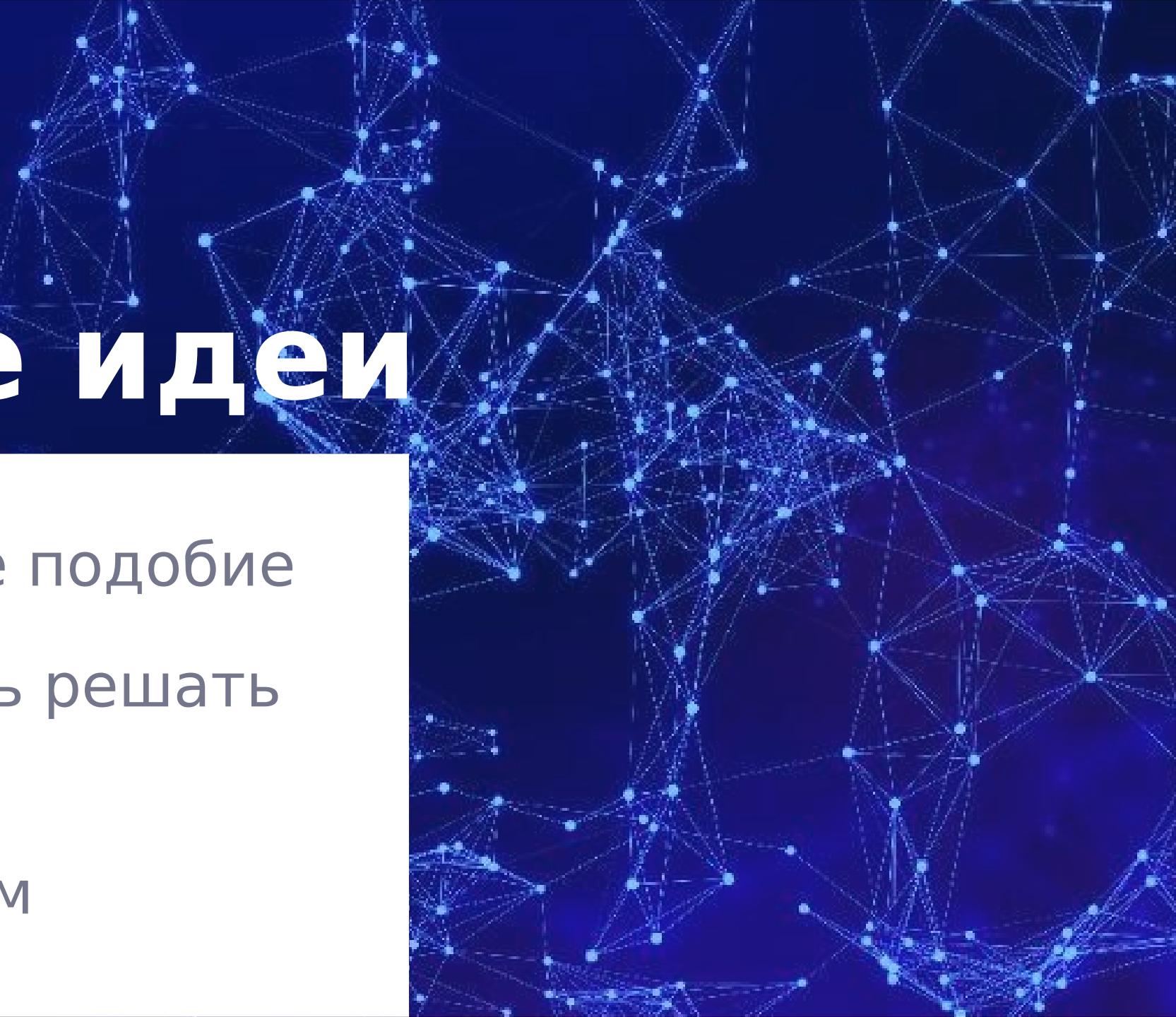


Ян ЛеКун



# Базовые идеи

- Биологическое подобие
- Необходимость решать задачи
- Коннекционизм



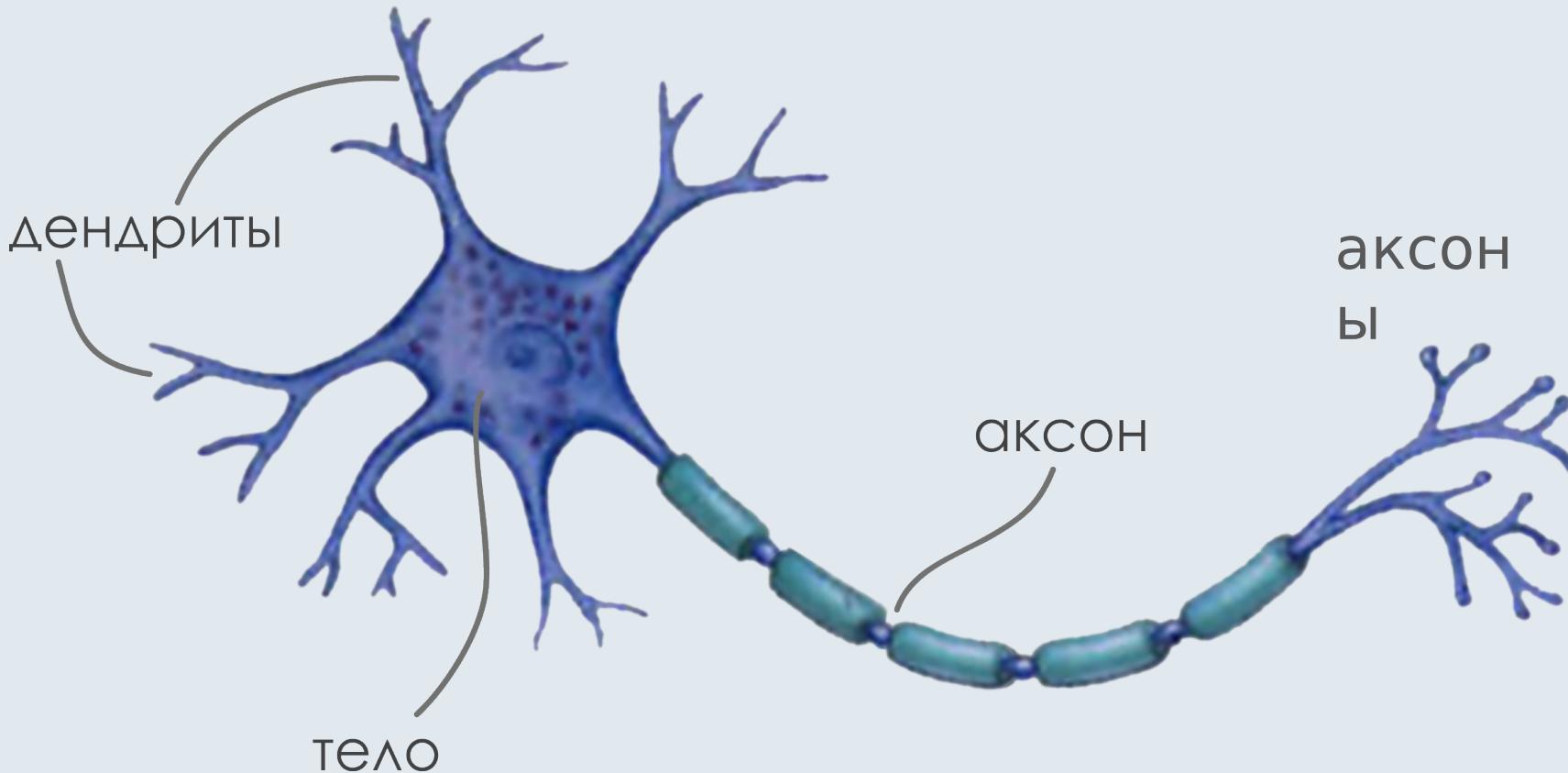
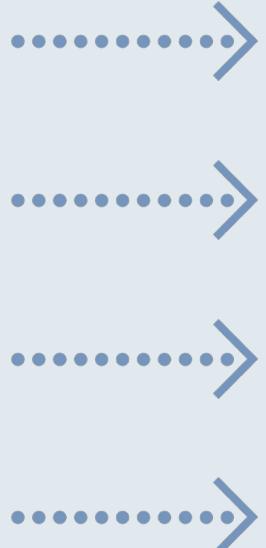
---

# Реализация нейрона МакКалока-Питца

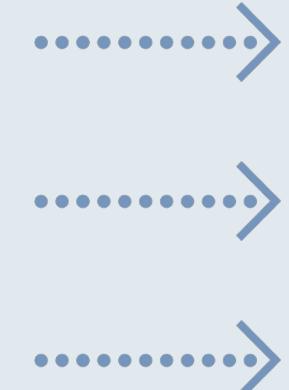


# Физиологический нейрон

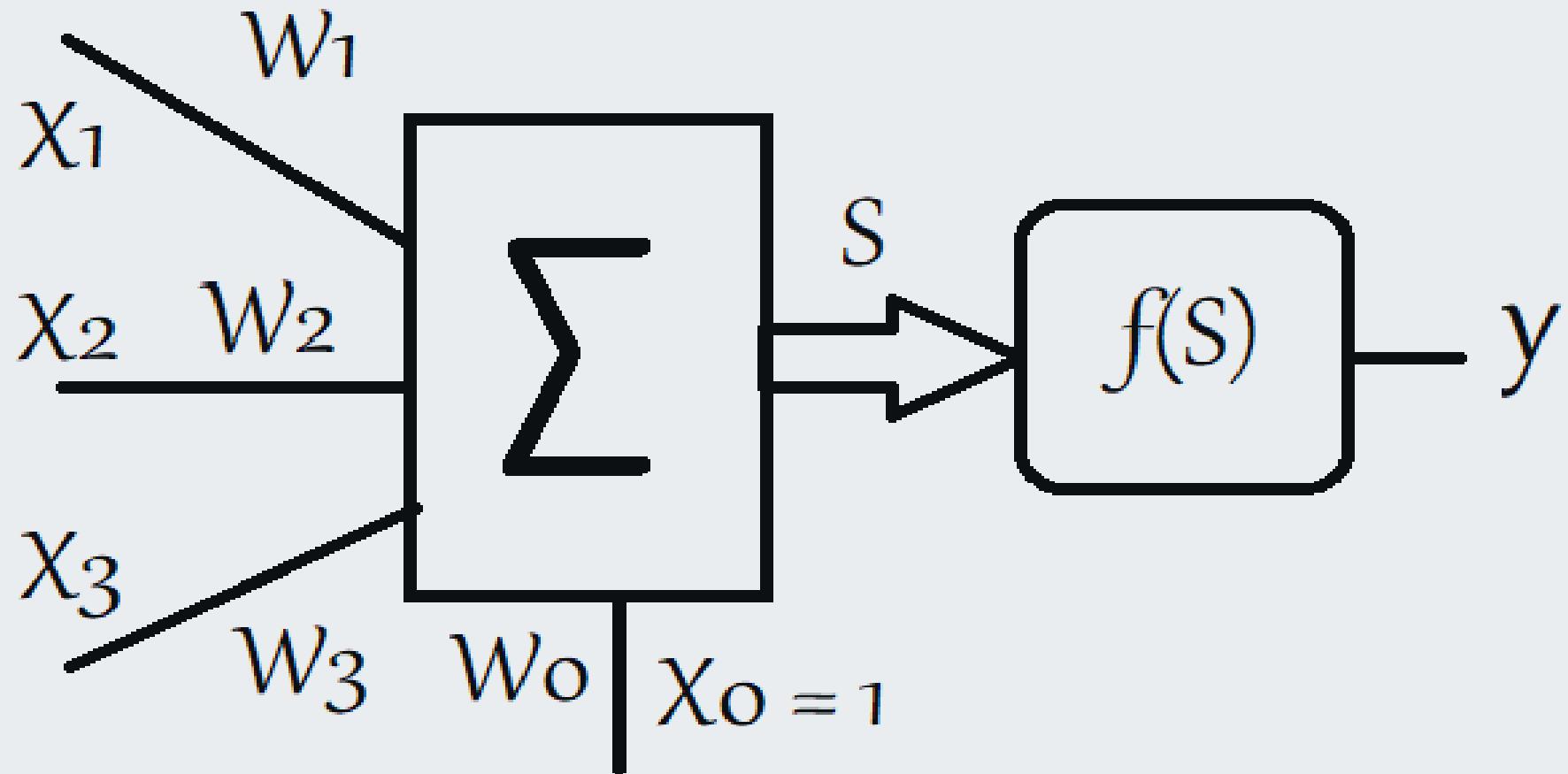
Входно  
й  
сигнал



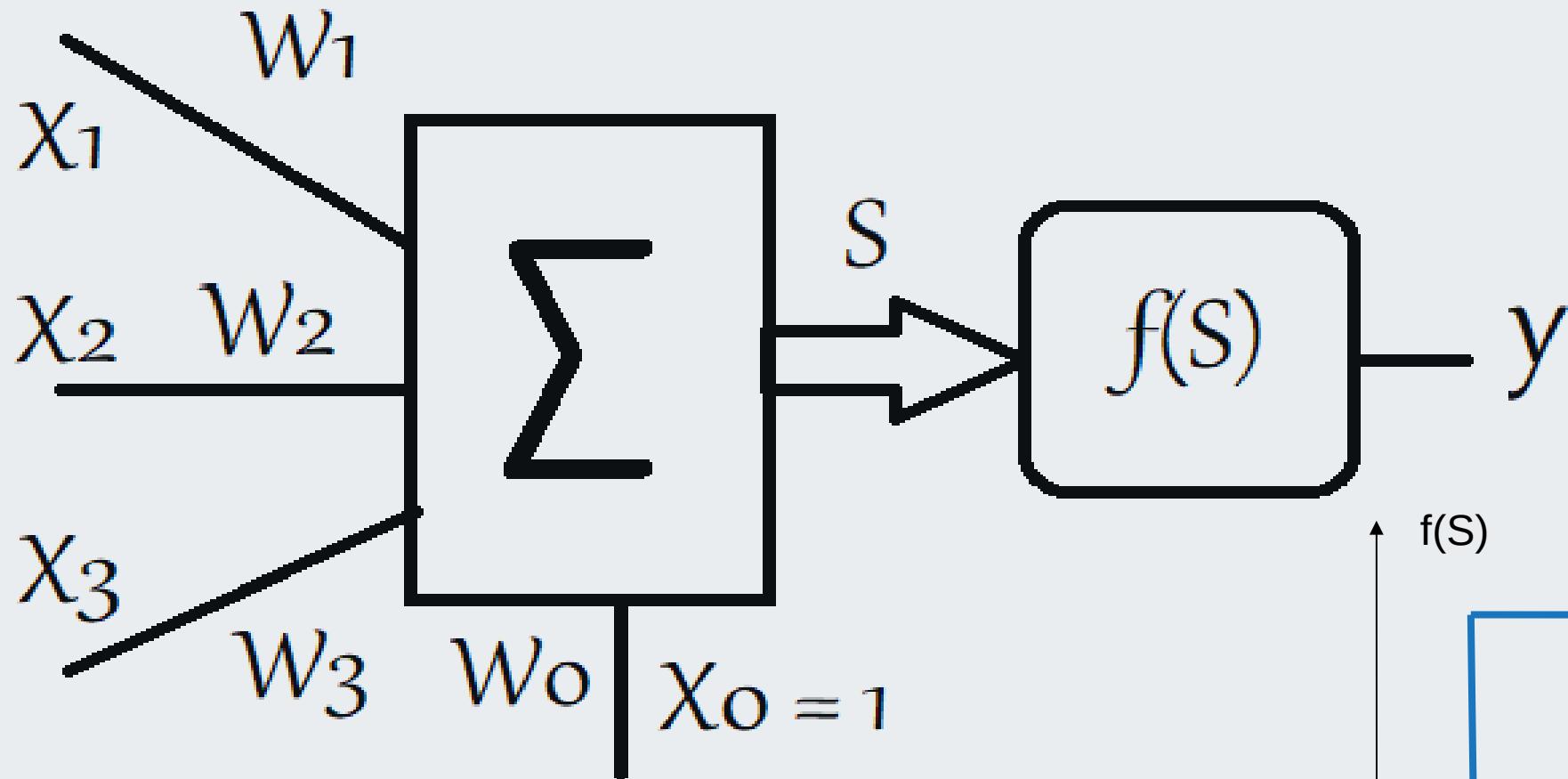
Выходно  
й  
сигнал



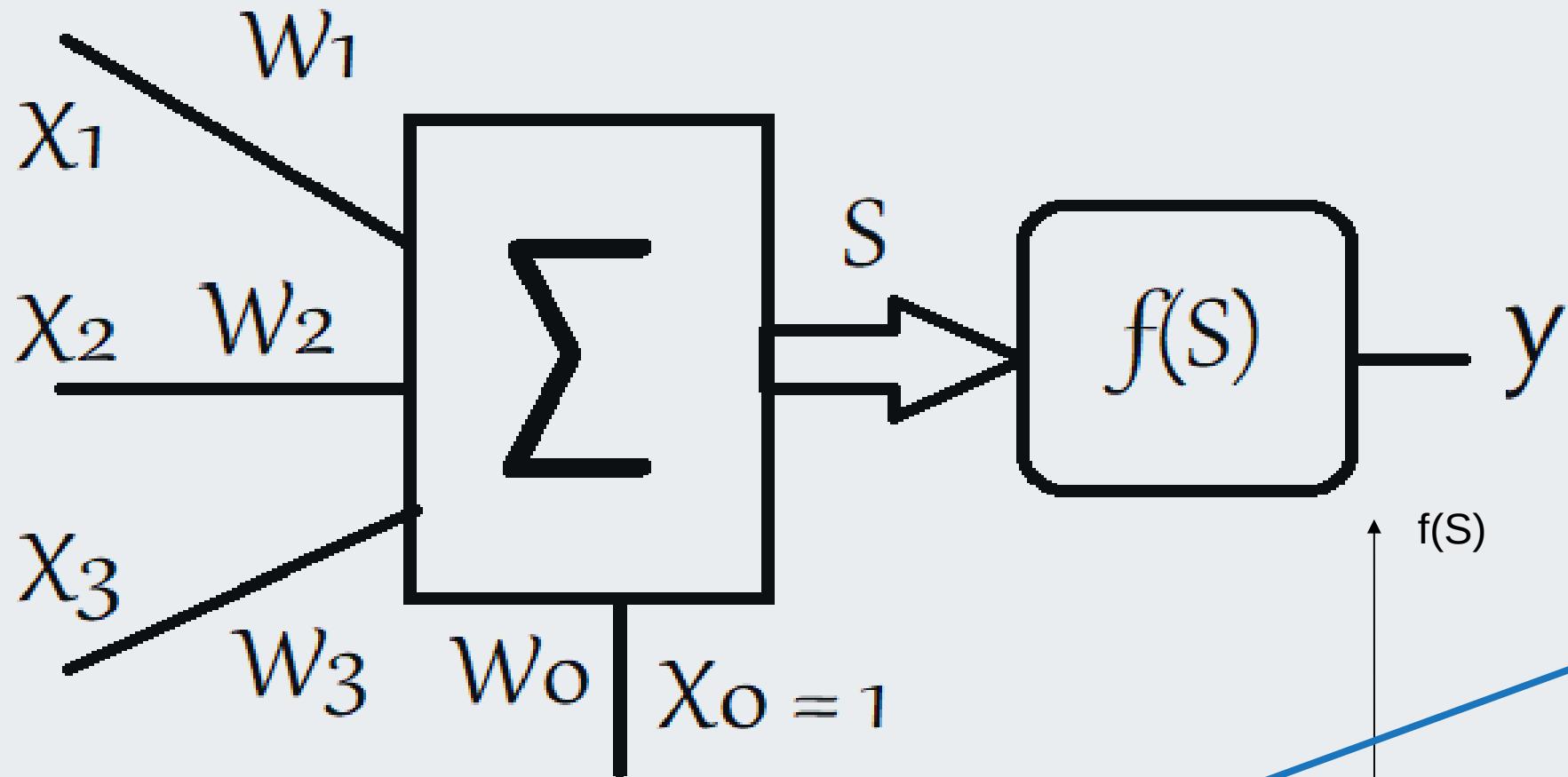
# Формальный нейрон



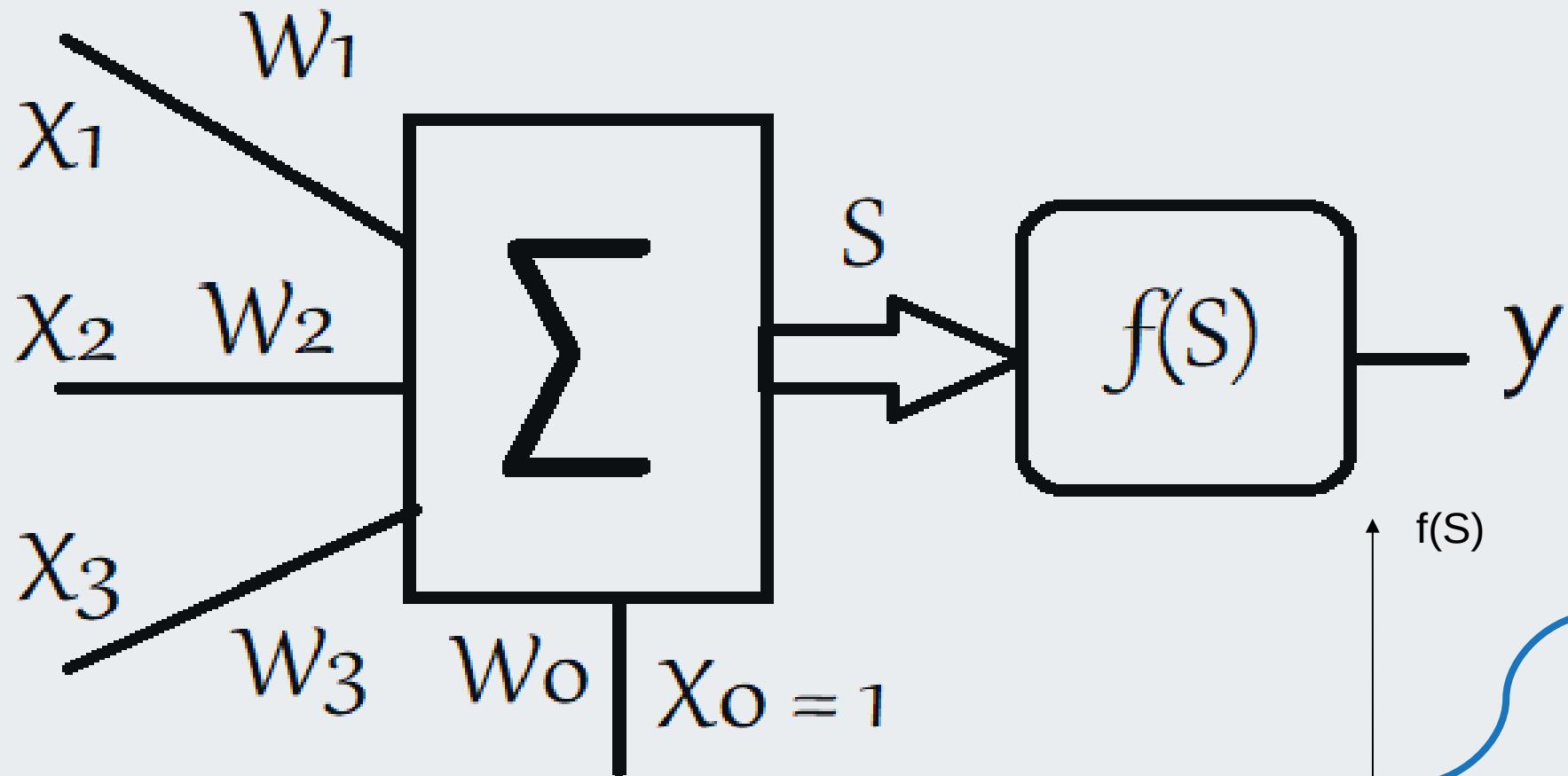
# Формальный нейрон



# Формальный нейрон



# Формальный нейрон

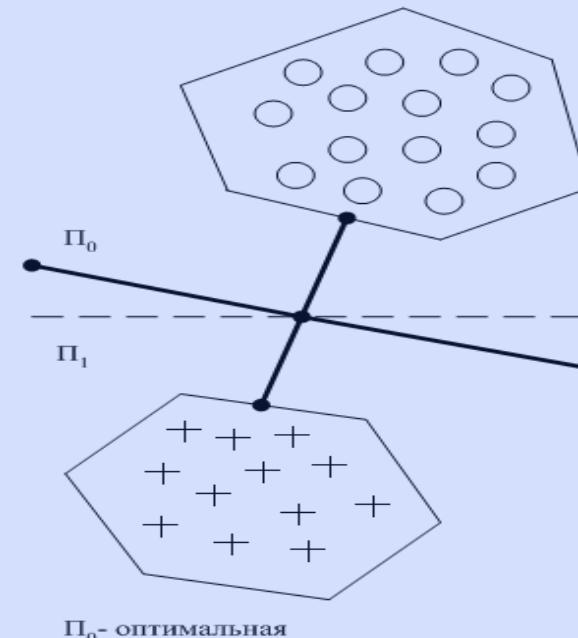
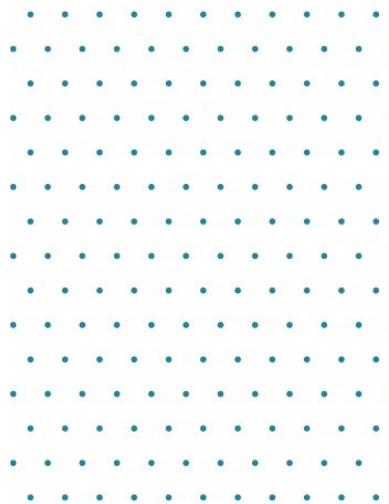


---

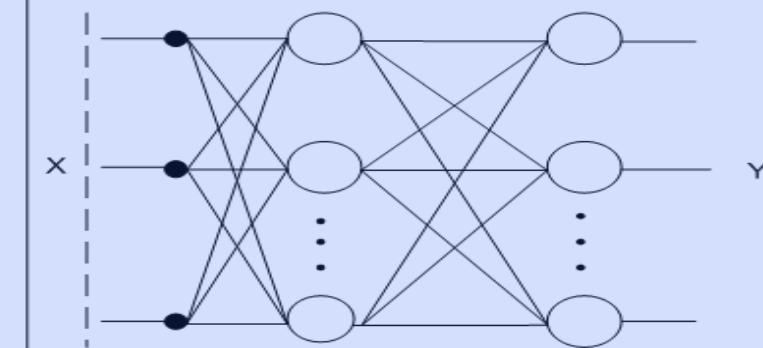
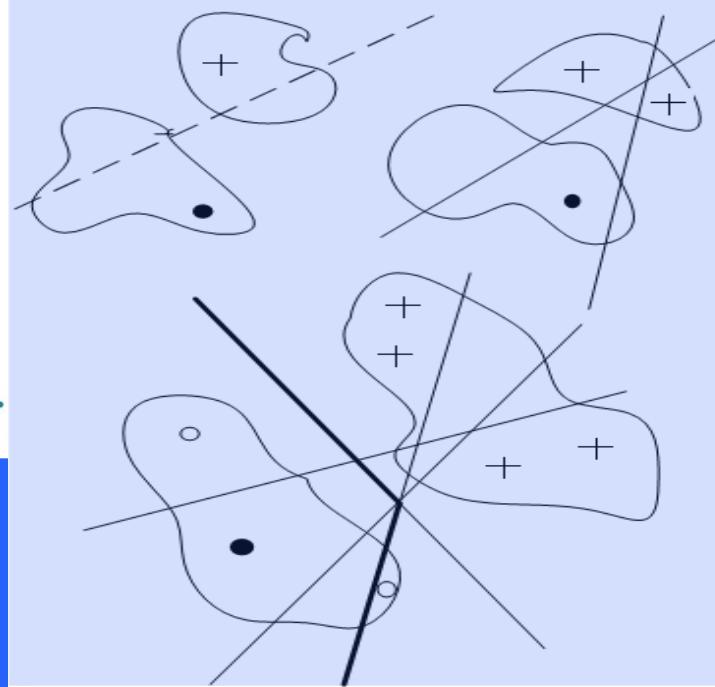
# Обучение нейрона по Дельта-правилу



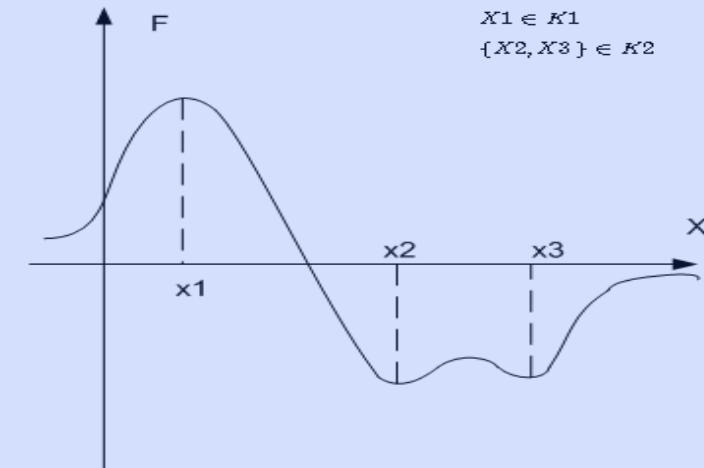
# ОБУЧЕНИЕ СЕТЕЙ



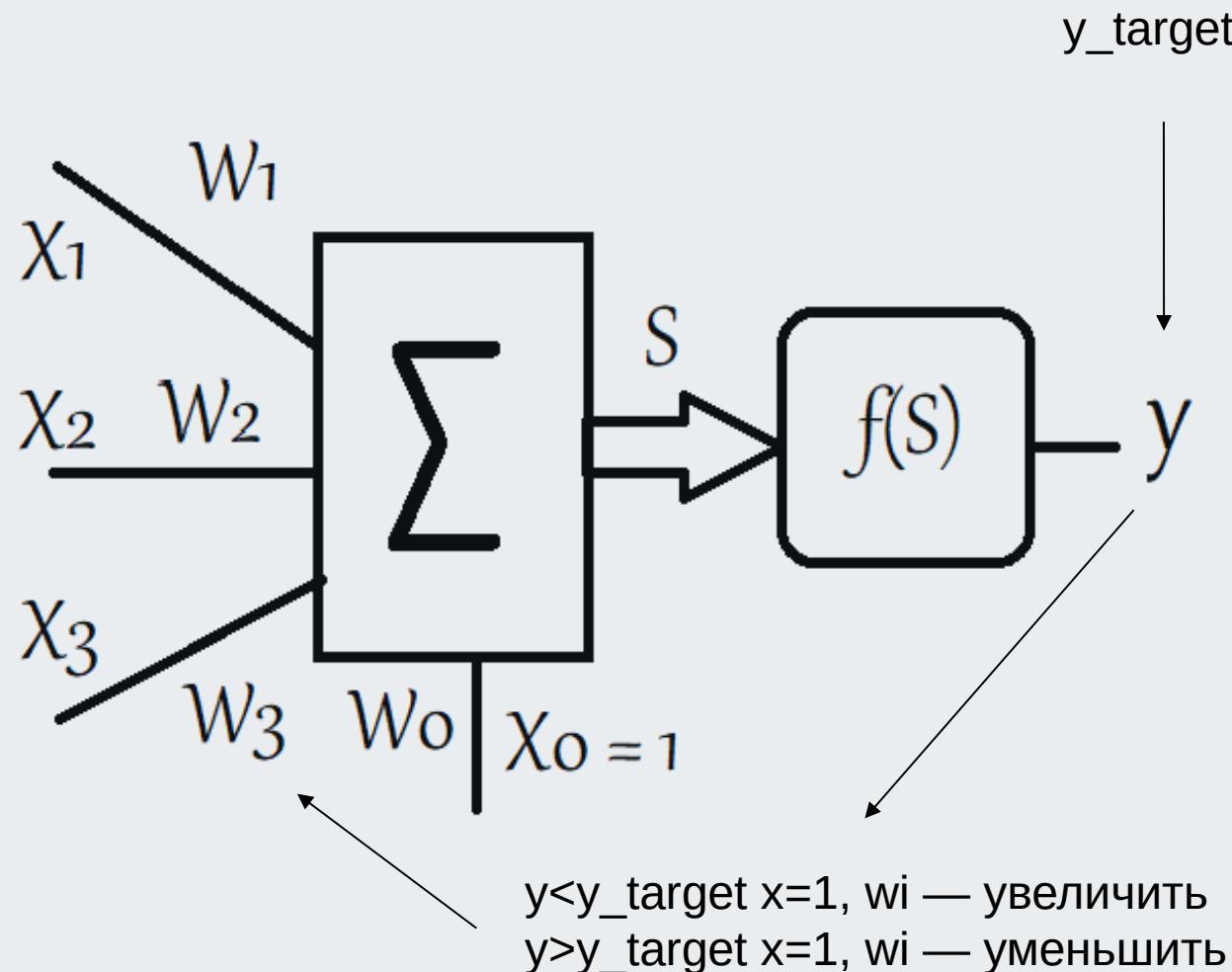
## Формирование гиперплоскости



## Потенциальная функция



# Обучение по правилу Хебба



$$x_i \in \{0,1\}$$

$$y = \begin{cases} 1, s \geq 0 \\ -1, s < 0 \end{cases}$$

$$y \in \{-1,1\}$$

$$w_i = w_i \pm \Delta$$

# Обучение по Дельта- правилу

$$x_i \in [0,1]$$

$$y = S, \quad S = \sum_{i=0,n} w_i x_i, x_0 = 1$$

$$y \in [0,1]$$

$$E = \frac{1}{2} (y(S) - \hat{y})^2 \quad , \quad \frac{\partial E}{\partial w_i} = (y(S) - \hat{y}) \frac{\partial f(S)}{\partial S} \frac{\partial S}{\partial w_i}$$

$$\frac{\partial E}{\partial w_i} = (y(S) - \hat{y}) \cdot 1 \cdot x_i$$

$$\Delta w_i = -\eta x_i (y(S) - \hat{y})$$

$$w_i = w_i - \eta (y(S) - \hat{y}) x_i$$

# Обобщенное Дельта-правило

$$x_i \in [0,1]$$

$$y = f(S), S = \sum_{i=0,n} w_i x_i, x_0 = 1$$

$$y \in [0,1]$$

$$E = \frac{1}{2} (y(S) - \hat{y})^2$$

$$\frac{\partial E}{\partial w_i} = (y(S) - \hat{y}) \frac{\partial f(S)}{\partial w_i} \frac{\partial S(X,W)}{\partial w_i} = (y(S) - \hat{y}) x_i \frac{\partial f(S)}{\partial w_i}$$

$$w_i = w_i - \eta (y(S) - \hat{y}) x_i f'(S)$$

# Сигмоид Дельта-правило

$$x_i \in [0,1]$$

$$y = \frac{1}{1+e^{-S/\lambda}}, S = \sum_{i=0,n} w_i x_i, x_0 = 1$$

$$y \in [0,1]$$

$$E = \frac{1}{2} (y(S) - \hat{y})^2$$

$$\frac{\partial E}{\partial w_i} = (y(S) - \hat{y}) x_i \frac{\partial f(S)}{\partial w_i} = (y(S) - \hat{y}) x_i f(S)(1 - f(S))$$

$$w_i = w_i - \eta (y(S) - \hat{y}) x_i f(S)(1 - f(S))$$

$$w_i = w_i - \eta (y(S(X,W)) - \hat{y}) x_i y(S(X,W))(1 - y(S(X,W)))$$



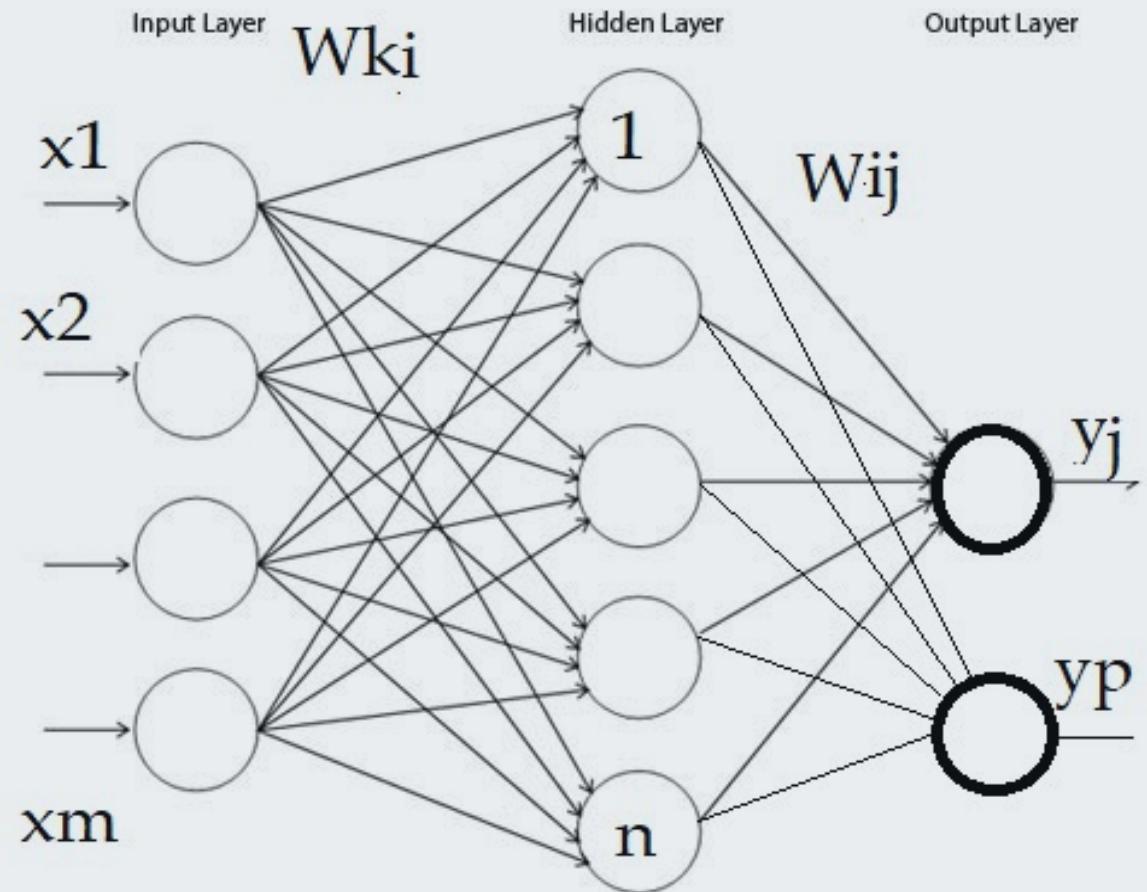
# Реализация слоя нейронной сети

# Нейронная сеть

Слой

Множество слоев

$$y_j = f_j \left( \sum_{i=0,n} w_{ij} f_i \left( \sum_{k=0,m} w_{ki} x_k \right) \right)$$



# Теорема Минского-Пайпerta

Многослойный линейный перцептрон – Эквивалентен  
однослойному линейному перцептруну

$$y_j = f_j \left( \sum_{i=0,n} w_{ij} f_i \left( \sum_{k=0,m} w_{ki} x_k \right) \right)$$

$$y_j = W_j^p \left( W^{p-1} \left( W^{p-2} \dots \left( W^1 X \right) \right) \right) = WX$$

# Посчитаем настраиваемые параметры

2 входа

2 нейрона в слое 1

1 нейрон в слое 2

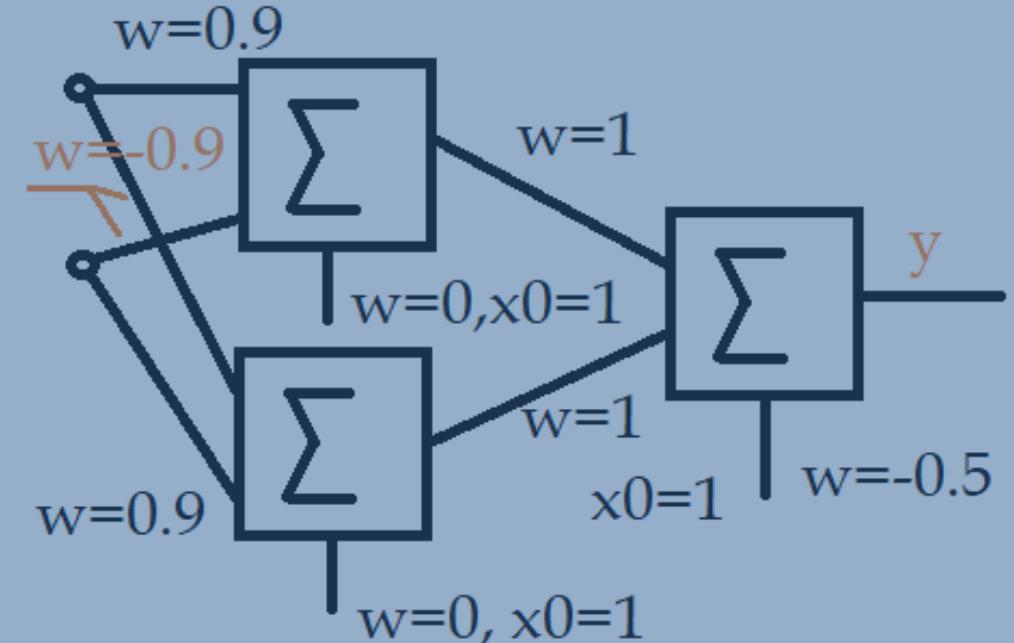
Число параметров слоя:

Число входов x число  
нейронов + число нейронов

Число параметров:

Слой 1:  $2*2+2$

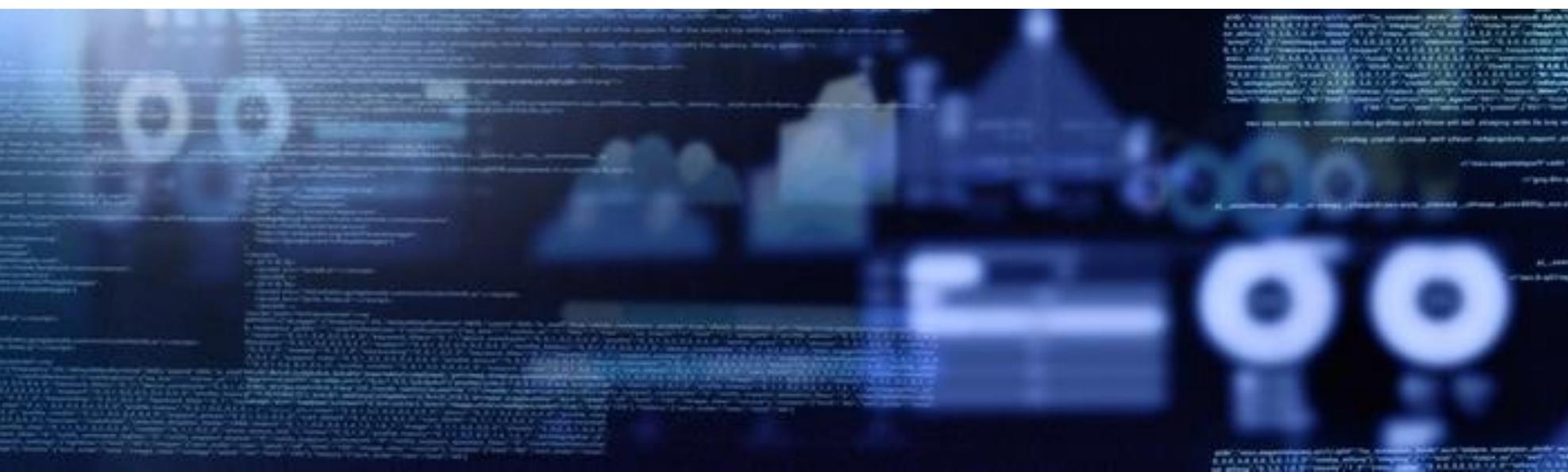
Слой 2:  $2*1+1$



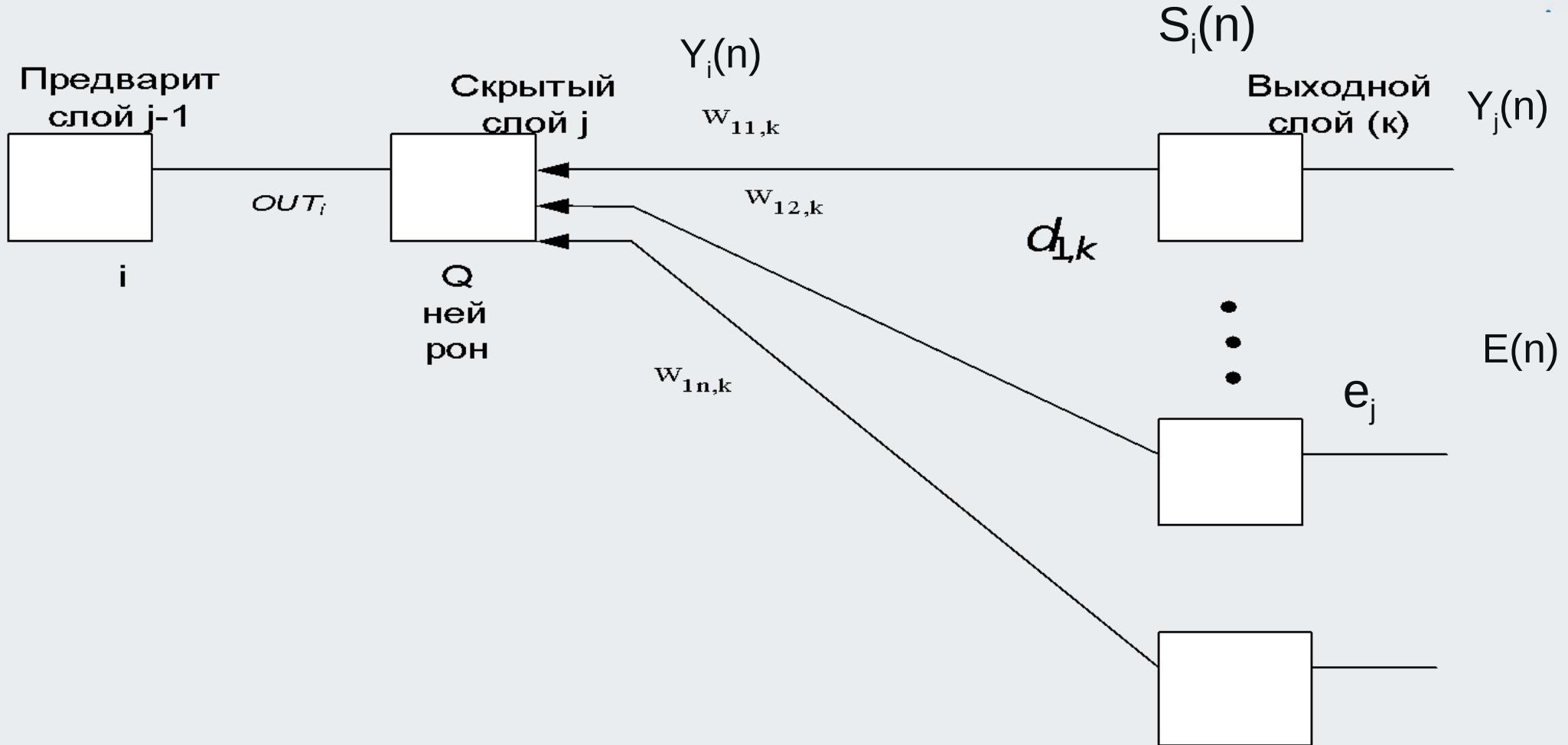
# ЭПИЗОД 2

---

Метод Обратного  
Распространения Ошибки



# Фрагмент сети



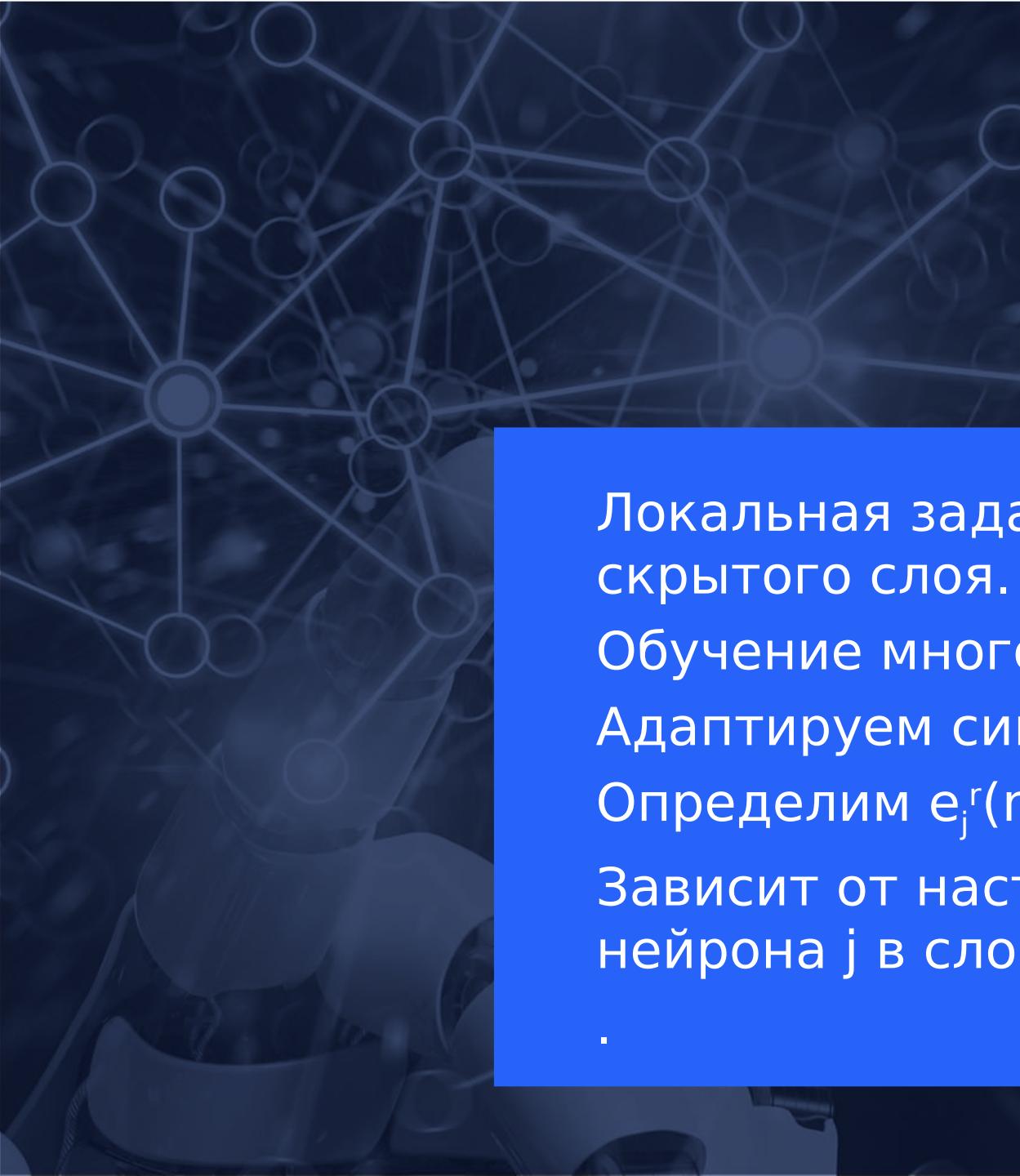
# Обучение по ошибке

Глобальная задача – сложная.

$$E(n) = \sum e_j(n)^2$$

$$e_j(n) = D_j - Y_j(n)$$

Зависит от всех настраиваемых параметров.



## Обучение по ошибке

Локальная задача - нет  $D_j$  для нейронов скрытого слоя.

Обучение многократное.

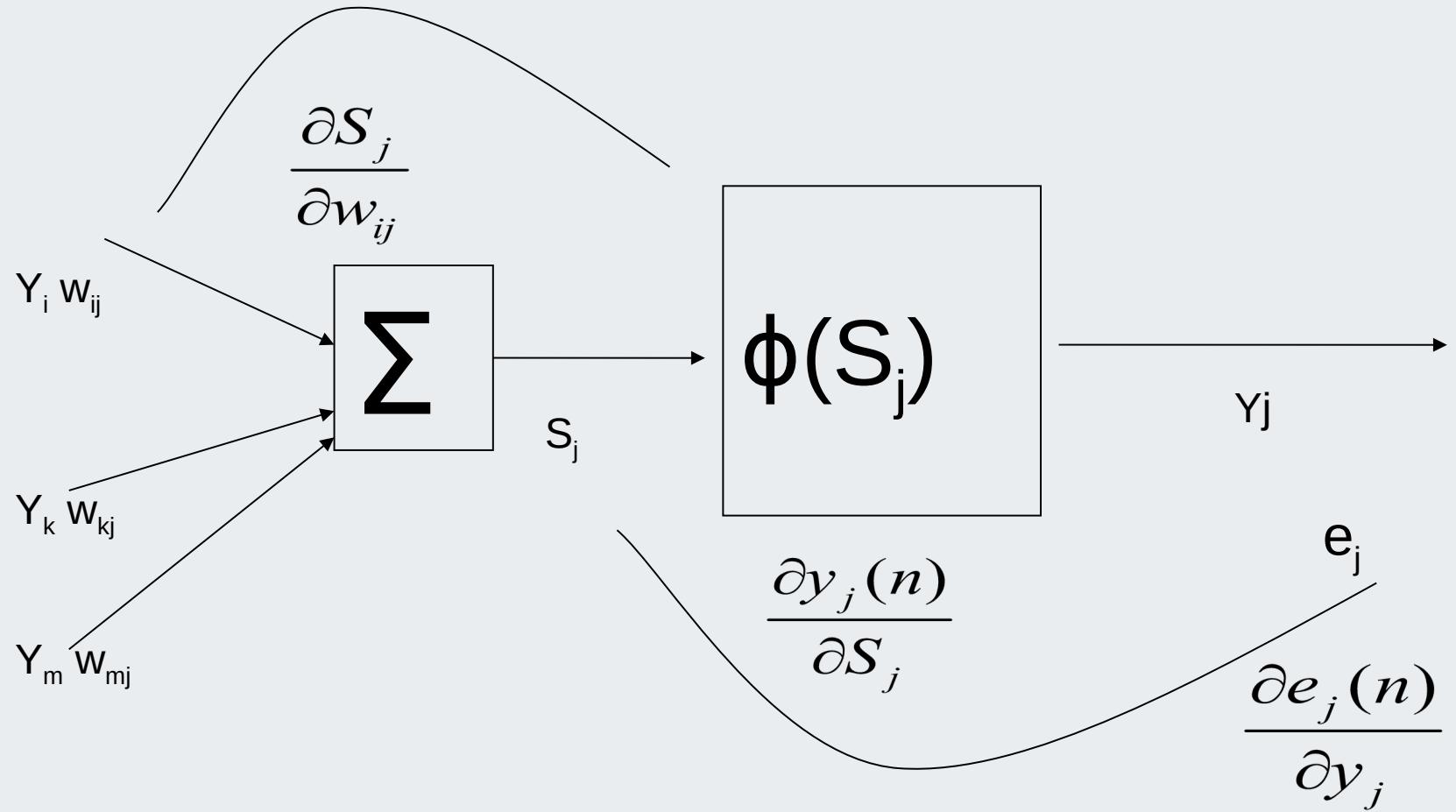
Адаптируем синаптические веса по ошибке.

Определим  $e_j^r(n)$  для нейронов скрытого слоя  $r$ .

Зависит от настраиваемых параметров только нейрона  $j$  в слое  $r$ .

.

# Нейрон j



$$\frac{\partial e_j(n)}{\partial w_{ij}}$$

$$\frac{\partial e_j(n)}{\partial S_j}$$

$$\frac{\partial e_j(n)}{\partial y_j}$$

# Обратное распространение ошибки выходной слой

$$\frac{\partial E(n)}{\partial w_{ij}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial S_j(n)} \frac{\partial S_j(n)}{\partial w_{ij}(n)}$$

$$E(n) = \frac{1}{2} \sum_{j=1}^k e_j(n)^2$$

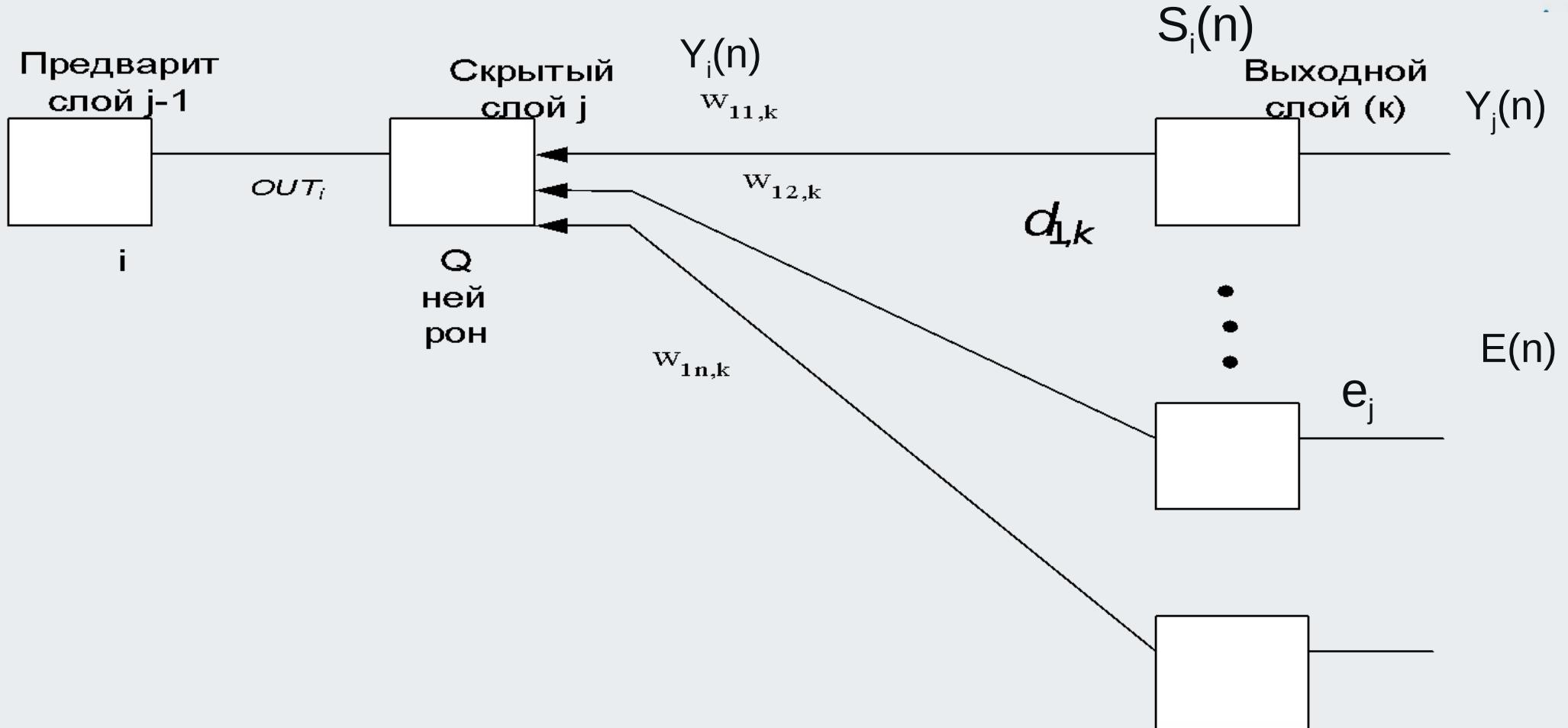
$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) = d_j - y_j(n), \quad \frac{\partial e_j(n)}{\partial y_j(n)} = -1$$

$$y_j(n) = \varphi(S_j) \Rightarrow \frac{\partial y_j(n)}{\partial S_j(n)} = \varphi'(S_j(n))$$

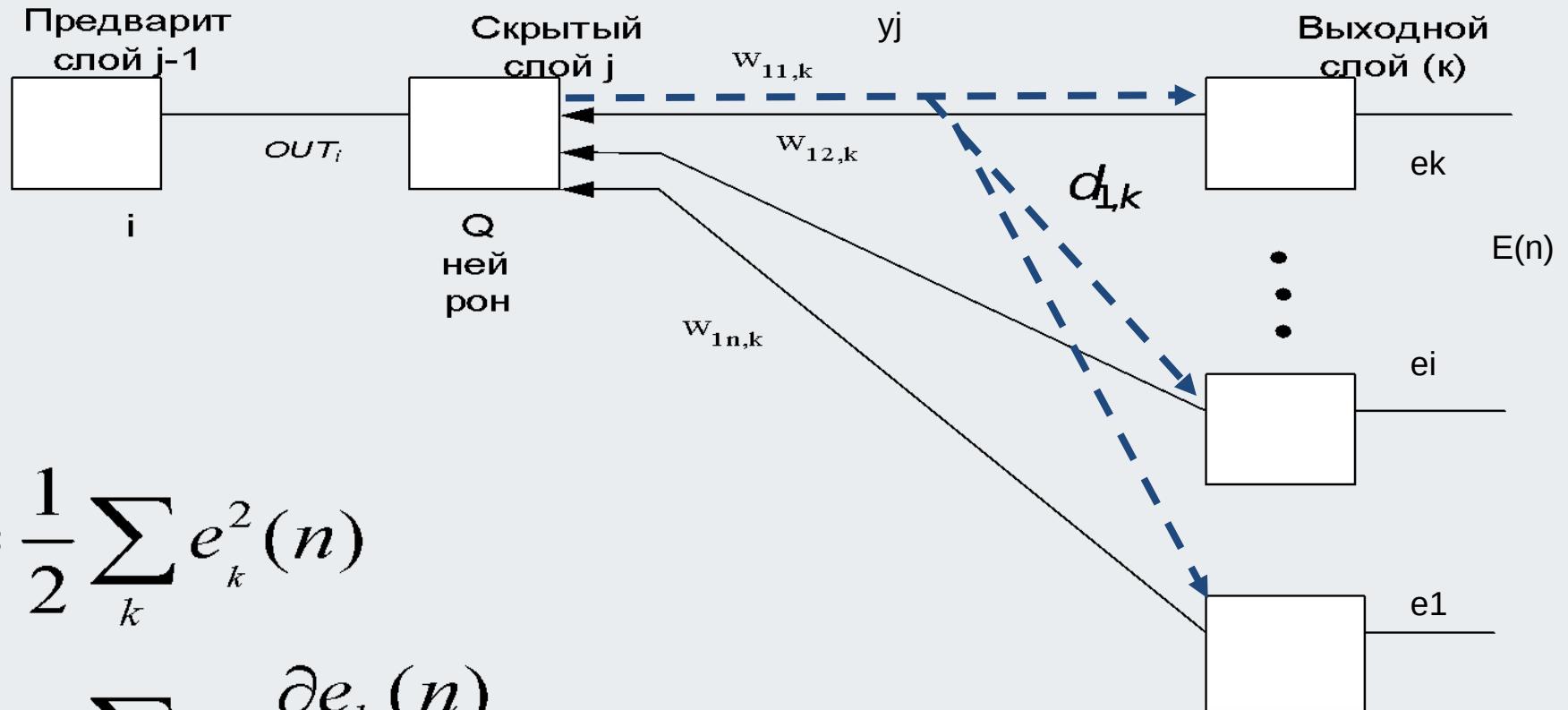
$$S_j(n) = \sum w_{ij} y_i(n) \Rightarrow \frac{\partial S_j(n)}{\partial w_{ij}(n)} = y_i(n)$$

$$\frac{\partial E(n)}{\partial w_{ij}(n)} = -e_j(n) \varphi'(S_j(n)) y_i(n) \Rightarrow \Delta w_{ij}(n) = -\eta \frac{\partial E(n)}{\partial w_{ij}(n)}$$

# Фрагмент сети



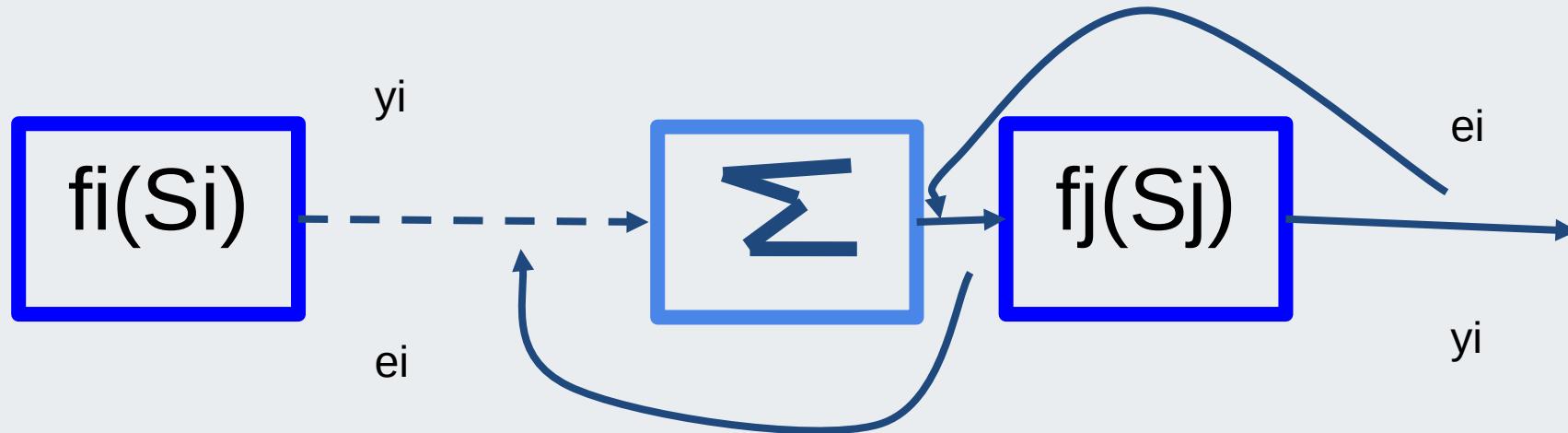
# Скрытый слой



$$E(n) = \frac{1}{2} \sum_k e_k^2(n)$$

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)}$$

# Скрытый слой



$$\delta_{ij}(n) = \frac{\partial e_j}{\partial y_i} = \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial S_j(n)} \frac{\partial S_j(n)}{\partial y_i} = -\varphi'_j(S_j(n)) w_{ij}$$

$$e_i = \sum_k e_k \delta_{ik}(n)$$

# Скрытый слой

$$S_k(n) = \sum_{j=0}^m w_{kj}(n) y_j(n)$$

$$\frac{\partial S_k(n)}{\partial y_j(n)} = w_{kj}(n)$$

$$\begin{aligned}\frac{\partial E(n)}{\partial y_j(n)} &= \sum_k e_k \frac{\partial e_k(n)}{\partial S_j(n)} \frac{\partial S_j(n)}{\partial y_j(n)} = \\ &= -\sum_k e_k(n) \varphi'_k(S_k(n)) w_{kj}(n) = -\sum_k \delta_k(n) w_{kj}(n)\end{aligned}$$

---

# Реализация обратного распространения ошибки



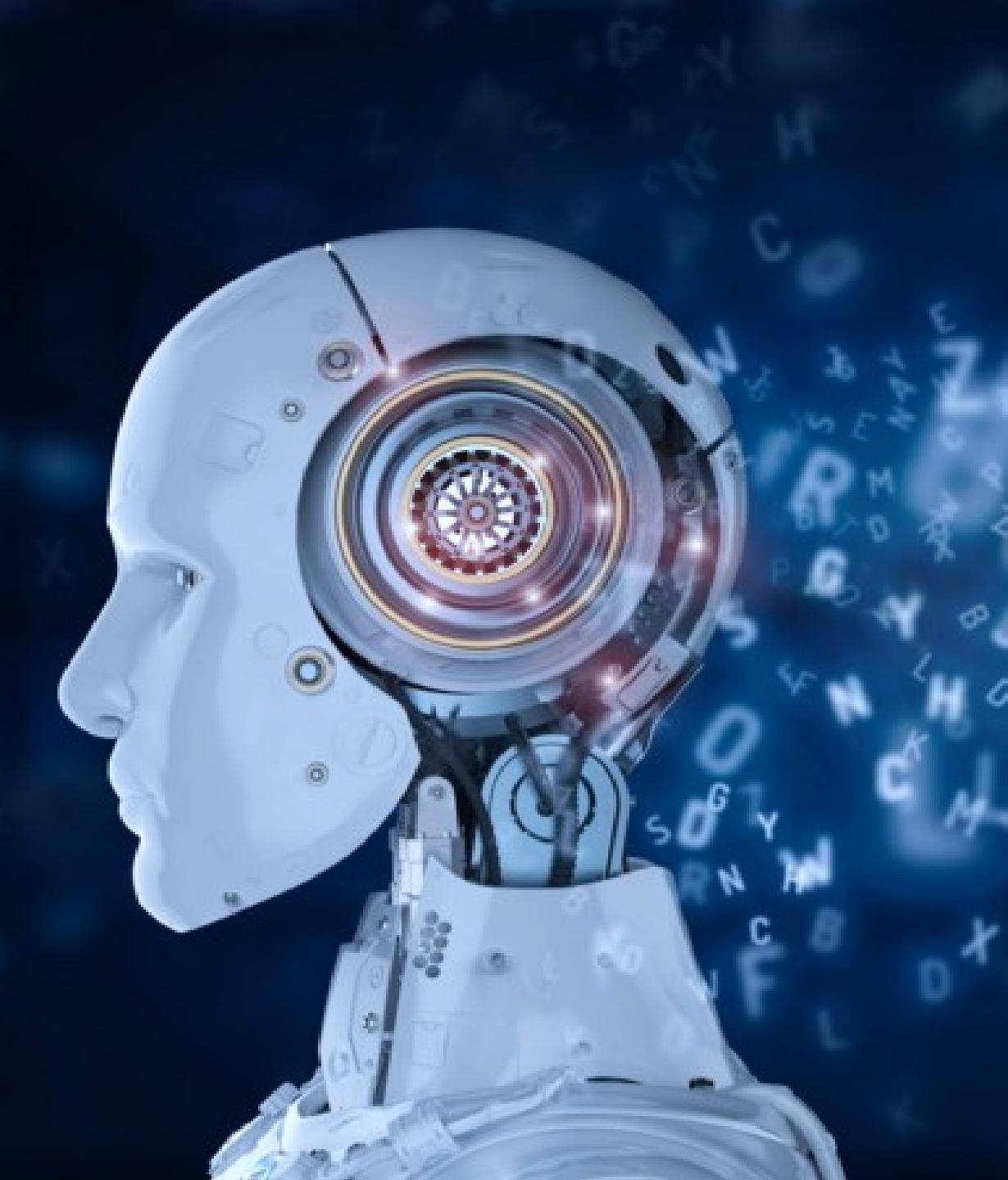
---

# Обучение нелинейного MLP



# Проблемы обучения нелинейного MLP

- Переобучение,
- Недообучение,
- Высокая  
Вычислительная  
Сложность,
- Затухание Градиента



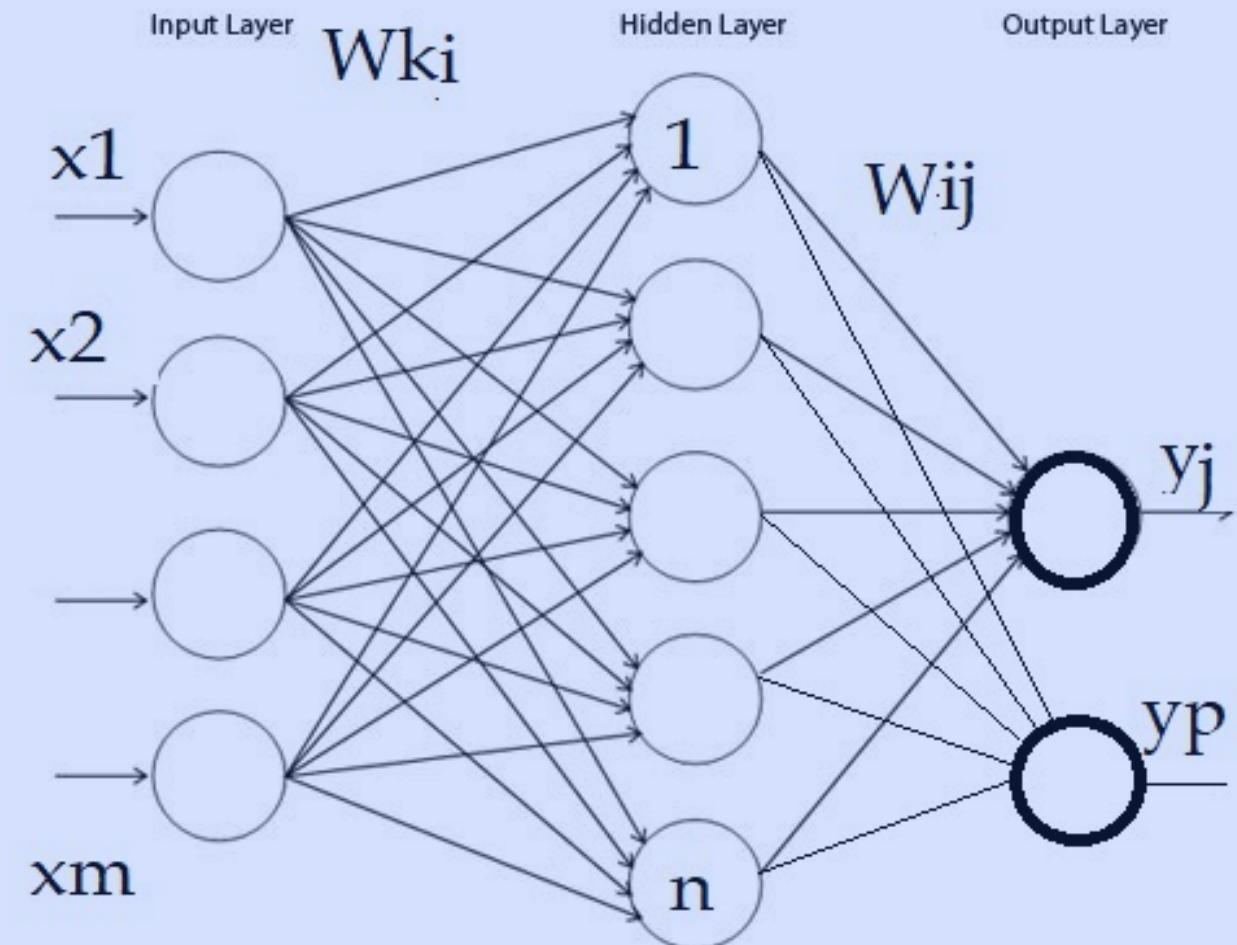


# Стохастический Градиент

---

# НЕЙРОННАЯ СЕТЬ

$$y_j = f_j \left( \sum_{i=0,n} w_{ij} f_i \left( \sum_{k=0,m} w_{ki} x_k \right) \right)$$





# Особенности SGD

- Низкая вычислительная сложность
- Экспоненциальная оценка
- Реализации во всех развитых библиотеках

$$y_j = f_j \left( \sum_{i=0,n} w_{ij} f_i \left( \sum_{k=0,m} w_{ki} x_k \right) \right)$$



# ПОИСК ПАРАМЕТРОВ НЕЙРОННОЙ СЕТИ

## Стохастический Градиентный спуск

$$a(X_j, W) = X_j W$$

$$1. \quad W_0$$

$$2. \quad Q = \frac{1}{n} \sum_{j=1}^n (X_j W - Y_j)^2$$

$$3. \quad X_j \in X$$

$$4. \quad e_j = (X_j W - Y_j)^2$$

$$5. \quad W^{t+1} = W^t - \eta \nabla e_j$$

$$6. \quad Q = \lambda e_j + (1 - \lambda) Q$$

7. проверка на  $Q, \Delta W : \langle true : end \quad false : \kappa \quad n.3 \rangle$

# ПОИСК ПАРАМЕТРОВ НЕЙРОННОЙ СЕТИ

## Стохастический Градиентный спуск

- Выпадает в локальные минимумы или седловые точки.
  - Производная на плато  $\approx$  нулю
  - Обрыв  $\rightarrow$  срыв.
  - Некоторые параметры обновляются значительно реже других
  - Склонность к переобучению.
- 
- маленькая скорость обучения — алгоритм сходиться долго и застревать в локальных минимумах,
  - Большая скорость обучения — мимо глобальные минимумы



# Функция потерь

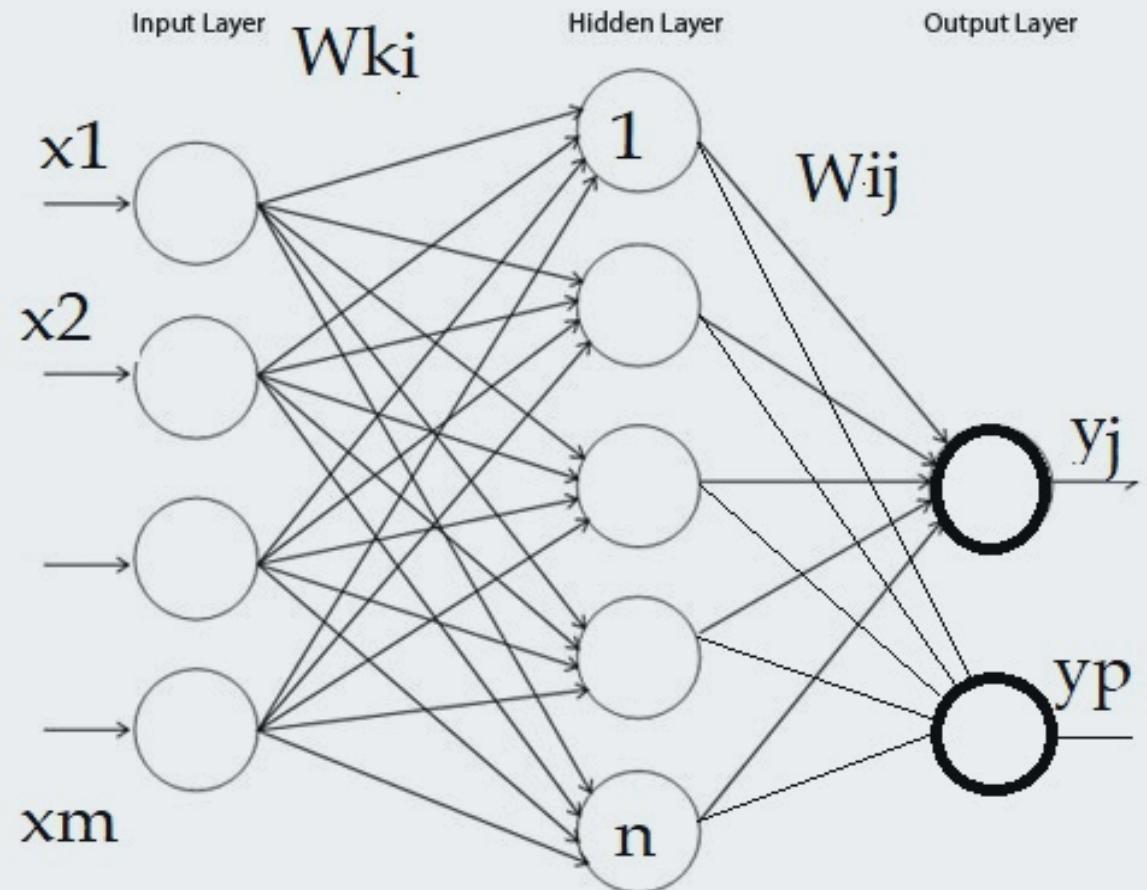
---

# Нейронная сеть

$$Q(W, X) = \sum_{i=1, N} L(X_i, W)$$

$$L(X_i, W) = (y_i(X, W) - \hat{y})^2$$

$$y_j = f_j \left( \sum_{i=0, n} w_{ij} f_i \left( \sum_{k=0, m} w_{ki} x_k \right) \right)$$



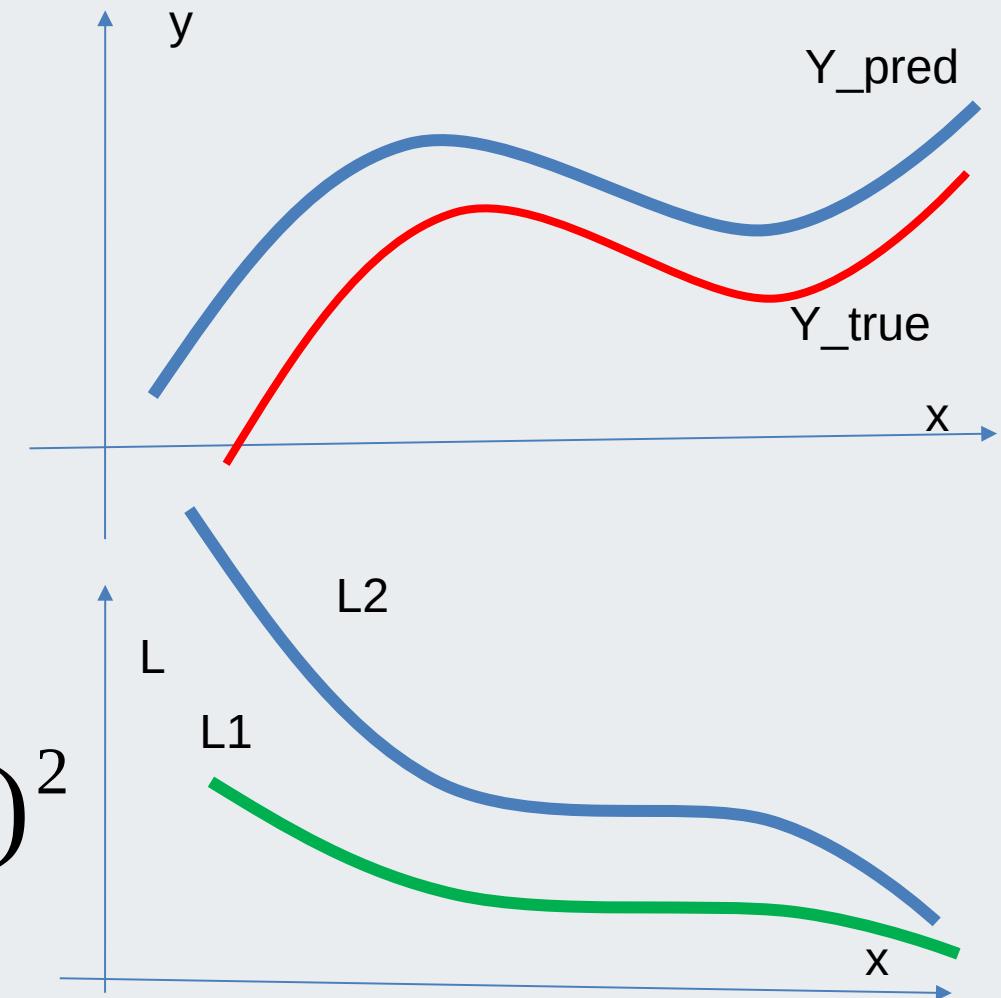
# Функция потерь

1. Абсолютная

$$L(X_i, W) = |y_i(X, W) - \hat{y}|$$

2. Квадратичная

$$L(X_i, W) = (y_i(X, W) - \hat{y})^2$$



# Функция потерь

3. Кросс - Энтропия - Бинарная \ Binary cross entropy

$$L(X_i, W) = -\frac{1}{N}(\hat{y} \cdot \log(y_i(X, W)) + (1 - \hat{y}) \cdot \log(1 - y_i(X, W)))$$

4. Взвешенная Кросс - Энтропия - Бинарная \ Weighted Binary cross entropy

$$L(X_i, W) = -\frac{1}{N}(\beta \cdot \hat{y} \cdot \log(y_i(X, W)) + (1 - \hat{y}) \cdot \log(1 - y_i(X, W)))$$

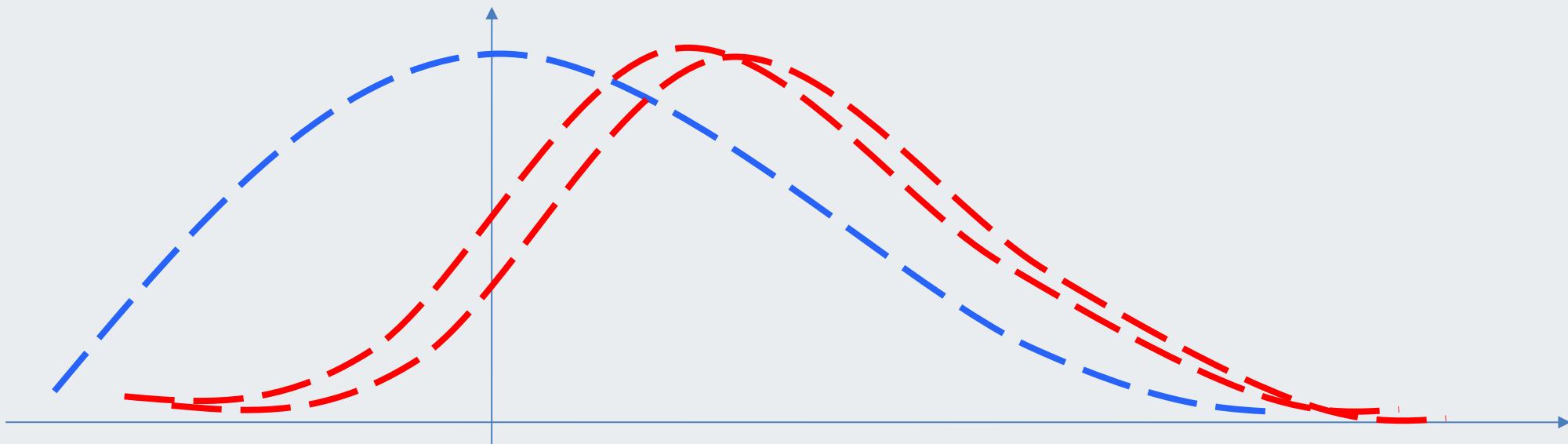
5. Balanced binary cross entropy

$$L(X_i, W) = -\frac{1}{N}(\beta \cdot \hat{y} \cdot \log(y_i(X, W)) + (1 - \beta)(1 - \hat{y}) \cdot \log(1 - y_i(X, W)))$$

# Функция потерь

3. Кросс - Энтропия - Бинарная \ Binary cross entropy

$$L(X_i, W) = -\frac{1}{N}(\hat{y} \cdot \log(y_i(X, W)) + (1 - \hat{y}) \cdot \log(1 - y_i(X, W)))$$



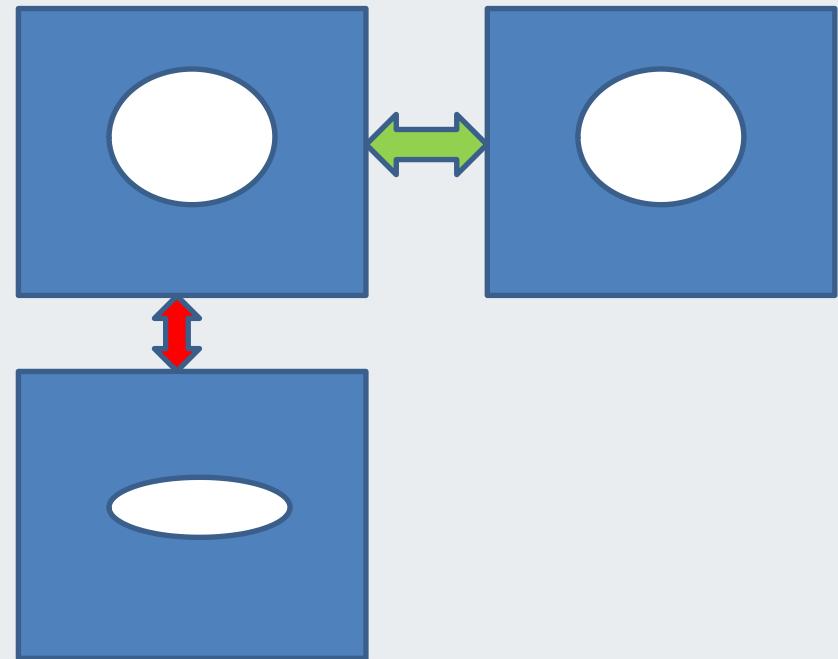
# Функция потерь

6. Дице коэффициент - Dice index

$$L(X, W) = 2 \frac{|y_i(X, W) \cap \hat{y}_i|}{|y_i(X, W)| + |\hat{y}_i|}$$

7. Jaccard loss - степень сходства

$$L(X_i, W) = \frac{\sum_{i=1, N} y_i(X, W) \cdot \hat{y}_i}{\sum_{i=1, N} y_i(X, W) + \sum_{i=1, N} \hat{y}_i - \sum_{i=1, N} y_i(X, W) \hat{y}_i}$$



---

# Регуляризация L1, L2





# ЗАДАЧИ РЕГУЛЯРИЗАЦИИ

---

- Сделать обучение более устойчивым к переобучению
- Лучше находить глобальный минимум функции оценки
- Снизить сложность модели

# СХЕМА РЕГУЛЯРИЗАЦИИ

---

$$\hat{Q}(X, W) = Q(X, W) + \lambda E(W, p)$$

$$W = \arg \min_W \hat{Q}(X, W)$$

$$\lambda = 0 : \hat{Q}(X, W) = Q(X, W)$$

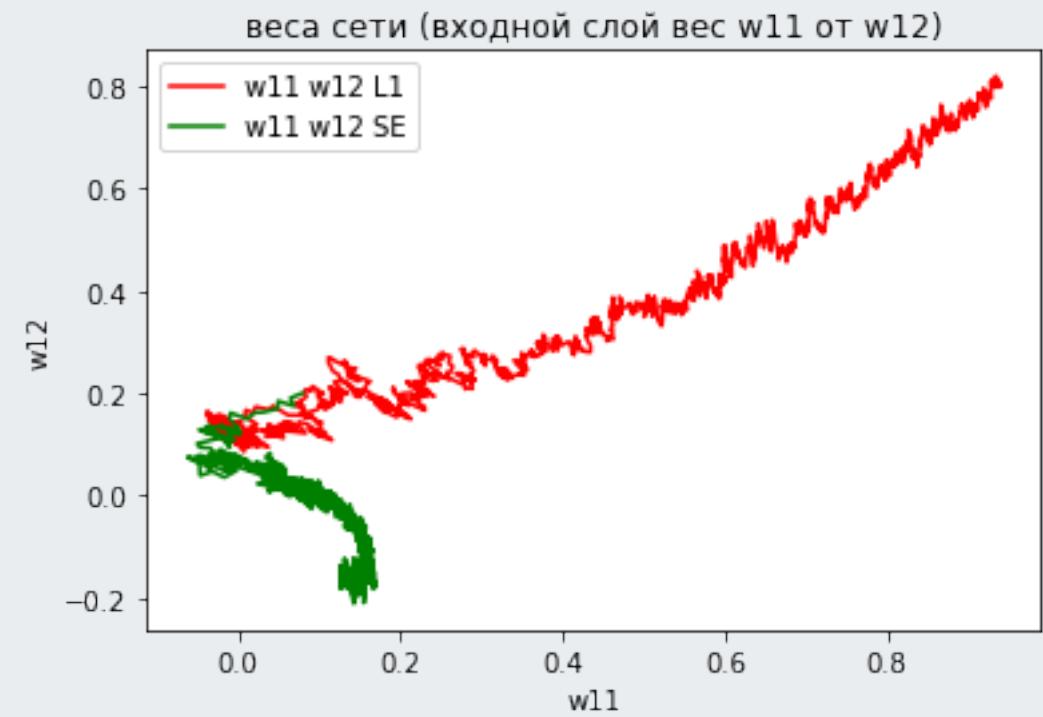
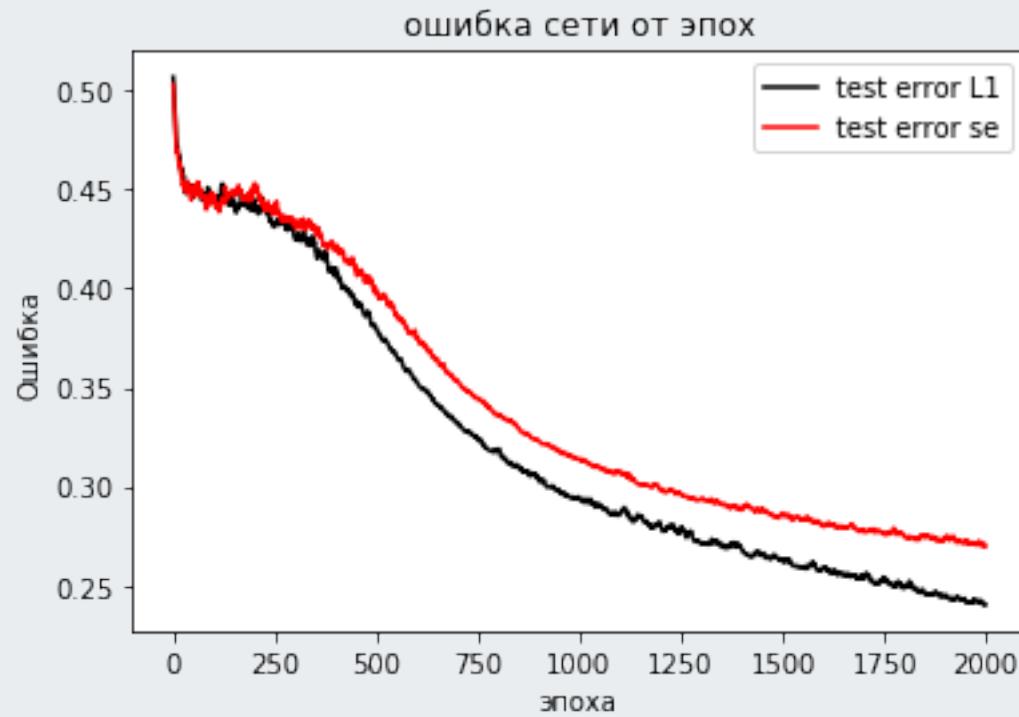
$$\lambda \gg 1 : \hat{Q}(X, W) \rightarrow \lambda E(W, p)$$



# Регуляризация L1

Точность test SE      78.67%  
Точность test SE+L1      85.33%

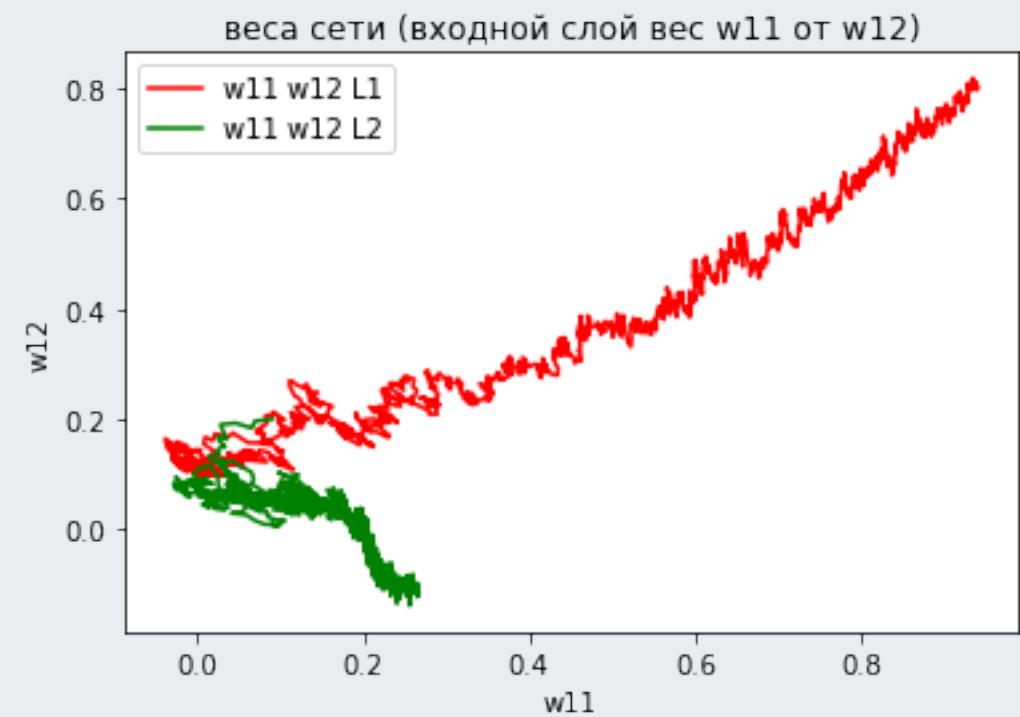
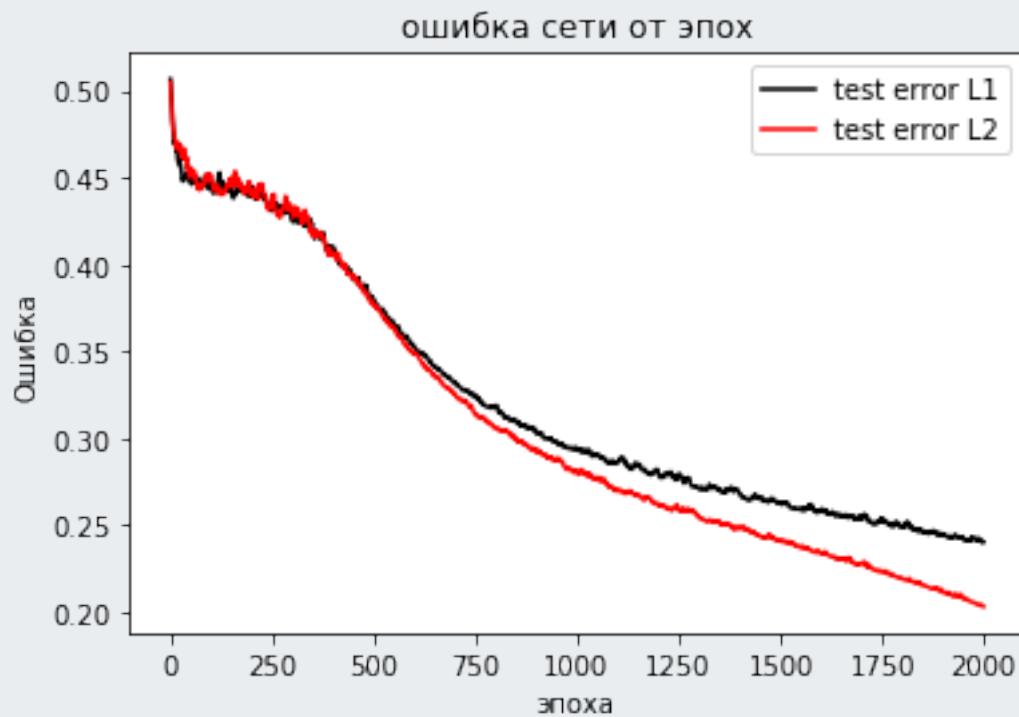
$$E(W, p=1) = |W|$$



# Регуляризация L2

Точность test SE	78.67%
Точность test SE+L1	85.33%
Точность test SE+L2	90.0%

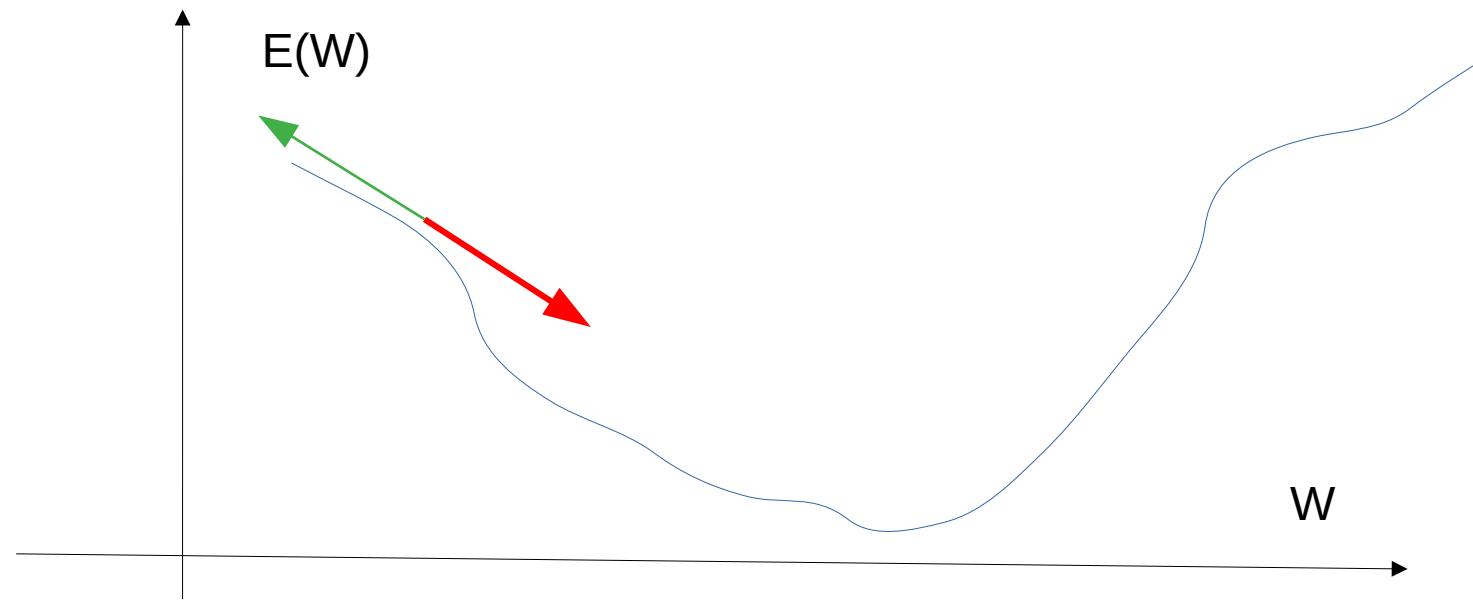
$$E(W, p=2) = \|W\|^2$$



---

# Модификации алгоритма адаптации параметров





# ПОИСК ПАРАМЕТРОВ НЕЙРОННОЙ СЕТИ

## Gradient Descent

$$a(X_j, W) = f(X_j W)$$

1.  $W_0$
2.  $X_j \in X$
3.  $e_j = (X_j W - Y_j)^2$
4.  $W^{t+1} = W^t - \eta \nabla e_j$
5.  $Q = \sum L(a(X_j, W), \hat{y})$
7. проверка на  $Q, \Delta W : \langle true : end \quad false : \kappa \quad n.2 \rangle$

# ПОИСК ПАРАМЕТРОВ НЕЙРОННОЙ СЕТИ

## Момент Нестерова

$$a(X_j, W) = X_j W$$

$$1. \quad W_0$$

$$2. \quad Q = \frac{1}{n} \sum_{j=1}^n (X_j W - Y_j)^2$$

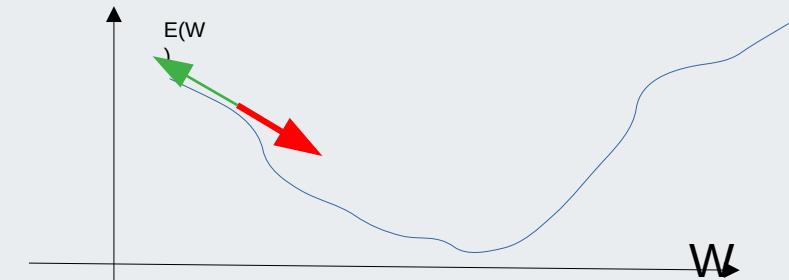
$$3. \quad X_j \in X$$

$$4. \quad L_j = (X_j W - Y_j)^2, m = m\gamma + \eta \nabla L_j$$

$$5. \quad W^{t+1} = W^t - m$$

$$6. \quad Q = \lambda e_j + (1 - \lambda) Q$$

7. проверка на  $Q, \Delta W : \langle true : end \quad false : \kappa \quad n.3 \rangle$



# ПОИСК ПАРАМЕТРОВ НЕЙРОННОЙ СЕТИ

## AdaGrad

$$a(X_j, W) = X_j W$$

$$1. \quad W_0$$

$$2. \quad Q = \frac{1}{n} \sum_{j=1}^n (X_j W - Y_j)^2$$

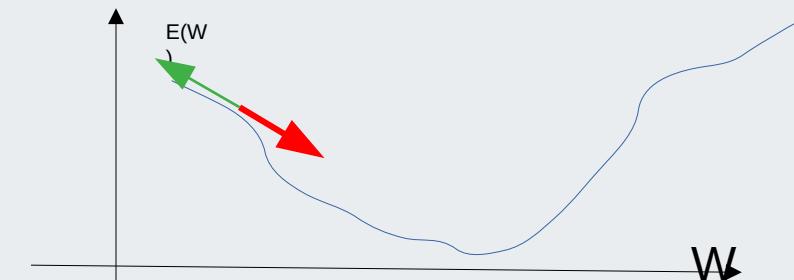
$$3. \quad X_j \in X$$

$$4. \quad L_j = (X_j W - Y_j)^2,$$

$$5. \quad W^{t+1} = W^t - \frac{\eta}{\sqrt{(\nabla L_j)^2 + \varepsilon}} \nabla L_j$$

$$6. \quad Q = \lambda e_j + (1 - \lambda) Q$$

7. проверка на  $Q$ ,  $\Delta W : \langle \text{true} : \text{end} \quad \text{false} : \kappa \quad n.3 \rangle$



# ПОИСК ПАРАМЕТРОВ НЕЙРОННОЙ СЕТИ

## Adam

$$a(X_j, W) = X_j W$$

$$1. \quad W_0$$

$$2. \quad Q = \frac{1}{n} \sum_{j=1}^n (X_j W - Y_j)^2$$

$$3. \quad X_j \in X$$

$$4. \quad L_j = (X_j W - Y_j)^2,$$

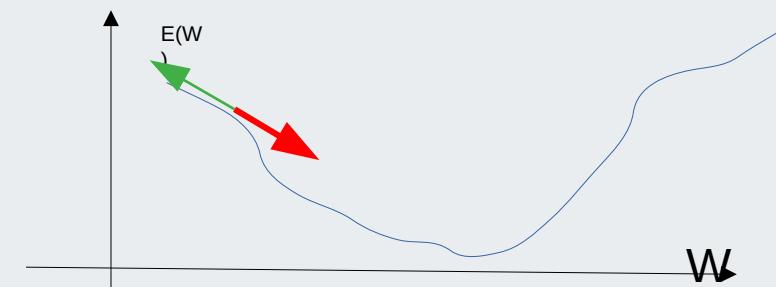
$$m = W^t \cdot \beta_1 + (1 - \beta_1) \nabla L_j$$

$$l = \Delta W^t \cdot \beta_2 + (1 - \beta_2) * (\nabla L_j)^2$$

$$5. \quad W^{t+1} = W^t - \frac{\eta}{\sqrt{\frac{l}{(1 - \beta_2^t)}}} \frac{m}{(1 - \beta_1^t)}$$

$$6. \quad Q = \lambda e_j + (1 - \lambda) Q$$

7. проверка на  $Q$ ,  $\Delta W : \langle true : end \quad false : \kappa \quad n.3 \rangle$



---

# Методы высоких порядков



# ПОИСК ПАРАМЕТРОВ НЕЙРОННОЙ СЕТИ

## Методы 2-го порядка

$$a(X_j, W) = X_j W$$

$$1. \quad W_0$$

$$2. \quad Q = \frac{1}{n} \sum_{j=1}^n (X_j W - Y_j)^2$$

$$3. \quad X_j \in X$$

$$4. \quad L_j = (X_j W - Y_j)^2$$

$$5. \quad W^{t+1} = W^t - \eta (\nabla^2)^{-1} \nabla L_j$$

$$6. \quad Q = \lambda e_j + (1 - \lambda) Q$$

$$7. \quad \text{проверка на } Q, \quad \Delta W : \langle \text{true} : \text{end} \quad \text{false} : \kappa \quad n.3 \rangle$$

[BFGS - https://habr.com/ru/post/333356/](https://habr.com/ru/post/333356/)

# ПОИСК ПАРАМЕТРОВ НЕЙРОННОЙ СЕТИ

Методы 2-го порядка

BFGS - <https://habr.com/ru/post/333356/>



# Резюме

1. Универсального алгоритма нет
2. Параметры нужно подбирать
3. Делаем несколько экспериментов в каждой конфигурации



# Резюме

1. SGD стоит использовать на небольших сбалансированных наборах данных.
2. Adam эффективен для решения задач на больших наборах данных.
3. Методы 2-го порядка (BFGS) — высокая эффективность для небольших наборов данных.

# ВОПРОСЫ ?

