

# Машинное обучение

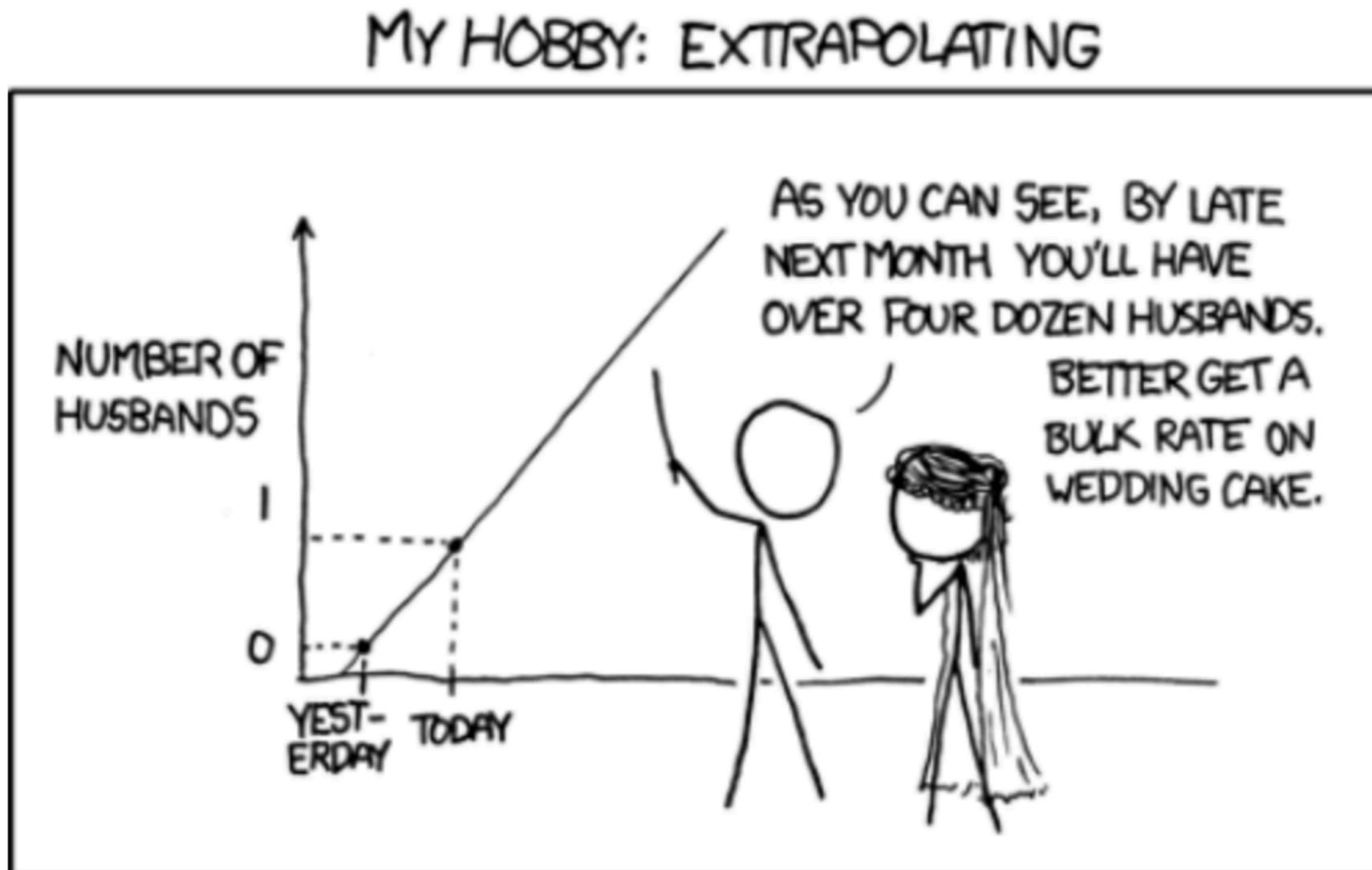
## Часть II

*Власов Кирилл Вячеславович*



2019

# Линейные модели



# Линейные модели

Линейная модель - взвешенная сумма признаков и член смещения (*bias term*), который также называют свободным членом (*intercept term*)

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

$\hat{y}$  Предсказываемое значение

$n$  Количество признаков

$x_1, x_2, \dots, x_n$  Значения признаков

$\theta_0, \theta_1, \theta_2, \dots, \theta_n$  Веса признаков и свободный член

Векторная форма:

$$\hat{y} = h_{\theta}(x) = \theta^T \cdot x$$

$\theta$  Вектор весов и свободный член

$x$  Вектор значений признаков примера, где  $x_0 = 0$

# Линейные модели

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

Средняя квадратичная ошибка (Mean Squared Error, MSE):

$$MSE = \frac{1}{l} \sum_{i=1}^l (f(x_i) - y_i)^2$$

$$MSE = \frac{1}{l} \sum_{i=1}^l (\hat{y}^{(i)} - y_i)^2 \rightarrow \frac{1}{l} \sum_{i=1}^l (\theta^T \cdot x^{(i)} - y_i)$$

# Метод наименьших квадратов

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

Средняя квадратичная ошибка (Mean Squared Error, MSE):

$$MSE = \frac{1}{l} \sum_{i=1}^l (f(x_i) - y_i)^2$$

$$MSE = \frac{1}{l} \sum_{i=1}^l (\hat{y}^{(i)} - y_i)^2 \rightarrow \frac{1}{l} \sum_{i=1}^l (\theta^T \cdot x^{(i)} - y_i)^2 \rightarrow \min$$

Аналитический способ поиска оптимальных весов (нормальное уравнение):

$$\hat{\theta} = (X^T \cdot X)^{-1} \cdot X^T \cdot y$$

$X$  Матрица объектов признаков

$y$  Вектор целевой переменной

$\hat{\theta}$  Оптимальный вектор весов, который сводит к минимуму MSE

Подробный и понятный вывод: [Открытый курс машинного обучения: Тема 4](#)

# Метод наименьших квадратов

## Теорема Гаусса – Маркова

оценки [метода наименьших квадратов](#) оптимальны в классе линейных несмешённых оценок

*Условия для парной регрессии:*

1. модель данных правильно специфицирована. Нет лишних переменных, или учтены все важные  $Y = \theta_0 + \theta \cdot X + \epsilon$
2. все  $X$  детерминированы и не все равны между собой. Иными словами, переменные не должны быть постоянными.
3. Ошибки не носят систематического характера, то есть  $E(\epsilon_i) = 0 \forall i$
4. Дисперсия ошибок одинакова (гомоскедастичность) и равна некоторой  $\sigma^2 = const$
5. Ошибки некоррелированы, то есть  $cov(\epsilon_i, \epsilon_j) = 0 \forall i, j$

*Условия для Множественной регрессии:*

1. модель данных правильно специфицирована. Нет лишних переменных, или учтены все важные
2.  $rang(X) = m$
3.  $E(\epsilon_i) = 0 \forall i$
4.  $cov(\epsilon_i, \epsilon_j) = 0 \forall i, j$

# Реализация в python

# Проблемы нормального уравнения

$$\hat{\theta} = (X^T \cdot X)^{-1} \cdot X^T \cdot y$$

# Проблемы нормального уравнения

$$\hat{\theta} = (X^T \cdot X)^{-1} \cdot X^T \cdot y$$

$(X^T \cdot X)$  – матрица размером  $n \times n$ , где  $n$  – количество признаков

Вычислительная сложность для обратной матрицы:  $O(n^3)$

# Градиентный спуск

$$MSE = \frac{1}{l} \sum_{i=1}^l (\hat{y}^{(i)} - y_i)^2 \rightarrow \frac{1}{l} \sum_{i=1}^l (\theta^T \cdot x^{(i)} - y_i)^2 \rightarrow \min$$

# Градиентный спуск

$$MSE = \frac{1}{l} \sum_{i=1}^l (\hat{y}^{(i)} - y_i)^2 \rightarrow \frac{1}{l} \sum_{i=1}^l (\theta^T \cdot x^{(i)} - y_i)^2 \rightarrow \min$$

*Градиент - Вектор указывающий направление наибольшего возрастания функции, компоненты которого равны частным производным по всем её аргументам.*

$$\nabla \varphi = \left( \frac{\partial \varphi}{\partial x_1}, \frac{\partial \varphi}{\partial x_2}, \dots, \frac{\partial \varphi}{\partial x_n} \right)$$

# Градиентный спуск

$$MSE = \frac{1}{l} \sum_{i=1}^l (\hat{y}^{(i)} - y_i)^2 \rightarrow \frac{1}{l} \sum_{i=1}^l (\theta^T \cdot x^{(i)} - y_i)^2 \rightarrow \min$$

*Градиент - Вектор указывающий направление наибольшего возрастания функции, компоненты которого равны частным производным по всем её аргументам.*

$$\nabla \varphi = \left( \frac{\partial \varphi}{\partial x_1}, \frac{\partial \varphi}{\partial x_2}, \dots, \frac{\partial \varphi}{\partial x_n} \right)$$

$$\frac{\partial MSE(\theta)}{\partial \theta_j} = \frac{2}{l} \sum_{i=1}^l (\theta^T \cdot x^{(i)} - y^{(i)}) \cdot x_j^{(i)}$$
$$\nabla MSE(\theta) = \begin{pmatrix} \frac{\partial MSE(\theta)}{\partial \theta_0} \\ \frac{\partial MSE(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial MSE(\theta)}{\partial \theta_n} \end{pmatrix} = \frac{2}{l} X^T \cdot (X \cdot \theta - y)$$

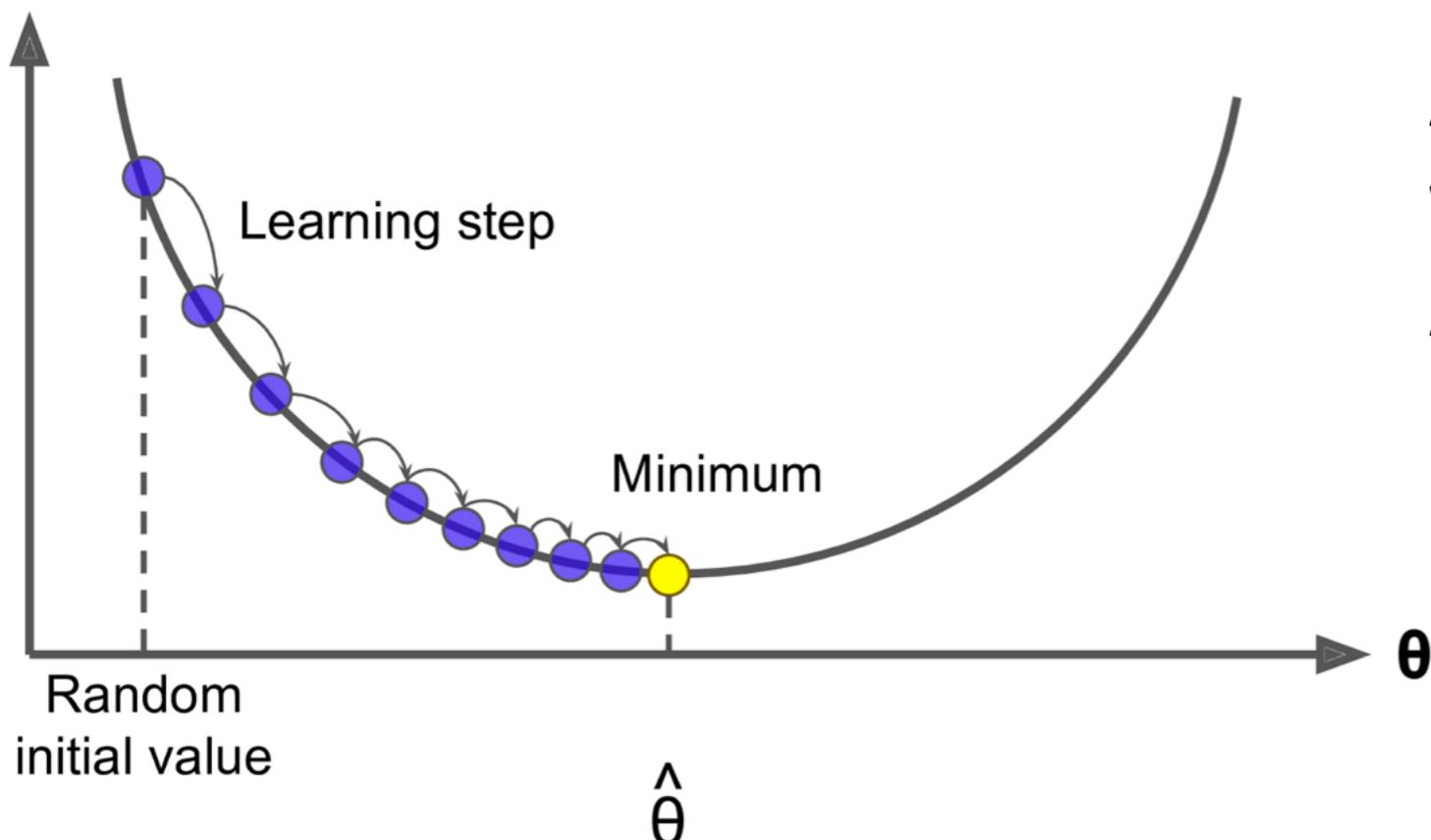
# Градиентный спуск

$$MSE = \frac{1}{l} \sum_{i=1}^l (\hat{y}^{(i)} - y_i)^2 \rightarrow \frac{1}{l} \sum_{i=1}^l (\theta^T \cdot x^{(i)} - y_i)^2 \rightarrow \min$$

Градиент - Вектор указывающий направление наибольшего возрастания функции, компоненты которого равны частным производным по всем её аргументам.

$$\nabla MSE(\theta) = \frac{2}{l} X^T \cdot (X \cdot \theta - y)$$

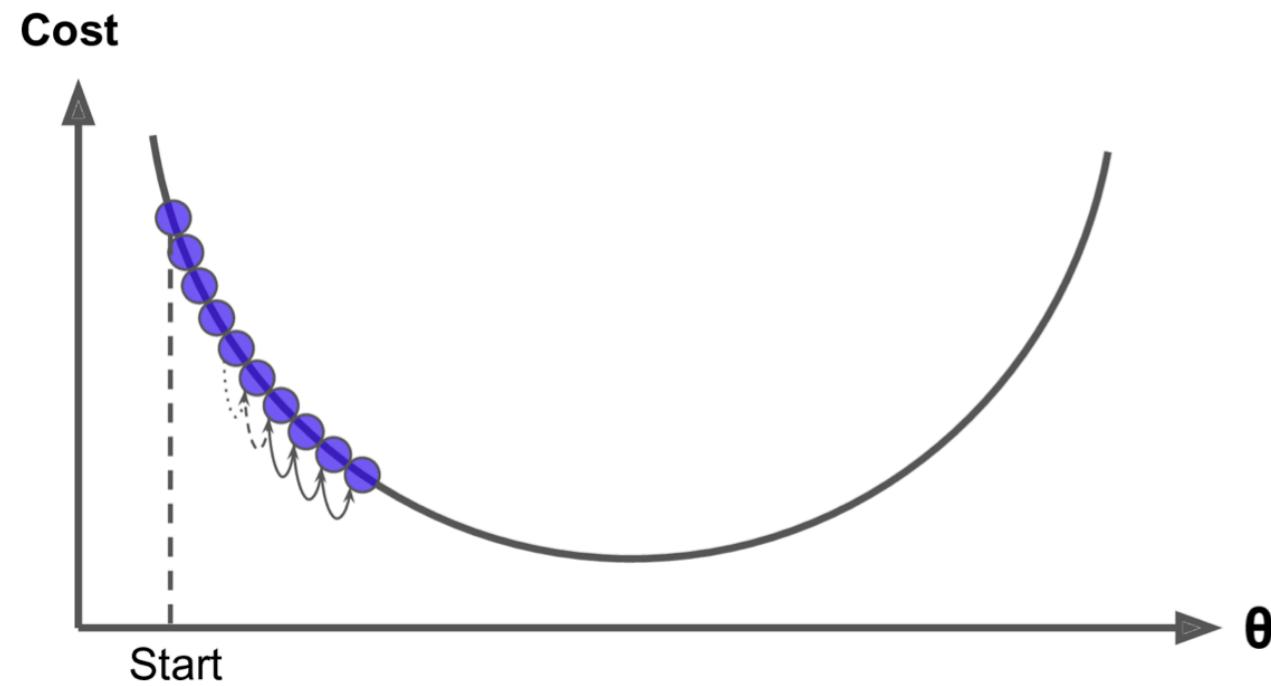
Cost



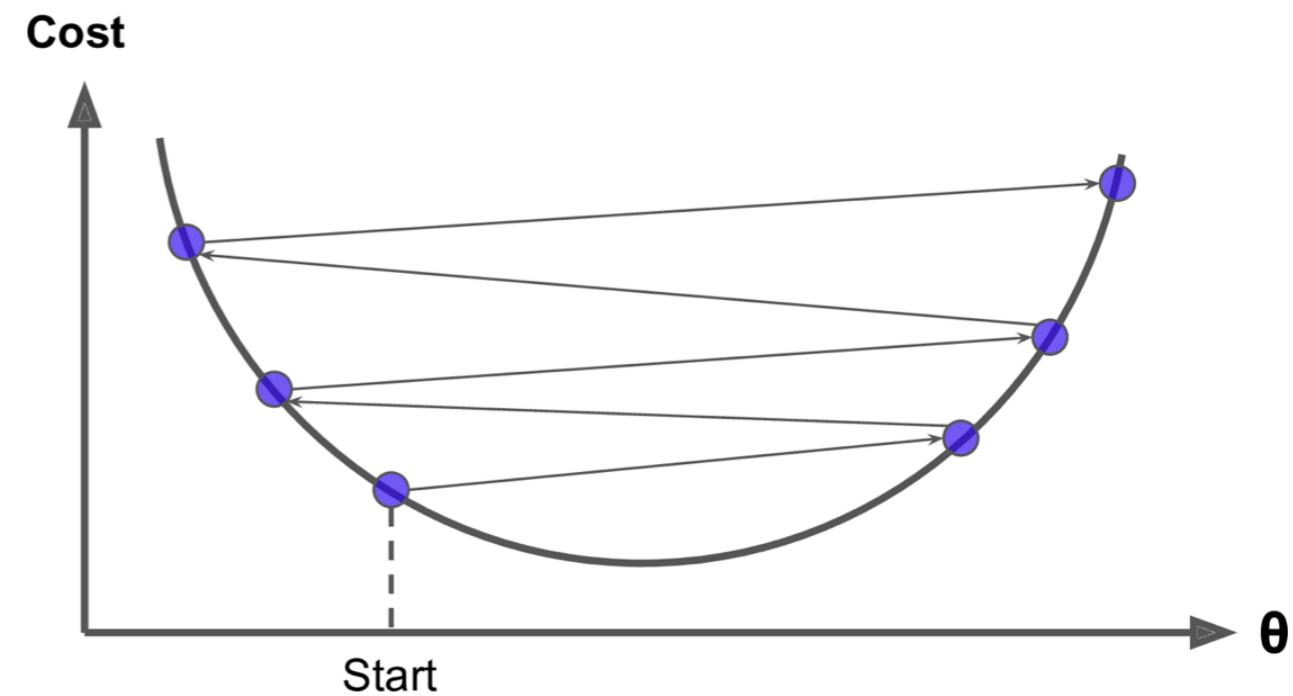
1. Случайно задаем веса
2. Считаем градиент в точке
3. Изменяем веса путем вычитания градиента
4. Повторяем п.2

\* Мы можем контролировать скорость обучения (learning rate) умножая градиент на шаг обучения

# Выбор шага обучения градиентного спуска



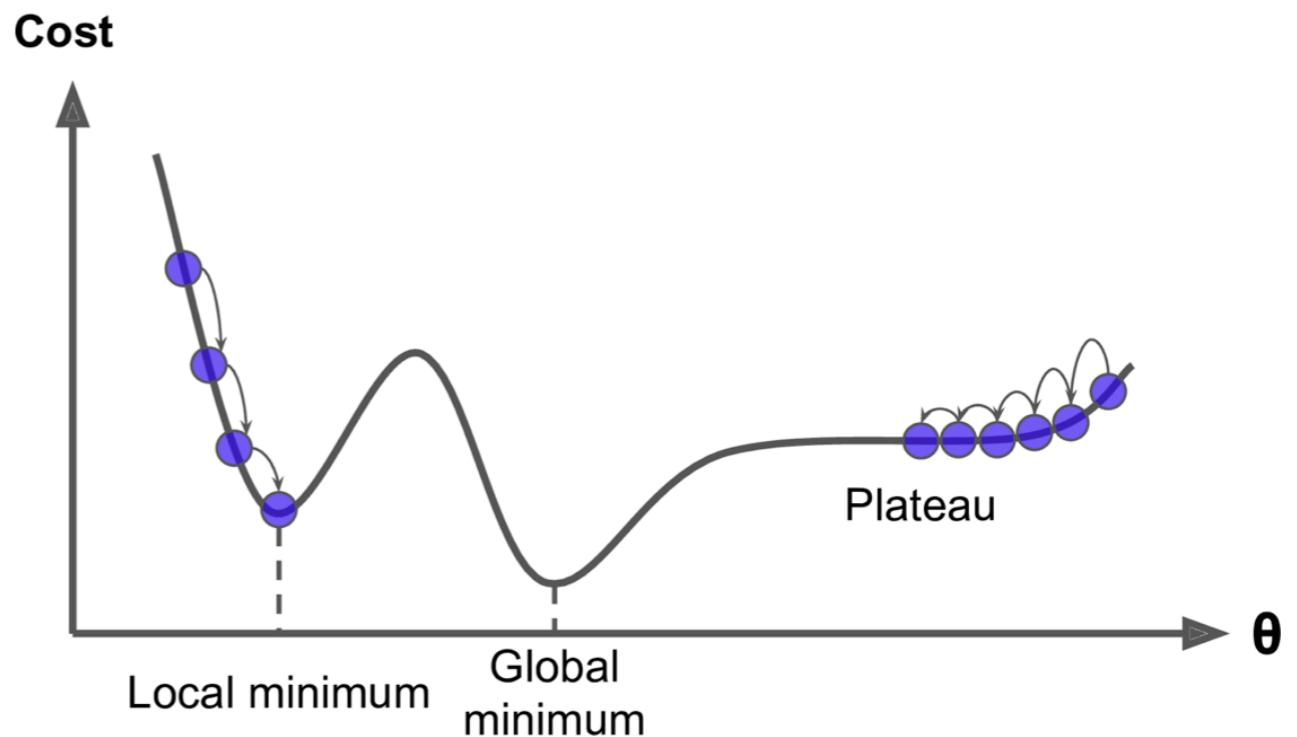
**Слишком маленький шаг обучения**  
рискуем не дойти до минимума



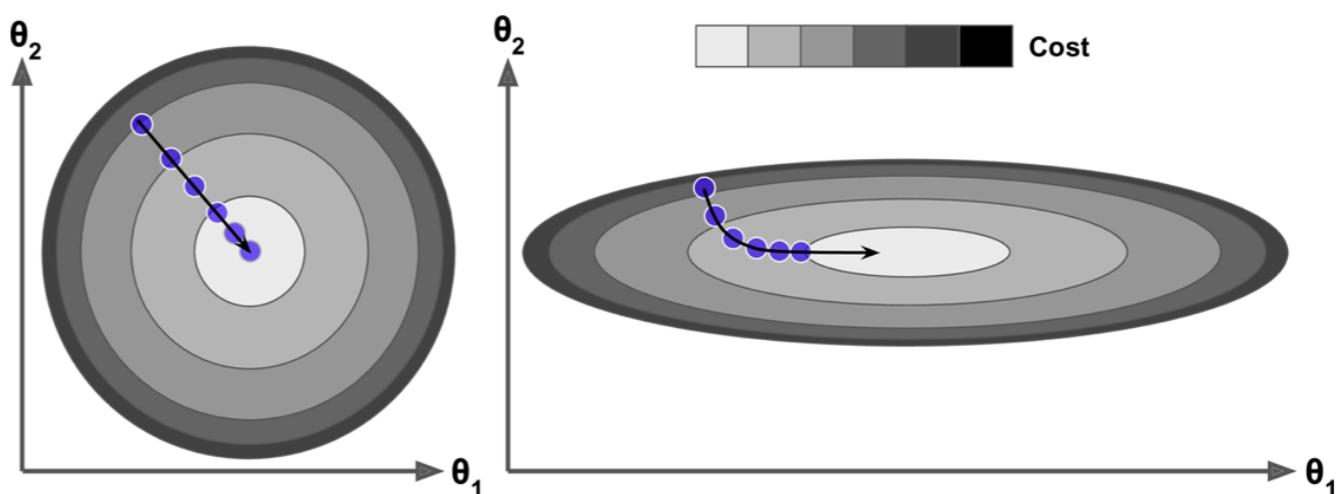
**Слишком большой шаг обучения**  
рискуем проскочить минимум

# Другие проблемы градиентного спуска

Не все функции потерь  
одинаково полезны  
имеют форму чаши  
(параболоид)



Важно масштабировать  
данные



# Реализация в python

# Стochastic gradient descent

$$MSE = \frac{1}{l} \sum_{i=1}^l (\hat{y}^{(i)} - y_i)^2 \rightarrow \frac{1}{l} \sum_{i=1}^l (\theta^T \cdot x^{(i)} - y_i)^2 \rightarrow \min$$

$$\frac{\partial MSE(\theta)}{\partial \theta_j} = \frac{2}{l} \sum_{i=1}^l (\theta^T \cdot x^{(i)} - y^{(i)}) \cdot x_j^{(i)}$$

# Стochastic gradient descent

$$MSE = \frac{1}{l} \sum_{i=1}^l (\hat{y}^{(i)} - y_i)^2 \rightarrow \frac{1}{l} \sum_{i=1}^l (\theta^T \cdot x^{(i)} - y_i)^2 \rightarrow \min$$

*Пакетный градиентный спуск*

$$\frac{\partial MSE(\theta)}{\partial \theta_j} = \frac{2}{l} \sum_{i=1}^l (\theta^T \cdot x^{(i)} - y^{(i)}) \cdot x_j^{(i)}$$
$$\frac{\partial MSE(\theta)}{\partial \theta_j} = (\theta^T \cdot x^{(i)} - y^{(i)}) \cdot x^{(i)}$$

$$\nabla MSE(\theta) = x_i^T \cdot (x_i \cdot \theta - y)$$

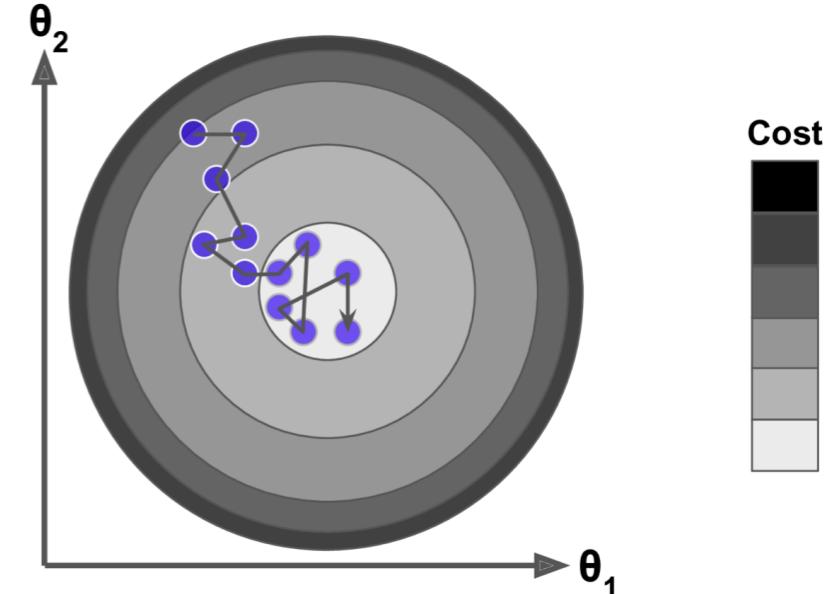
# Стохастический градиентный спуск

$$MSE = \frac{1}{l} \sum_{i=1}^l (\hat{y}^{(i)} - y_i)^2 \rightarrow \frac{1}{l} \sum_{i=1}^l (\theta^T \cdot x^{(i)} - y_i)^2 \rightarrow \min$$

$$\frac{\partial MSE(\theta)}{\partial \theta_j} = \frac{2}{l} \sum_{i=1}^l (\theta^T \cdot x^{(i)} - y^{(i)}) \cdot x_j^{(i)}$$

$$\frac{\partial MSE(\theta)}{\partial \theta_j} = (\theta^T \cdot x^{(i)} - y^{(i)}) \cdot x^{(i)}$$

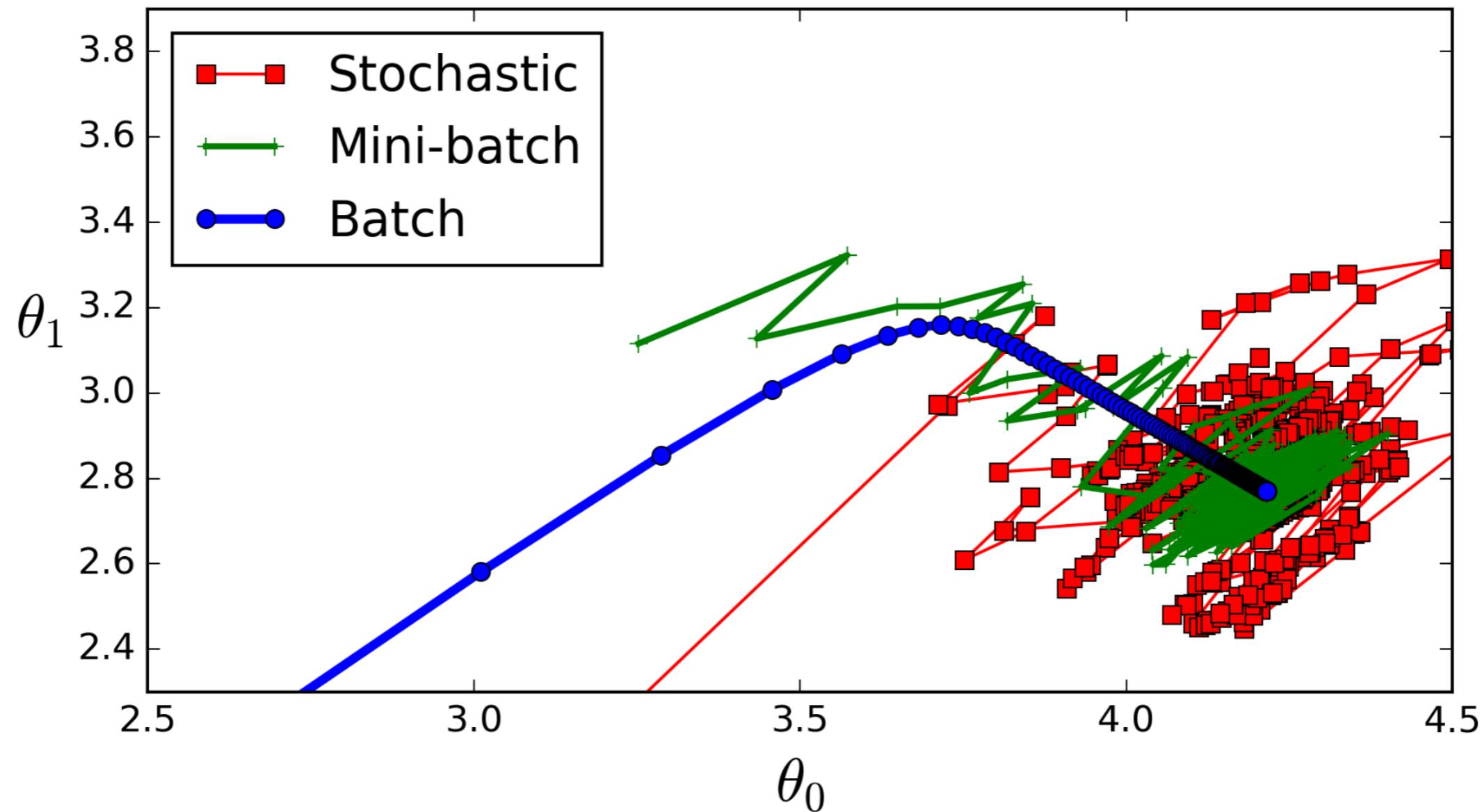
$$\nabla MSE(\theta) = x_i^T \cdot (x_i \cdot \theta - y)$$



Случайно выбираем объект, и двигаемся к минимуму.  
Каждый объект выборки может пройти несколько раз или быть не выбран вообще

# Выбор метода градиентного спуска

Можно скрестить два похода: Стохастический и пакетный и выбирать случайные подборки например из 100 объектов, тогда получится: метод называемый Mini-batch



# Регуляризация линейных моделей

1. Одно из условий Гаусса-Маркова:  $\text{rang}(X) = m$

Если оно не выполняется, то решение МНК  $\hat{\theta} = (X^T \cdot X)^{-1} \cdot X^T \cdot y$  не существует  
так как Матрица  $X^T \cdot X$  сингулярна (вырождена)

Значит нам нужно сделать так, чтобы сделать матрицу вырожденной (регулярной)

2. Мультиколлинеарность

Собственные значения будут стремиться к 0, а при обращении матрицы к  $\infty$

# Регуляризация линейных моделей

L2 - регуляризация, гребневая регрессия, ridge

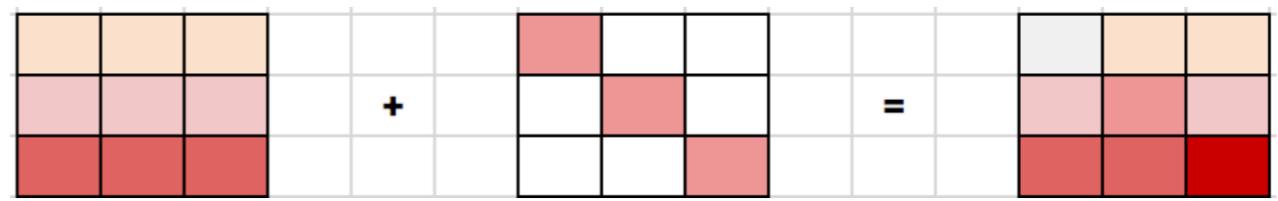
$$MSE(\theta) + \alpha \frac{1}{2} \sum_{i=1}^n \theta_i^2 \rightarrow \min$$

A diagram illustrating matrix addition and regularization. It shows three 3x3 matrices on a grid. The first matrix has orange, pink, and red cells. The second matrix has red cells. The third matrix has light blue, pink, and red cells. A plus sign (+) is between the first and second matrices, and an equals sign (=) is to the right of the second matrix.

# Регуляризация линейных моделей

L2 - регуляризация, гребневая регрессия, ridge

$$MSE(\theta) + \alpha \frac{1}{2} \sum_{i=1}^n \theta_i^2 \rightarrow \min$$



L1 - регуляризация, Lasso (Least absolute shrinkage and selection operator)

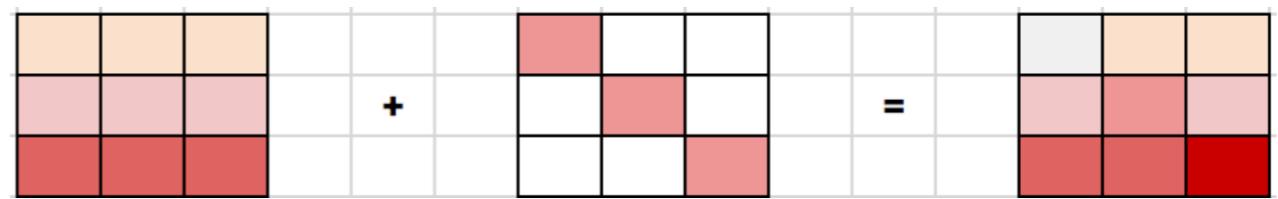
$$MSE(\theta) + \alpha \sum_{i=1}^n |\theta_i| \rightarrow \min$$

введение ограничений на норму вектора коэффициентов модели приводит к обращению в 0 некоторых коэффициентов модели.

# Регуляризация линейных моделей

L2 - регуляризация, гребневая регрессия, ridge

$$MSE(\theta) + \alpha \frac{1}{2} \sum_{i=1}^n \theta_i^2 \rightarrow \min$$



L1 - регуляризация, Lasso (Least absolute shrinkage and selection operator)

$$MSE(\theta) + \alpha \sum_{i=1}^n |\theta_i| \rightarrow \min$$

введение ограничений на норму вектора коэффициентов модели приводит к обращению в 0 некоторых коэффициентов модели.

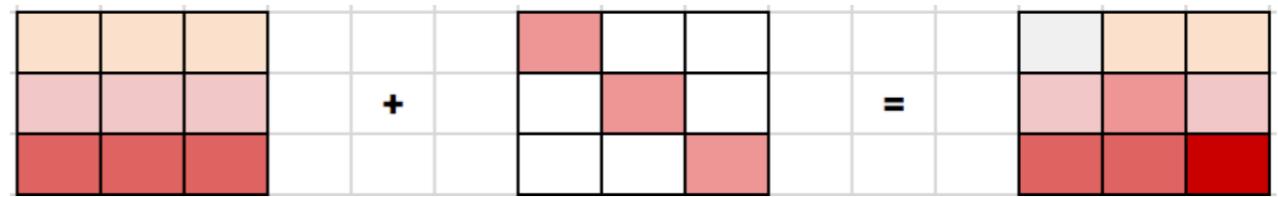
ElasticNet

$$MSE(\theta) + \alpha r \sum_{i=1}^n |\theta_i| + \alpha \frac{1-r}{2} \sum_{i=1}^n \theta_i^2 \rightarrow \min$$

# Регуляризация линейных моделей

L2 - регуляризация, гребневая регрессия, ridge

$$MSE(\theta) + \alpha \frac{1}{2} \sum_{i=1}^n \theta_i^2 \rightarrow \min$$



L1 - регуляризация, Lasso (Least absolute shrinkage and selection operator)

$$MSE(\theta) + \alpha \sum_{i=1}^n |\theta_i| \rightarrow \min$$

введение ограничений на норму вектора коэффициентов модели приводит к обращению в 0 некоторых коэффициентов модели.

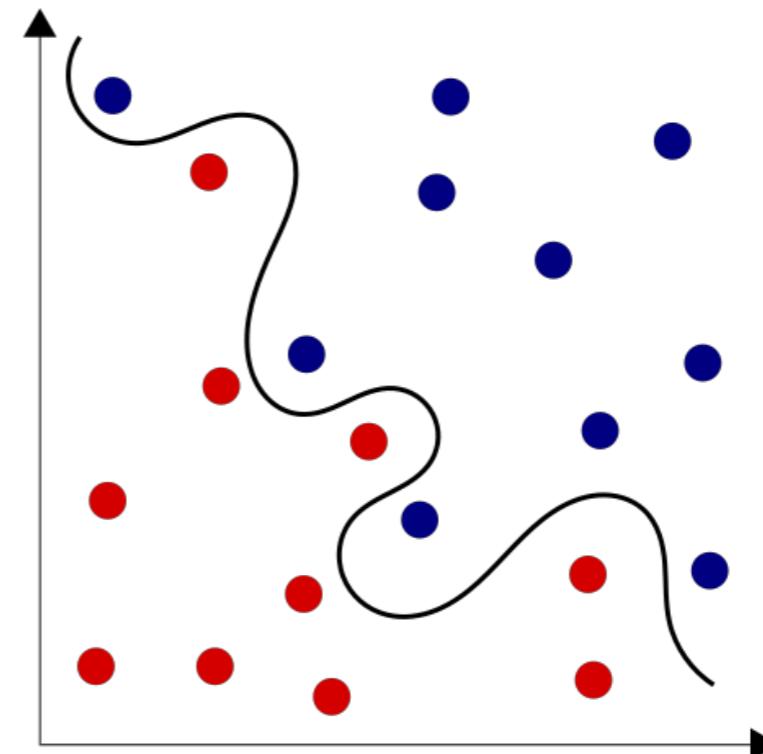
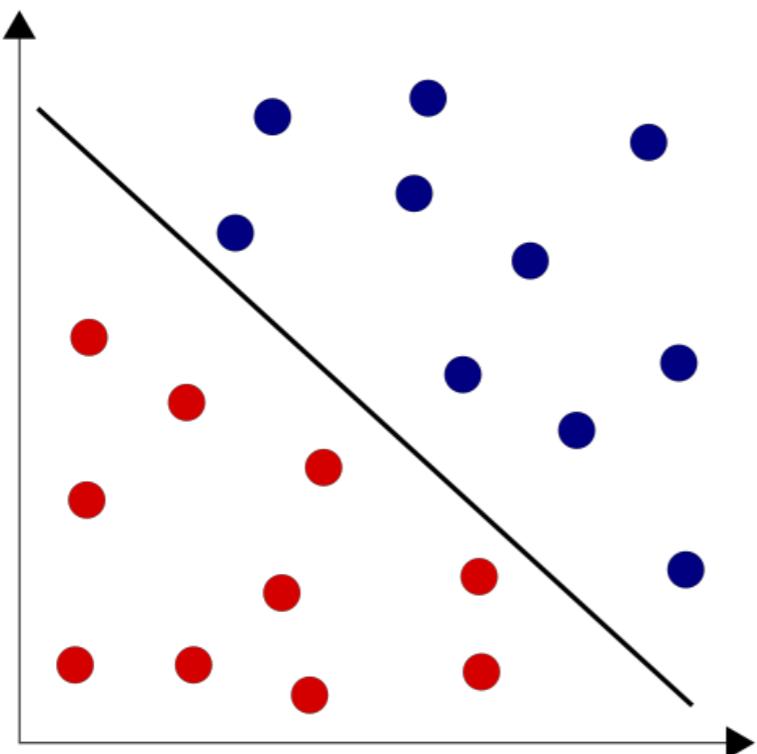
ElasticNet

$$MSE(\theta) + \alpha r \sum_{i=1}^n |\theta_i| + \alpha \frac{1-r}{2} \sum_{i=1}^n \theta_i^2 \rightarrow \min$$

# Линейные модели в задаче классификации

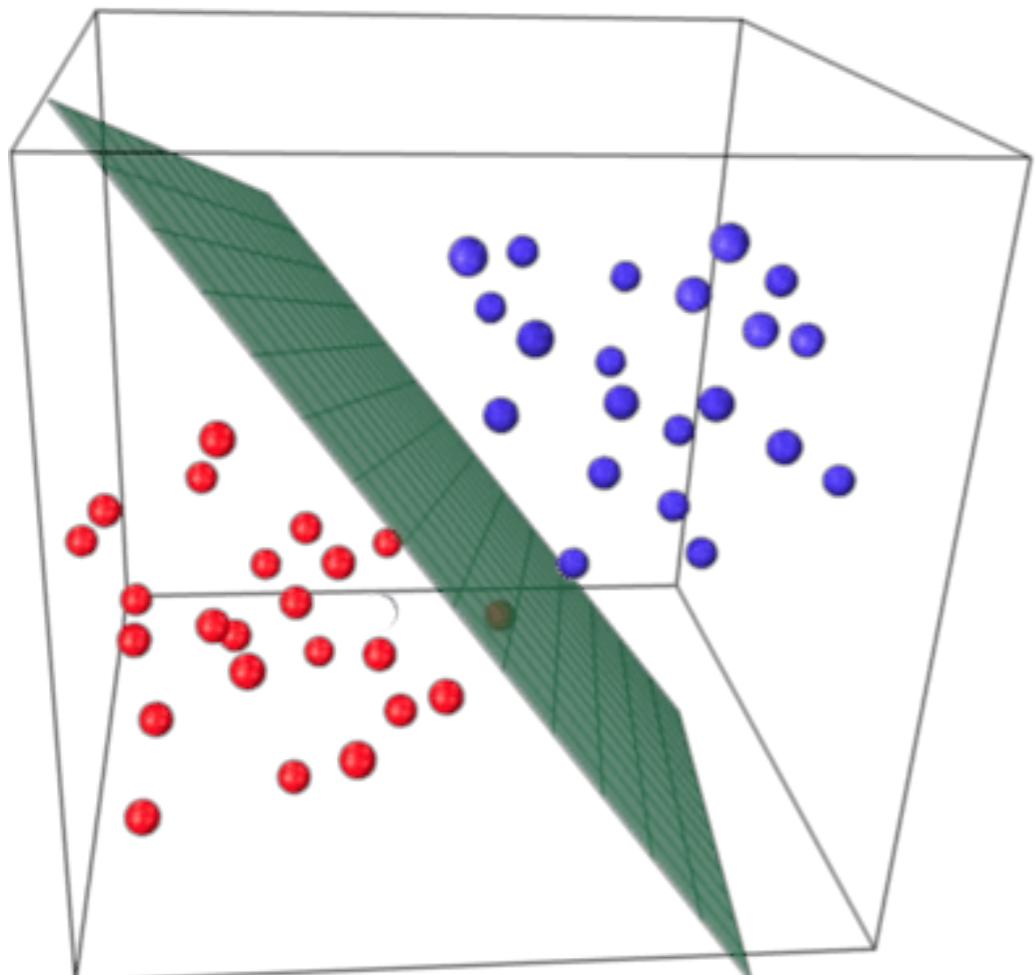
## Основная идея:

Предполагаем, что существует такая гиперплоскость, которая делит пространство на два полупространства в каждом из которых одно из двух значений целевого класса.



Если существует гиперплоскость которой можно разделить пространство на два класса без ошибок, то обучающая выборка называется *линейно разделимой*

# Линейные модели в задаче классификации



Дана обучающая выборка:

$$X_l = \{ (x_1, y_1), \dots, (x_l, y_l) \}$$

Для задачи классификации - Целевая  
переменная задана конечным числом меток

$$(x_1, y_1) \in \mathbb{R}^m \times \mathbb{Y}, \mathbb{Y} = \{-1; 1\}$$

Простейший классификатор:

$$a(x) = sign(\langle w, x \rangle + x_0) = sign(\vec{w}^T \cdot x)$$

$\vec{w}$  – нормаль гиперплоскости

$\vec{w}^T \cdot x_i$  – расстояние от гиперплоскости до  $x_i$ ,  
знак показывает отношение к классу

# Линейные модели в задаче классификации

$$a(x) = \text{sign}(\langle w, x \rangle + x_0) = \text{sign}(\vec{w}^T \cdot x)$$

доля правильных ответов (accuracy):

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l [a(x_i) = y_i]$$

# Линейные модели в задаче классификации

$$a(x) = \text{sign}(\langle w, x \rangle + x_0) = \text{sign}(\vec{w}^T \cdot x)$$

доля правильных ответов (accuracy):

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l [a(x_i) = y_i] \rightarrow \max$$

# Линейные модели в задаче классификации

$$a(x) = \text{sign}(\langle w, x \rangle + x_0) = \text{sign}(\vec{w}^T \cdot x)$$

доля правильных ответов (accuracy):

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l [a(x_i) = y_i] \rightarrow \max$$

доля неправильных ответов:

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l [a(x_i) \neq y_i] = \frac{1}{l} \sum_{i=1}^l [\text{sign}(\langle w, x_i \rangle) \neq y_i] \rightarrow \min$$

Проблемы?

# Линейные модели в задаче классификации

$$a(x) = \text{sign}(\langle w, x \rangle + x_0) = \text{sign}(\vec{w}^T \cdot x)$$

доля правильных ответов (accuracy):

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l [a(x_i) = y_i] \rightarrow \max$$

доля неправильных ответов:

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l [a(x_i) \neq y_i] = \frac{1}{l} \sum_{i=1}^l [\text{sign}(\langle w, x_i \rangle) \neq y_i] \rightarrow \min$$

## Проблемы:

1. Функционал дискретный относительно весов  $\Rightarrow$  мы не сможем искать минимум с помощью градиентных методов.
2. Функционал может иметь несколько глобальных минимумов  $\Rightarrow$  может быть много способов добиться оптимального количества ошибок.

# Линейные модели в задаче классификации

$$a(x) = \text{sign}(\langle w, x \rangle + x_0) = \text{sign}(\vec{w}^T \cdot x)$$

доля правильных ответов (accuracy):

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l [a(x_i) = y_i] \rightarrow \max$$

доля неправильных ответов:

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l [a(x_i) \neq y_i] = \frac{1}{l} \sum_{i=1}^l [\text{sign}(\langle w, x_i \rangle) \neq y_i] \rightarrow \min$$

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l \underbrace{[y_i \langle w, x_i \rangle < 0]}_{M_i} \rightarrow \min \quad M_i = y_i \langle w, x_i \rangle \text{ -- отступ (margin)}$$

Знак отступа говорит о корректности ответа классификатора  
(положительный отступ соответствует правильному ответу,  
отрицательный неправильному)  
абсолютная величина  $M$  – характеризует степень уверенности  
классификатора в своём ответе.

# Линейные модели в задаче классификации

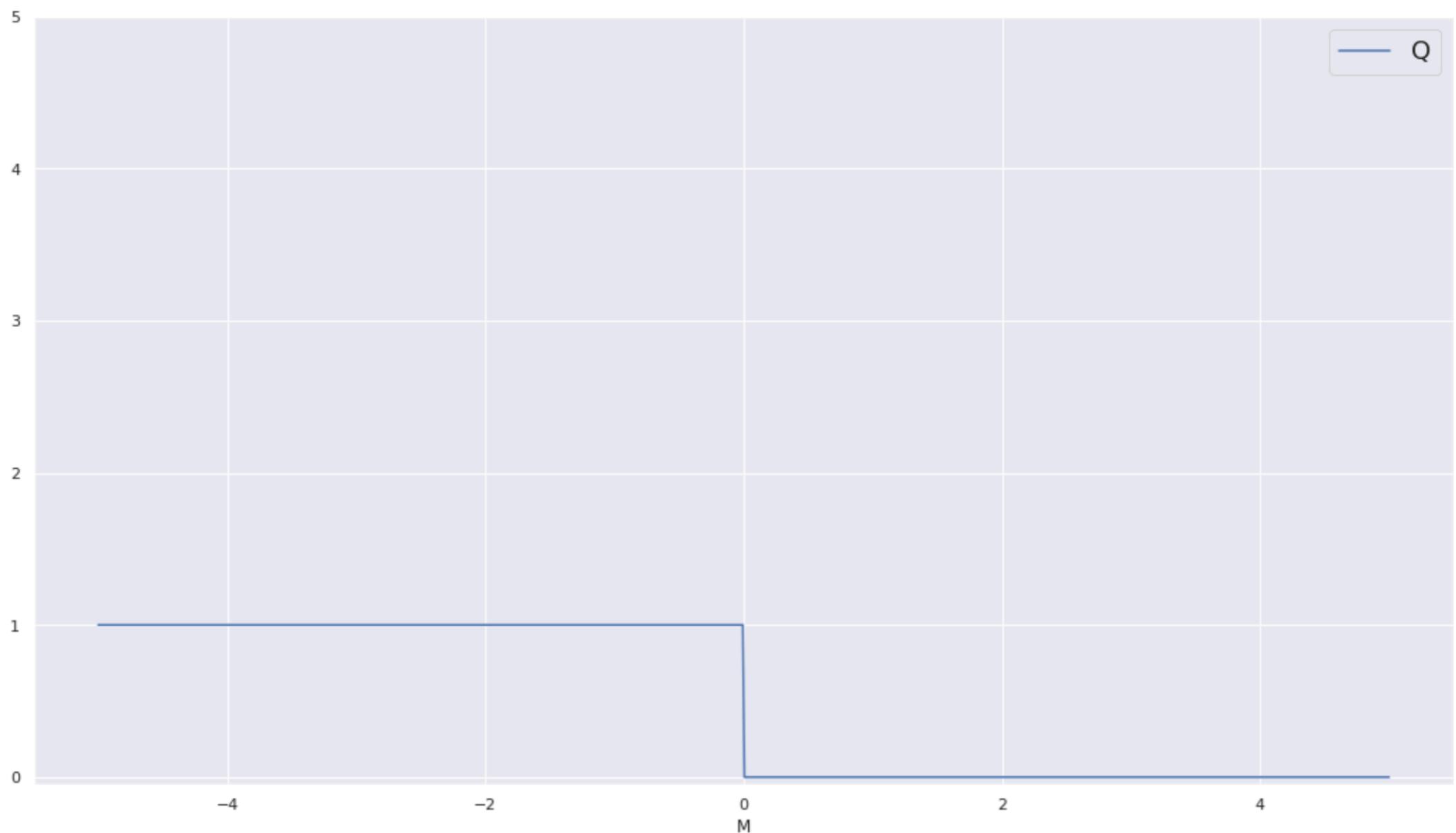
$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l \underbrace{y_i \langle w, x_i \rangle}_{M_i} < 0 \rightarrow \min$$

$$L(M) = [M < 0]$$

# Линейные модели в задаче классификации

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l \underbrace{[y_i \langle w, x_i \rangle < 0]}_{M_i} \rightarrow \min$$

$L(M) = [M < 0]$  – пороговая функции потерь



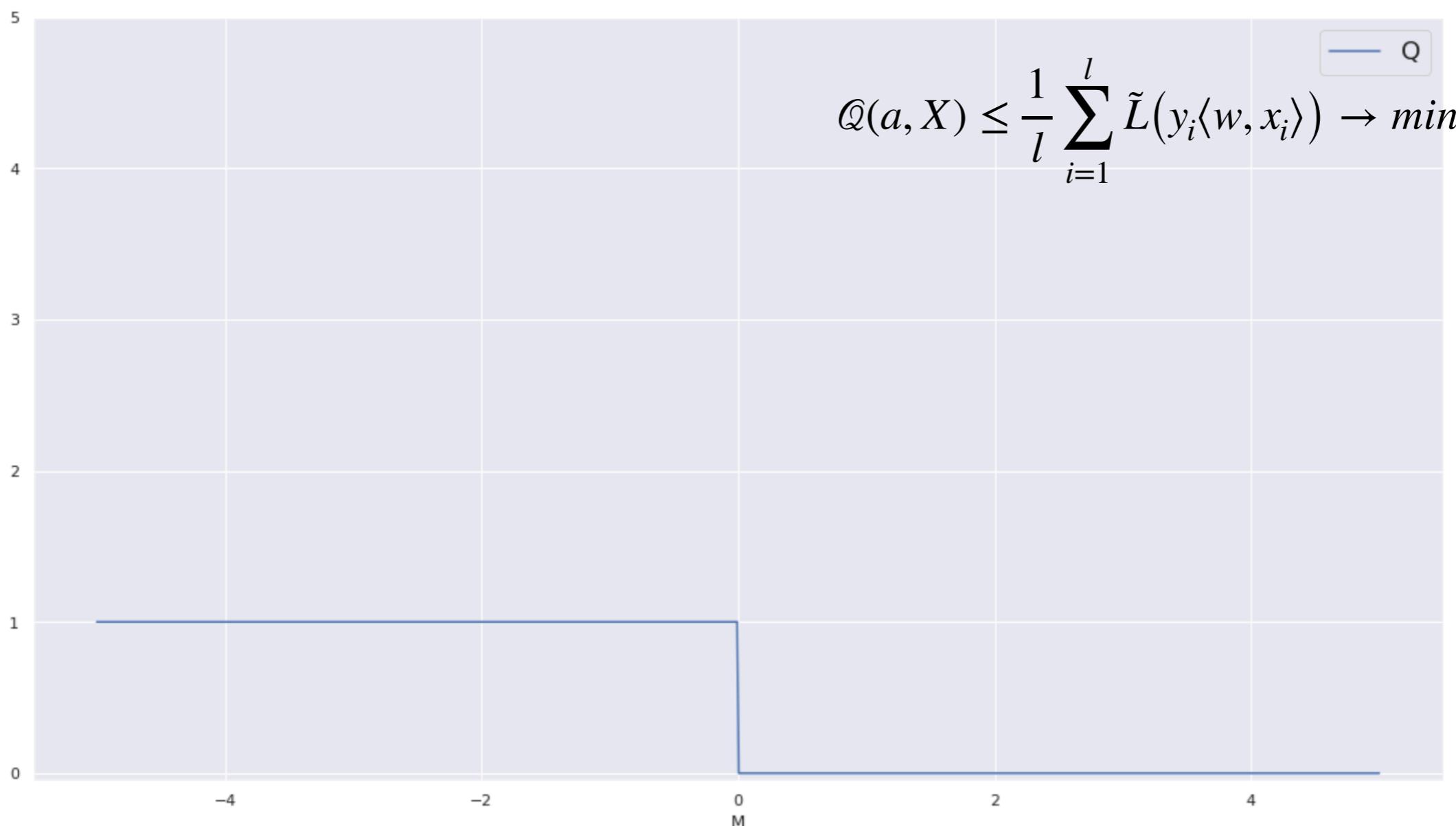
# Линейные модели в задаче классификации

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l \underbrace{[y_i \langle w, x_i \rangle < 0]}_{M_i} \rightarrow \min$$

$L(M) \leq \tilde{L}(M)$  – верхняя оценка функции потерь

$L(M) = [M < 0]$  – пороговая функции потерь

Если верхнюю оценку удастся приблизить к нулю, то и доля неправильных ответов тоже будет близка к нулю



$$\mathcal{Q}(a, X) \leq \frac{1}{l} \sum_{i=1}^l \tilde{L}(y_i \langle w, x_i \rangle) \rightarrow \min$$

# Линейные модели в задаче классификации

$$\tilde{L}(M) = (1 - M)^2$$

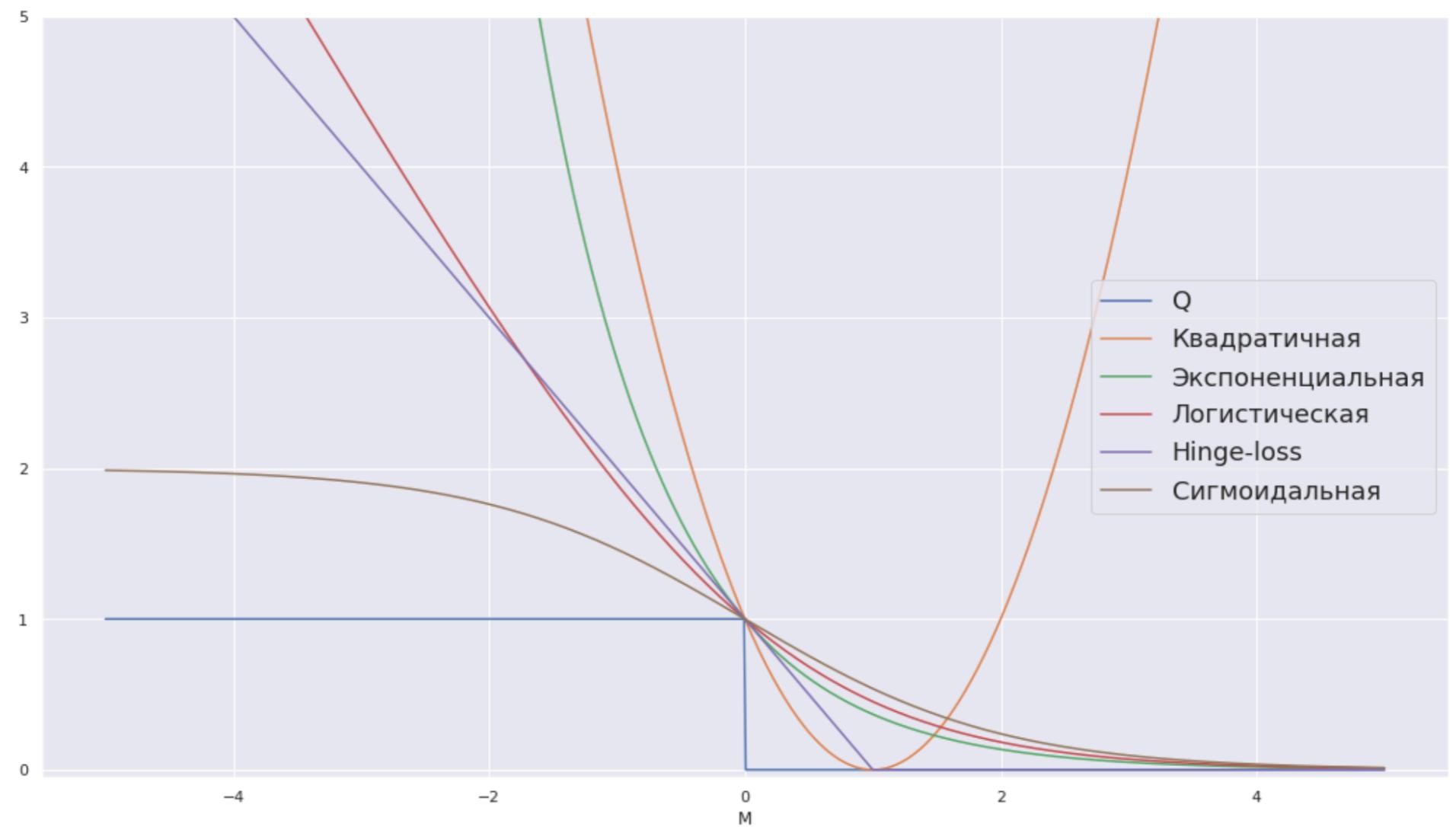
$L(M) = [M < 0]$  – пороговая функция потерь

$$\tilde{L}(M) = e^{-M}$$

$$\tilde{L}(M) = \log(1 + e^{-M})$$

$$\tilde{L}(M) = (1 - M)_+ = \max(0, 1 - M)$$

$$\tilde{L}(M) = \frac{2}{1 + e^{-M}}$$



# Линейные модели в задаче классификации

$$\tilde{L}(M) = (1 - M)^2$$

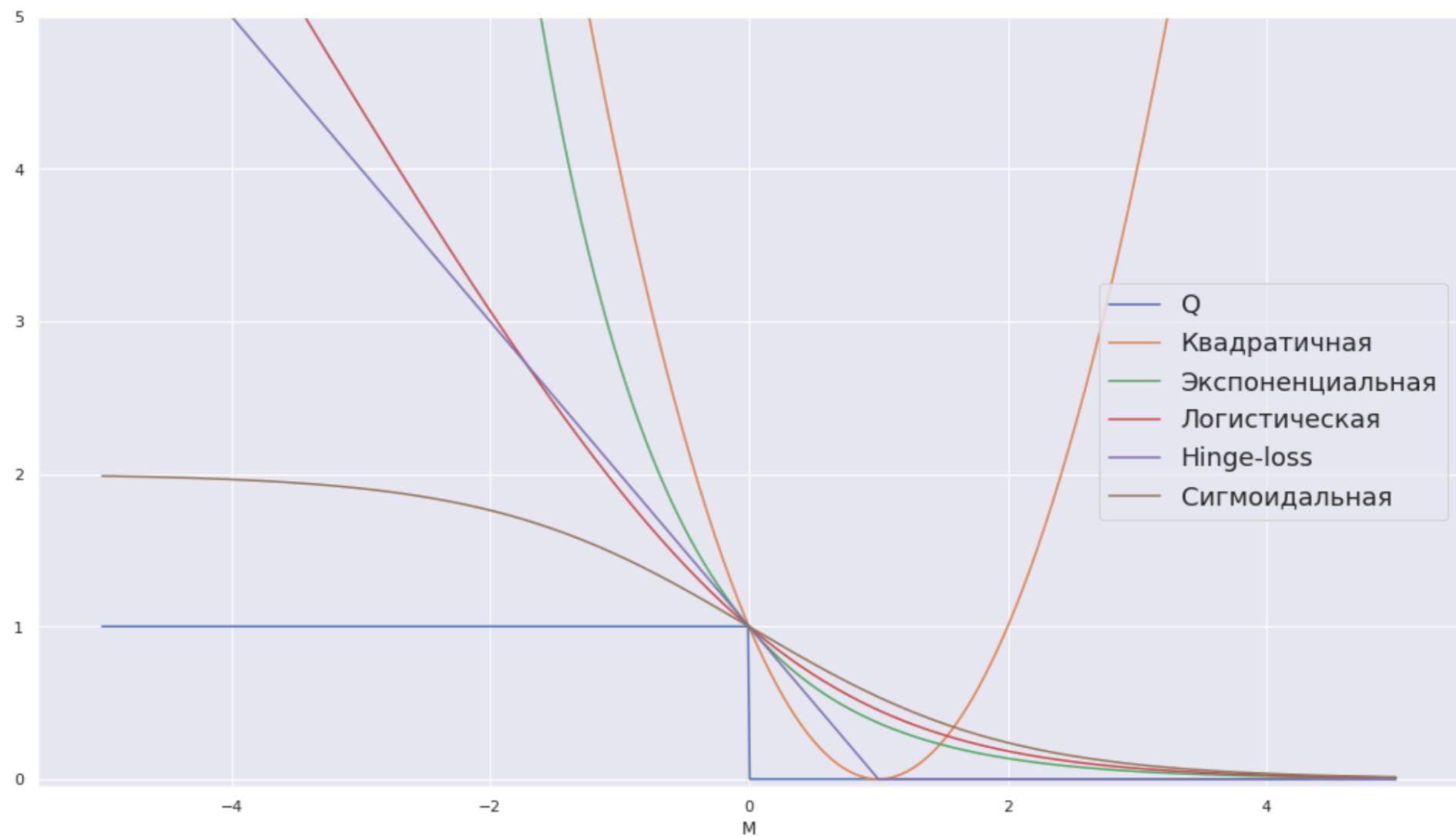
$L(M) = [M < 0]$  – пороговая функция потерь

$$\tilde{L}(M) = e^{-M}$$

$$\tilde{L}(M) = \log(1 + e^{-M})$$

$$\tilde{L}(M) = (1 - M)_+ = \max(0, 1 - M)$$

$$\tilde{L}(M) = \frac{2}{1 + e^{-M}}$$



# Линейные модели в задаче классификации

$$\tilde{L}(M) = \log(1 + e^{-M})$$

Минимизация эмпирического риска:

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l \log(1 + e^{-y_i \langle w, x_i \rangle}) \rightarrow \min$$

$$a(x) = \text{sign}(\vec{w}^T \cdot x)$$

# Линейные модели в задаче классификации

$$\tilde{L}(M) = \log(1 + e^{-M})$$

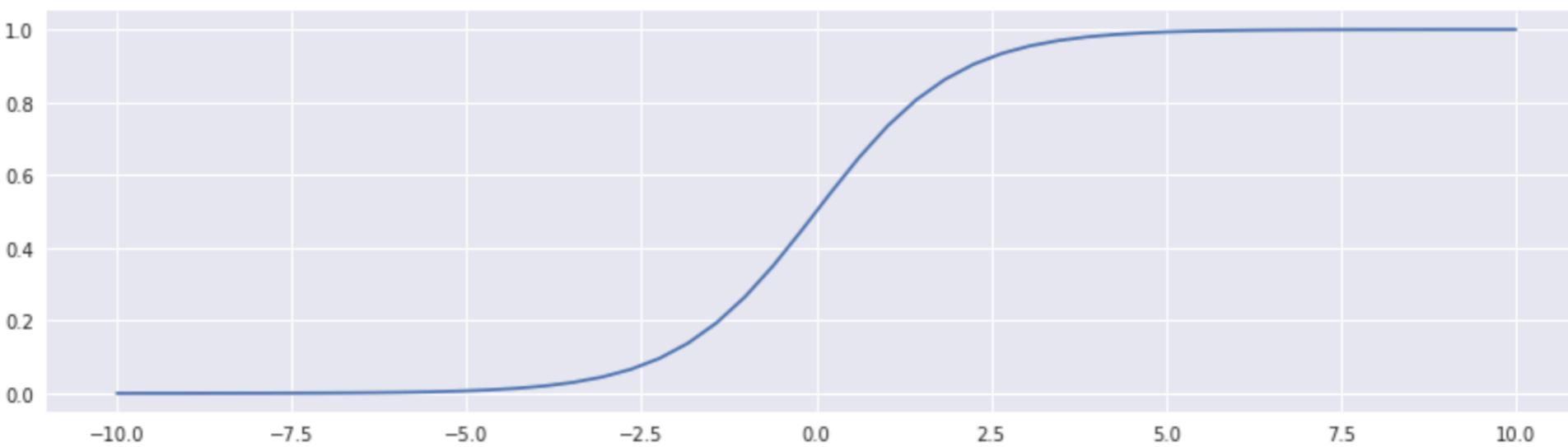
Минимизация эмпирического риска:

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l \log(1 + e^{-y_i \langle w, x_i \rangle}) \rightarrow \min$$

$$a(x) = \text{sign}(\vec{w}^T \cdot x)$$

Оценка апостериорных вероятностей принадлежности классам

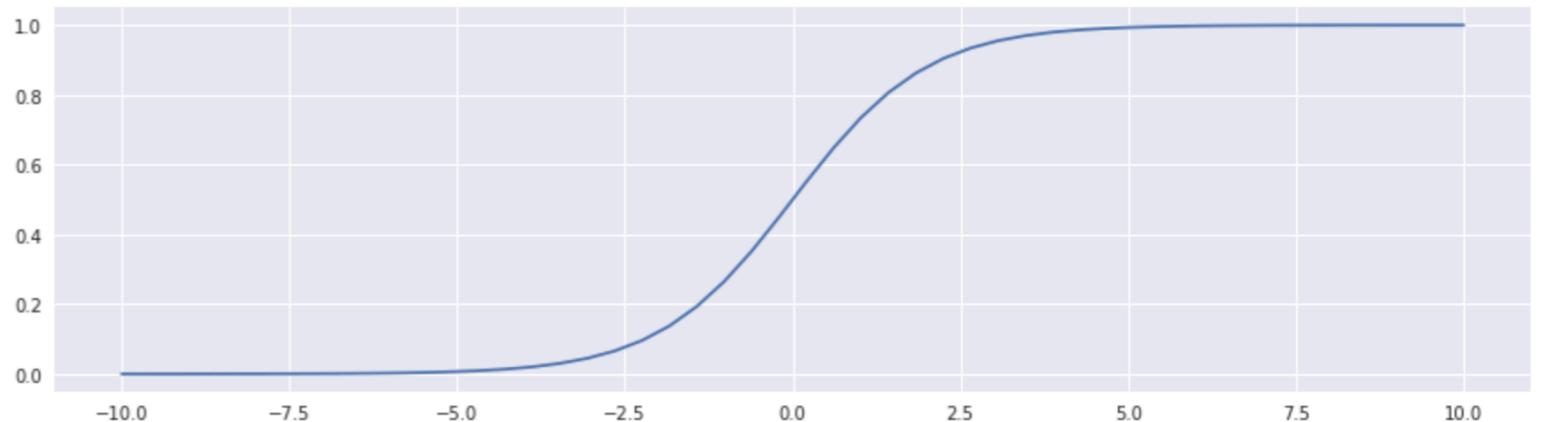
$$\vec{w}^T \cdot x \in R \quad f(x_i, w) = \sigma(z) = \frac{1}{1 + e^{-z}} \in [0; 1]$$



# Линейные модели в задаче классификации

$$y_i = f(x_i, w) + \varepsilon$$

$$f(x_i, w) = \sigma(z) = \frac{1}{1 + e^{-z}}$$



## Шансы

$$odds = \frac{p}{1 - p} \in [0; \infty] \quad \ln(odds) \in R$$

где  $p$  вероятность, что событие состоится (в нашем случае, что класс примет значение 1):

$$\ln(odds_+) = \ln\left(\frac{p}{1 - p}\right) = \ln(p) - \ln(1 - p)$$

$$\ln(odds_-) = \ln\left(\frac{1 - p}{p}\right) = \ln(1 - p) - \ln(p)$$

$$\ln(odds_+) = -\ln(odds_-) = w^T \cdot x \quad odds = e^{w^T x} \Rightarrow p = \frac{e^{w^T x}}{1 + e^{w^T x}}$$

# Линейные модели в задаче классификации

$$P(y_i = 1 | x_i, w) = \sigma(w^T x)$$

$$P(y_i = -1 | x_i, w) = \sigma(-w^T x)$$



$$P(y = y_i | x_i, w) = \sigma(y_i w^T x)$$

Правдоподобие (вероятность наблюдать вектор  $y$  при заданных значениях  $X$  и  $w$ )

Делаем предположение: объекты приходят независимо, из одного распределения

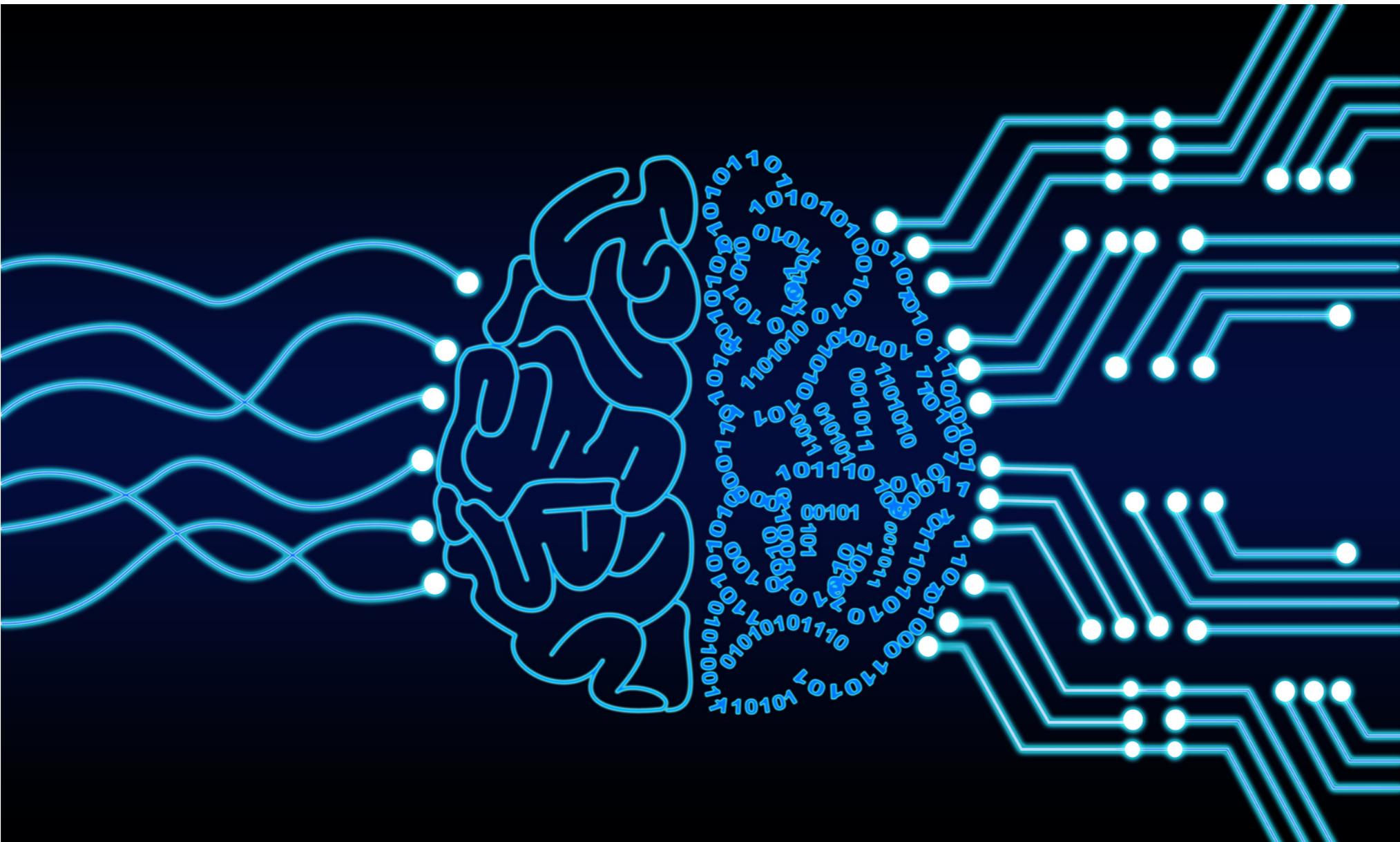
$$P(\vec{y} | X, w) = \prod_{i=1}^l P(y = y_i | x_i, w) \rightarrow \max$$

Так как логарифм монотонно возрастающая функция, то оценка  $w$  максимизирующая логарифм, будет максимизировать и само правдоподобие

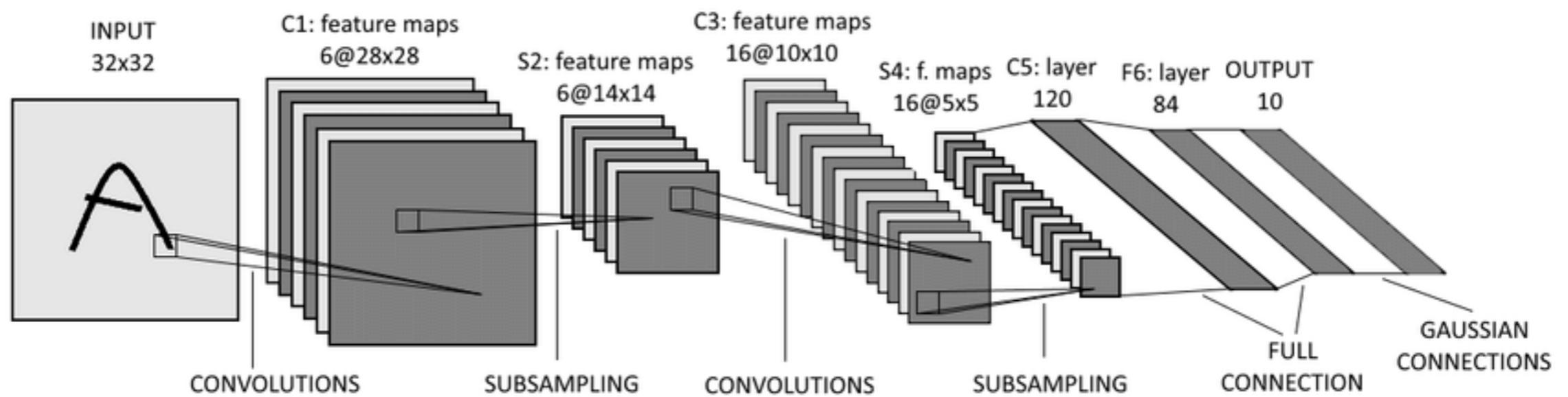
$$\log P(\vec{y} | X, w) = \sum_{i=1}^l \log \sigma(y_i w^T x) = \sum_{i=1}^l \log \frac{1}{1 + e^{-y_i \langle w, x_i \rangle}} = - \sum_{i=1}^l \log(1 + e^{-y_i \langle w, x_i \rangle})$$

$$\mathcal{Q}(a, X) = \frac{1}{l} \sum_{i=1}^l \log(1 + e^{-y_i \langle w, x_i \rangle}) \rightarrow \min$$

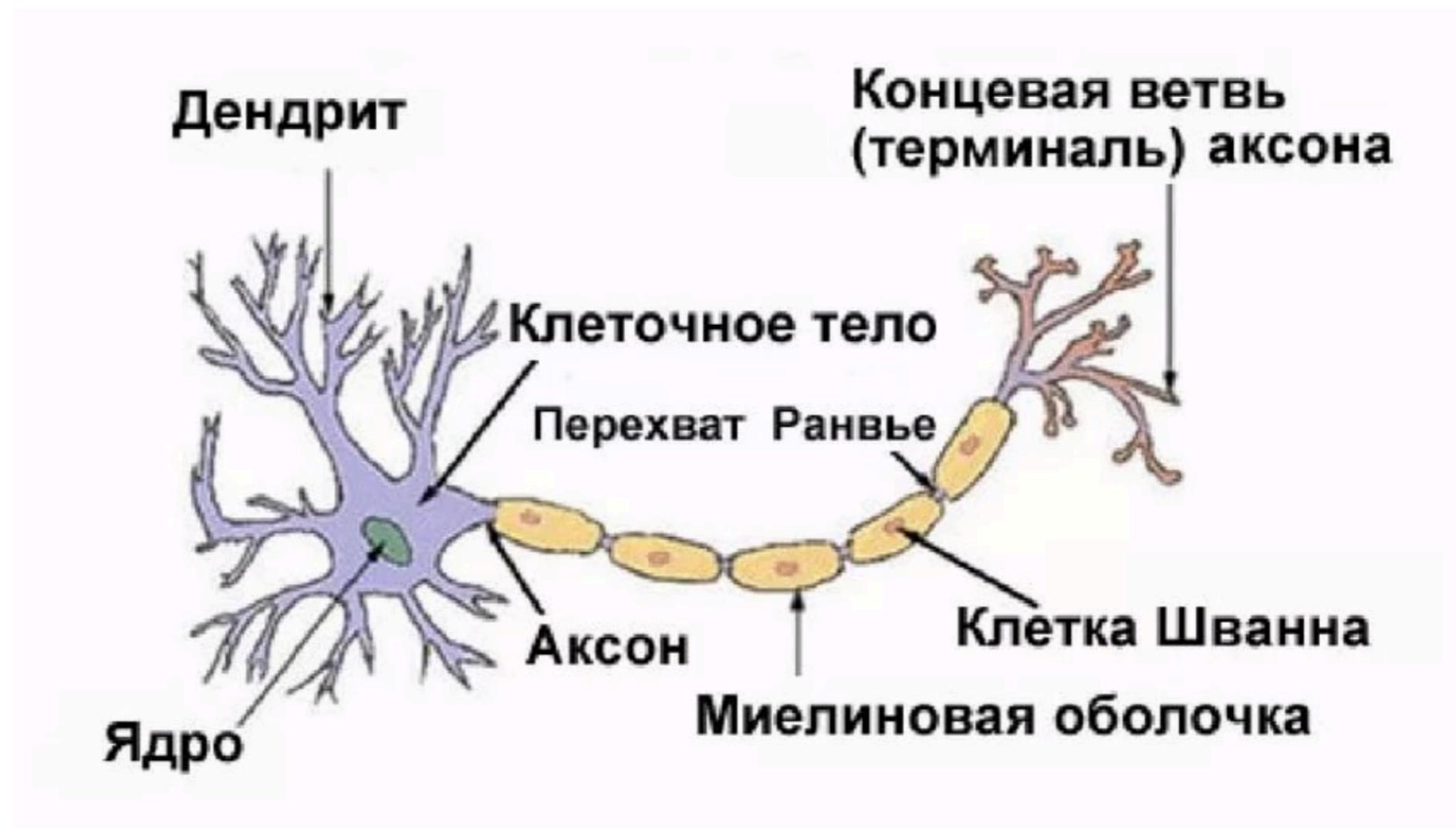
# Нейронные сети



# Сеть LeNet



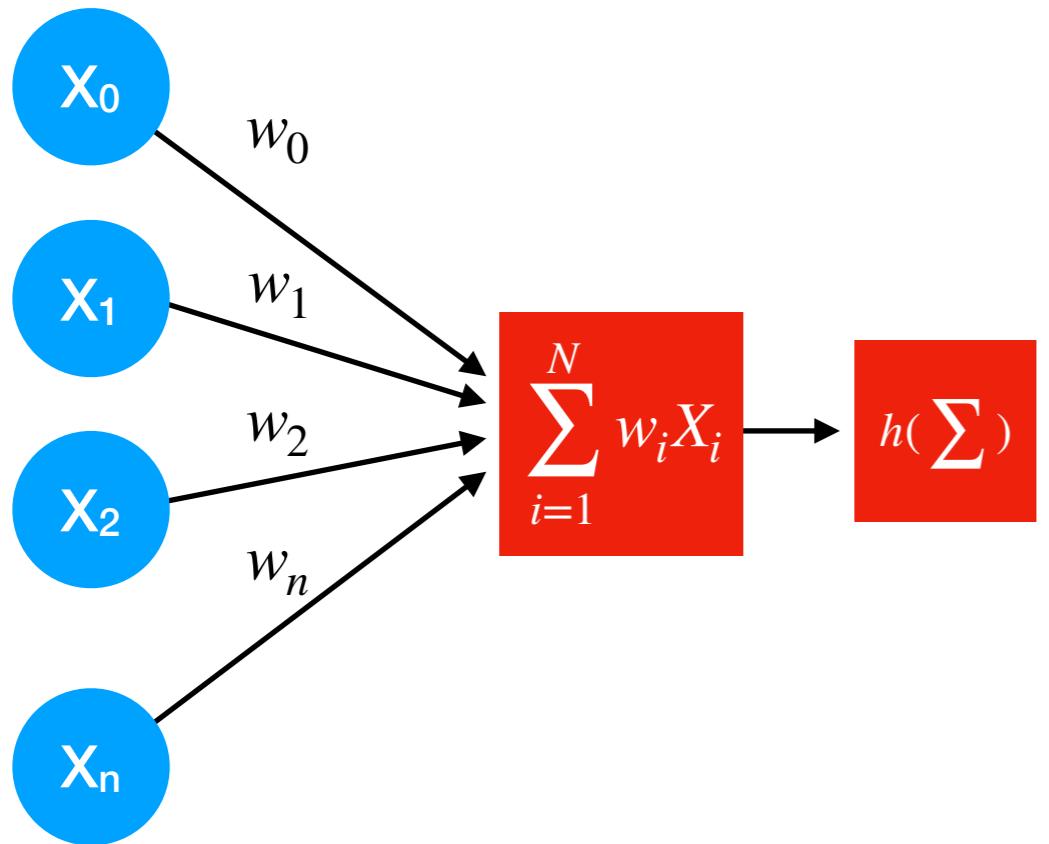
# Аналогия с биологическим нейроном



# Искусственный нейрон

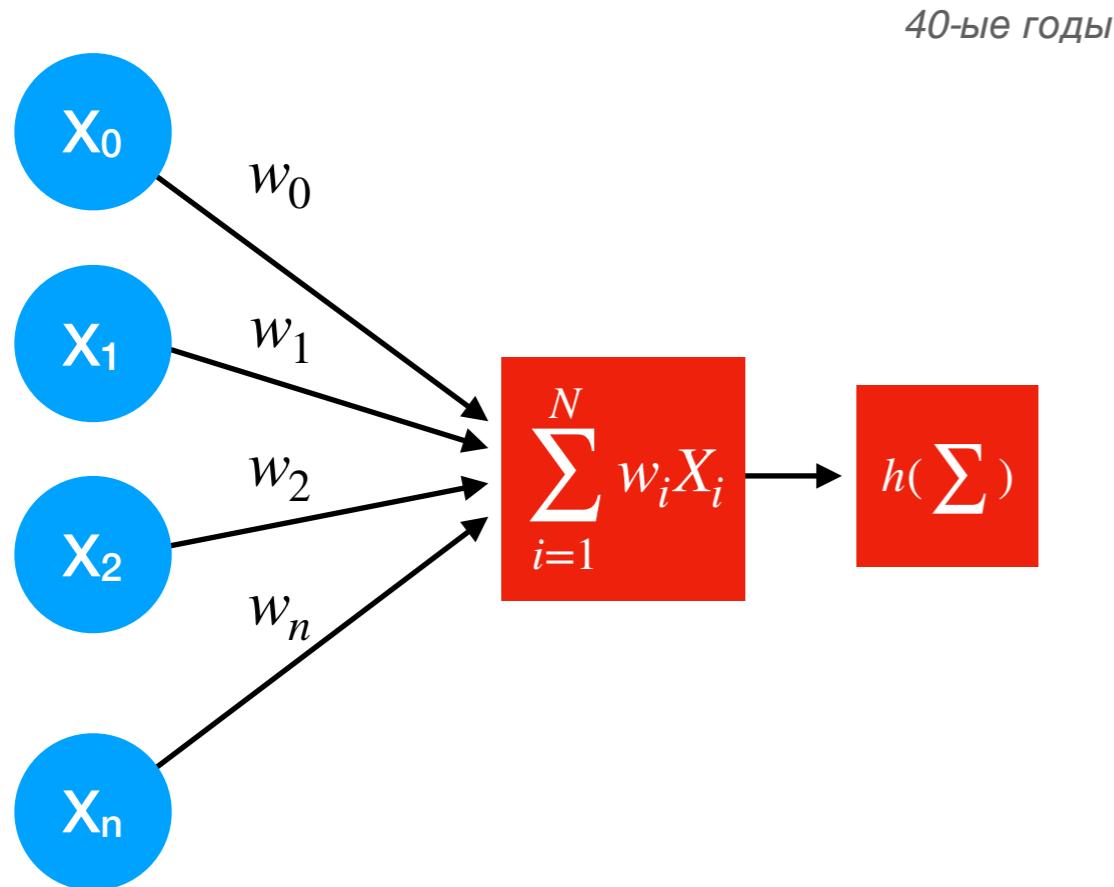
Математический нейрон Маккаллока – Питтса

40-ые годы

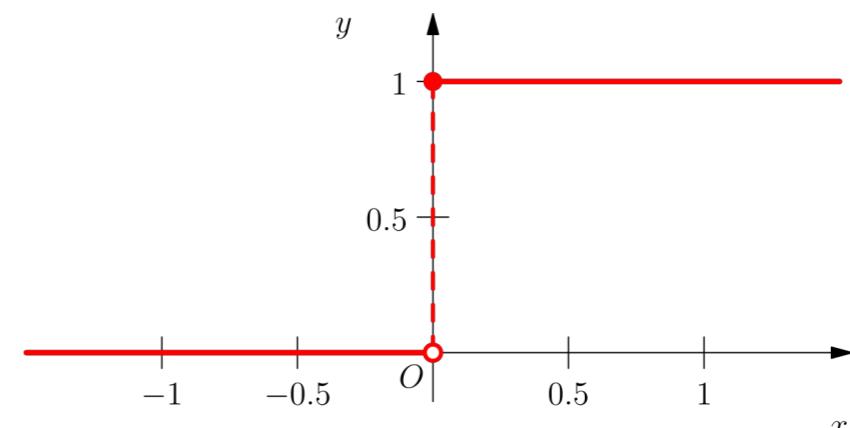


# Искусственный нейрон

Математический нейрон Маккаллока – Питтса

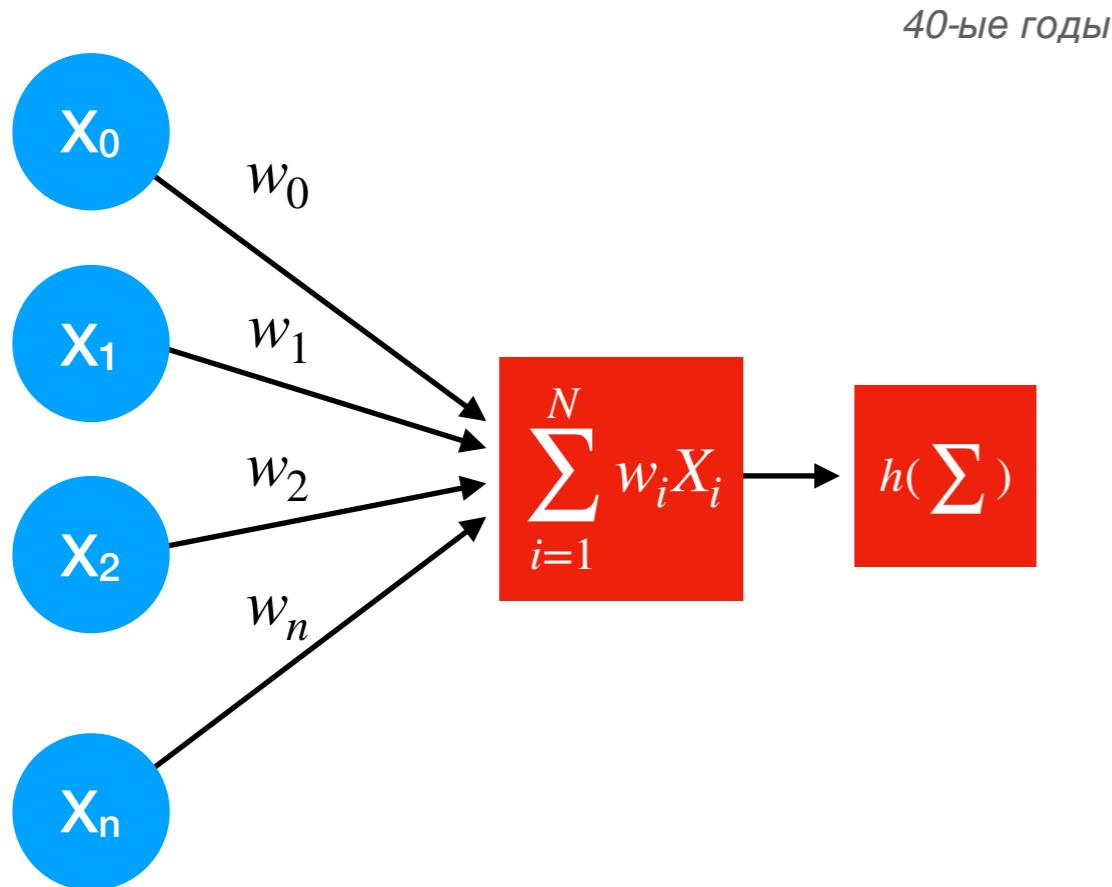


Функция Хевисайда

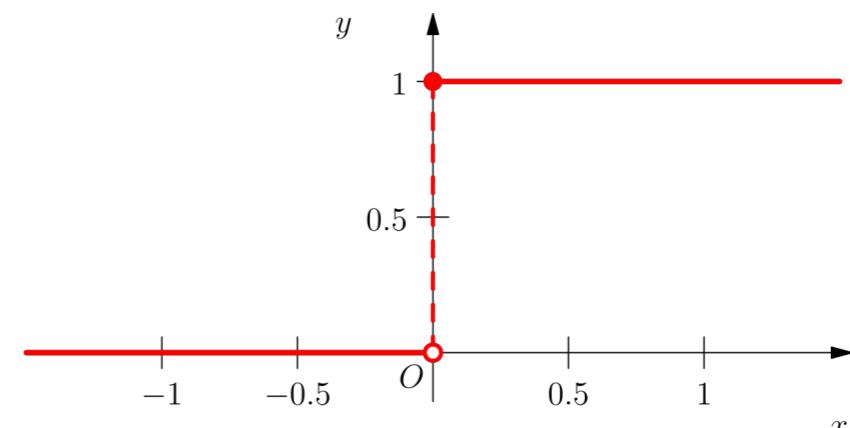


# Искусственный нейрон

Математический нейрон Маккаллока – Питтса



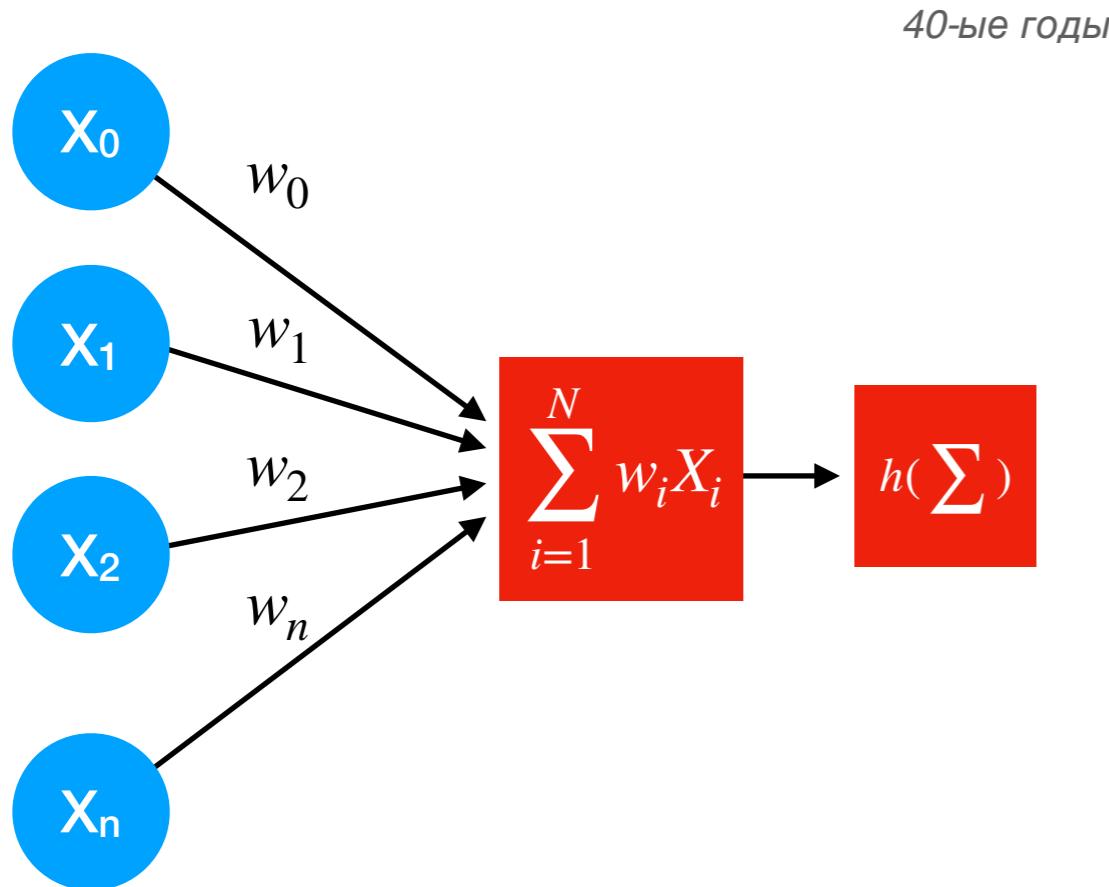
Функция Хевисайда



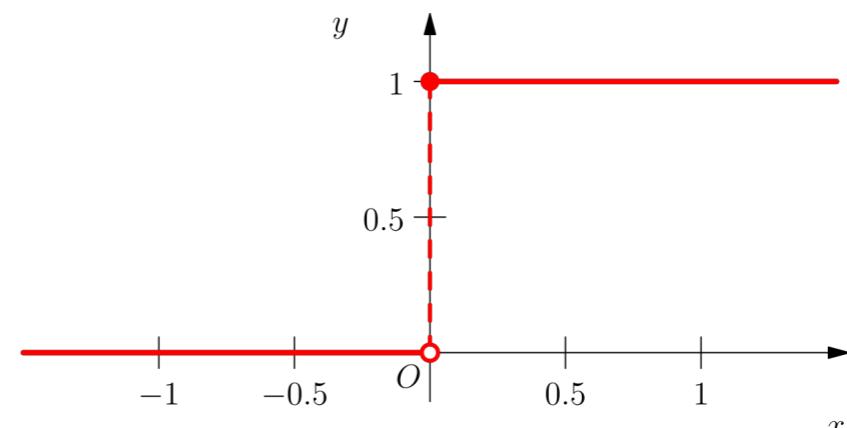
Обучение нейросети - Процесс подбора весов  $w_i$ , таким образом, чтобы сеть решала поставленную задачу.

# Искусственный нейрон

Математический нейрон Маккаллока – Питтса



Функция Хевисайда

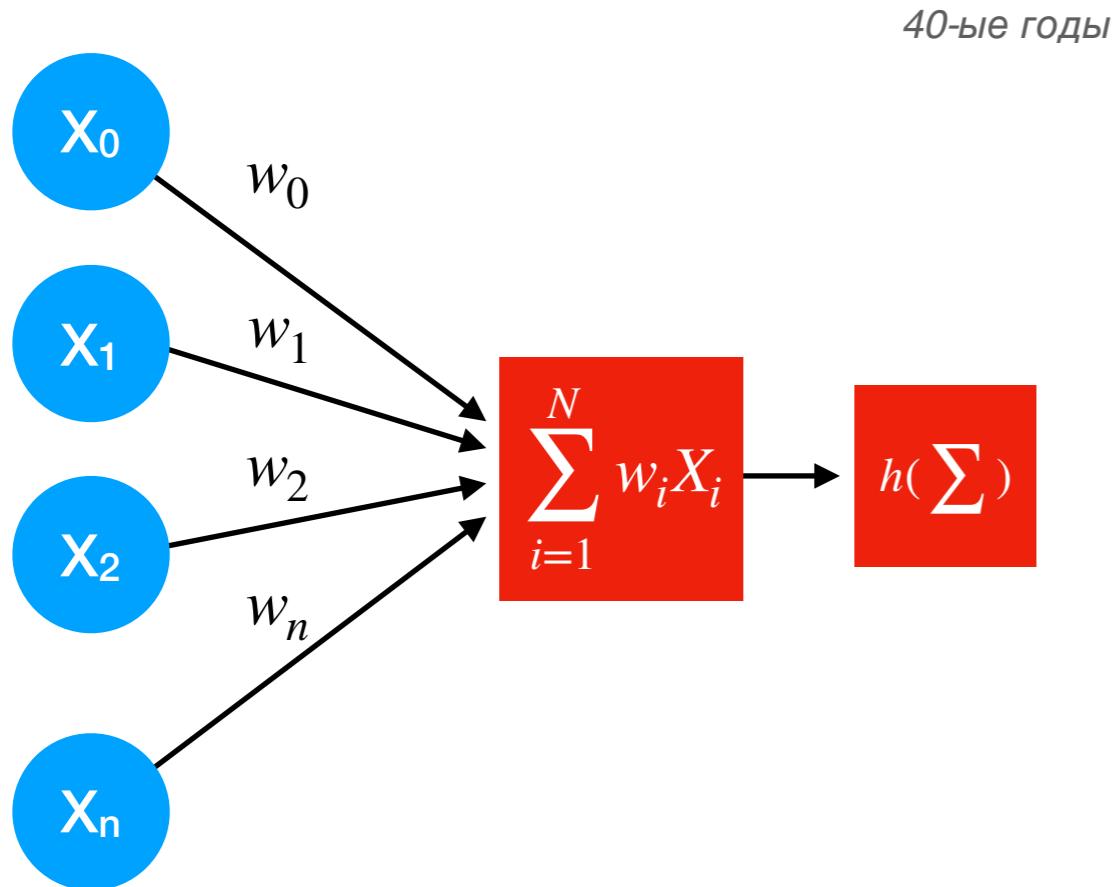


Обучение нейросети - Процесс подбора весов  $w_i$ , таким образом, чтобы сеть решала поставленную задачу.

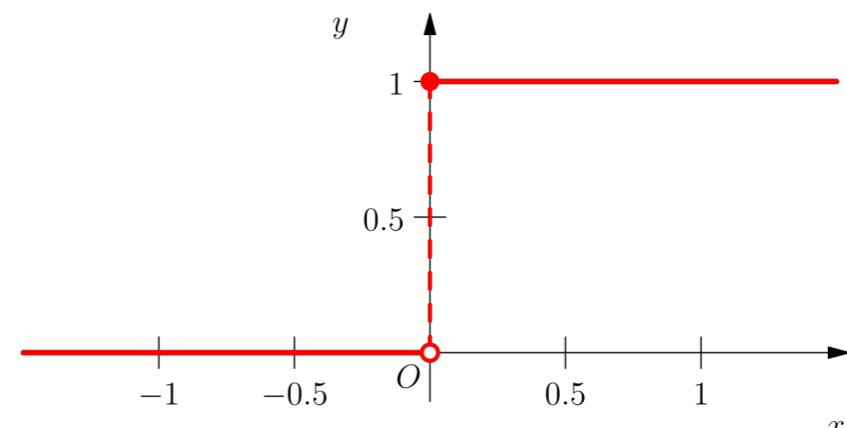
*В классической постановке, обучение не производилось, а веса задавались заранее*

# Искусственный нейрон

Математический нейрон Маккаллока – Питтса



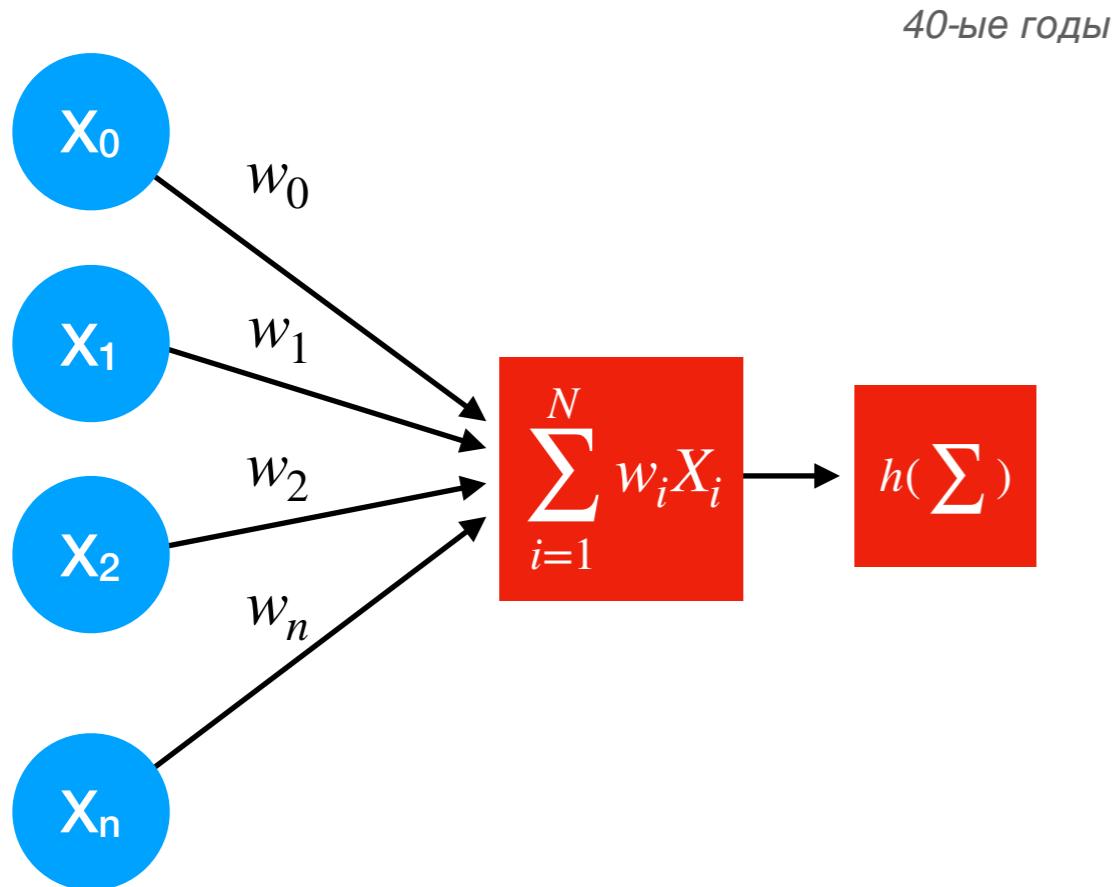
Функция Хевисайда



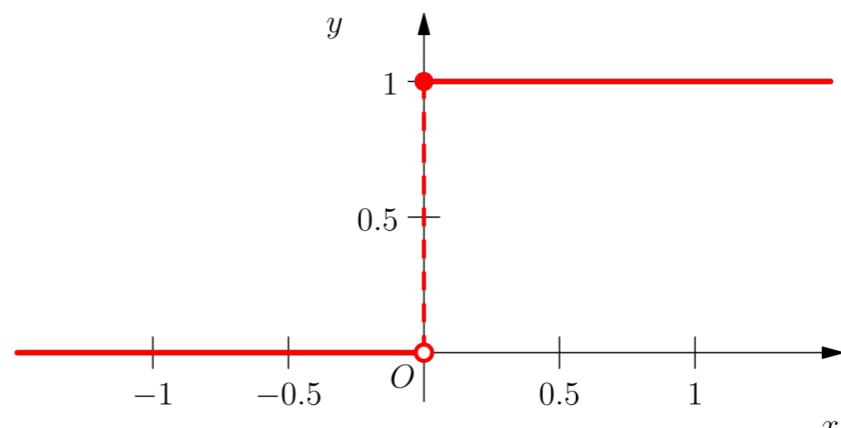
- Первое правило Хебба – *Если сигнал персептрана неверен и равен нулю, то необходимо увеличить веса тех входов, на которые была подана единица.*
- Второе правило Хебба – *Если сигнал персептрана неверен и равен единице, то необходимо уменьшить веса тех входов, на которые была подана единица.*

# Искусственный нейрон

## Математический нейрон Маккаллока – Питтса



## Функция Хевисайда

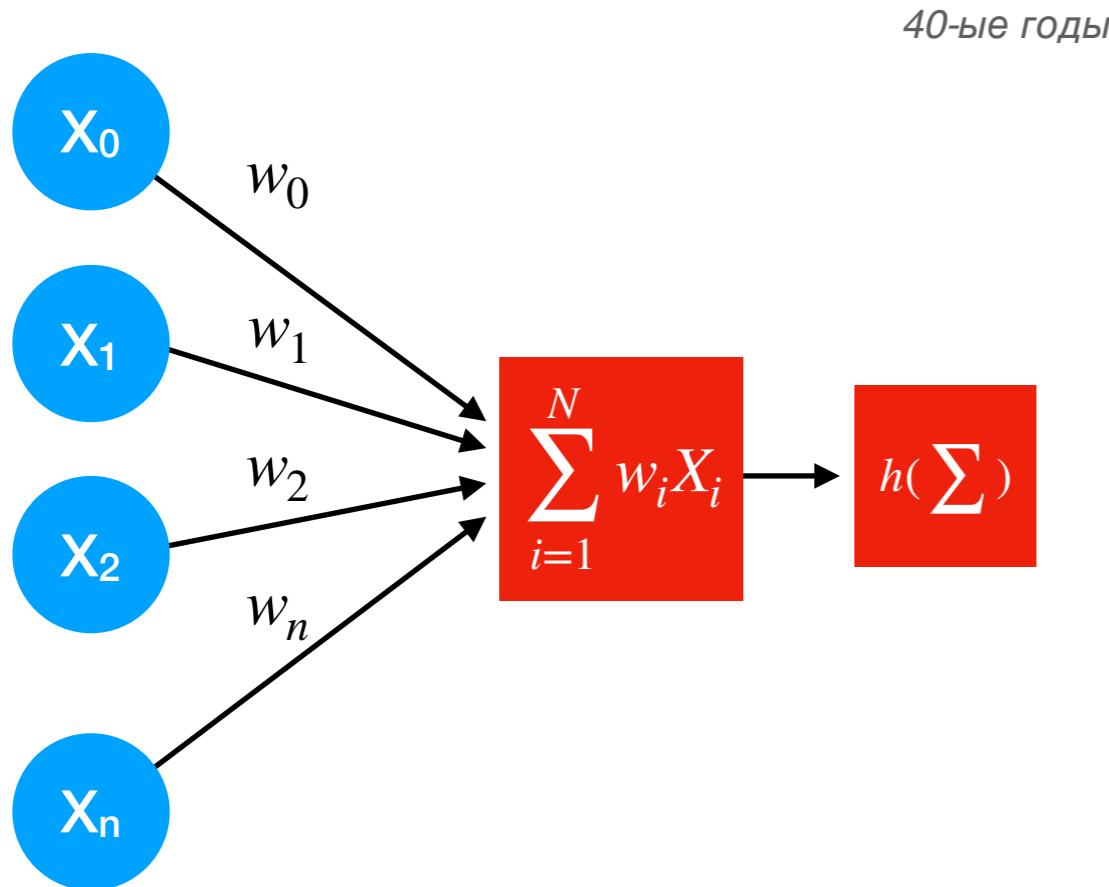


- Первое правило Хебба – *Если сигнал персептрана неверен и равен нулю, то необходимо увеличить веса тех входов, на которые была подана единица.*
- Второе правило Хебба – *Если сигнал персептрана неверен и равен единице, то необходимо уменьшить веса тех входов, на которые была подана единица.*

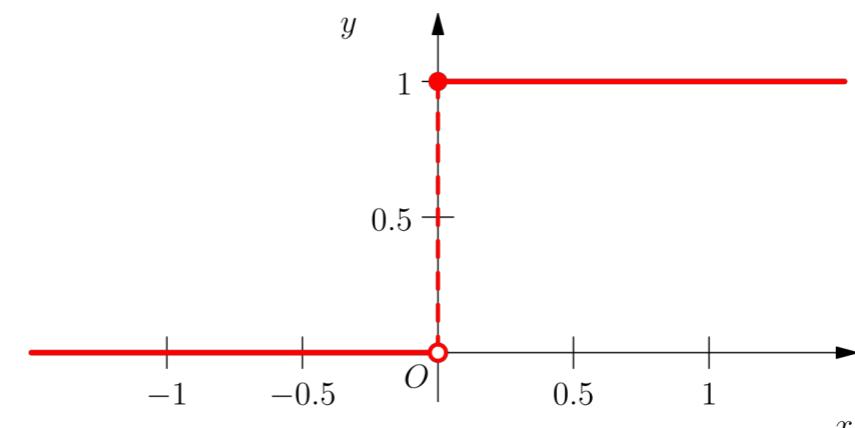
**Биологическая предпосылка:**  
Если нейрон срабатывает, то синоптическая связь укрепляется

# Искусственный нейрон

## Математический нейрон Маккаллока – Питтса

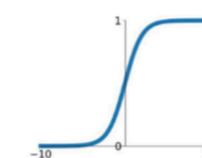


## Функция Хевисайда

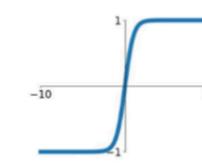


## Другие функции активации:

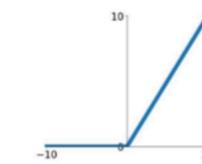
**Sigmoid**  
 $\sigma(x) = \frac{1}{1+e^{-x}}$



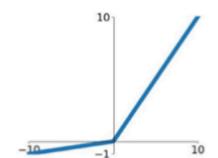
**tanh**  
 $\tanh(x)$



**ReLU**  
 $\max(0, x)$

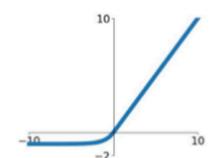


**Leaky ReLU**  
 $\max(0.1x, x)$



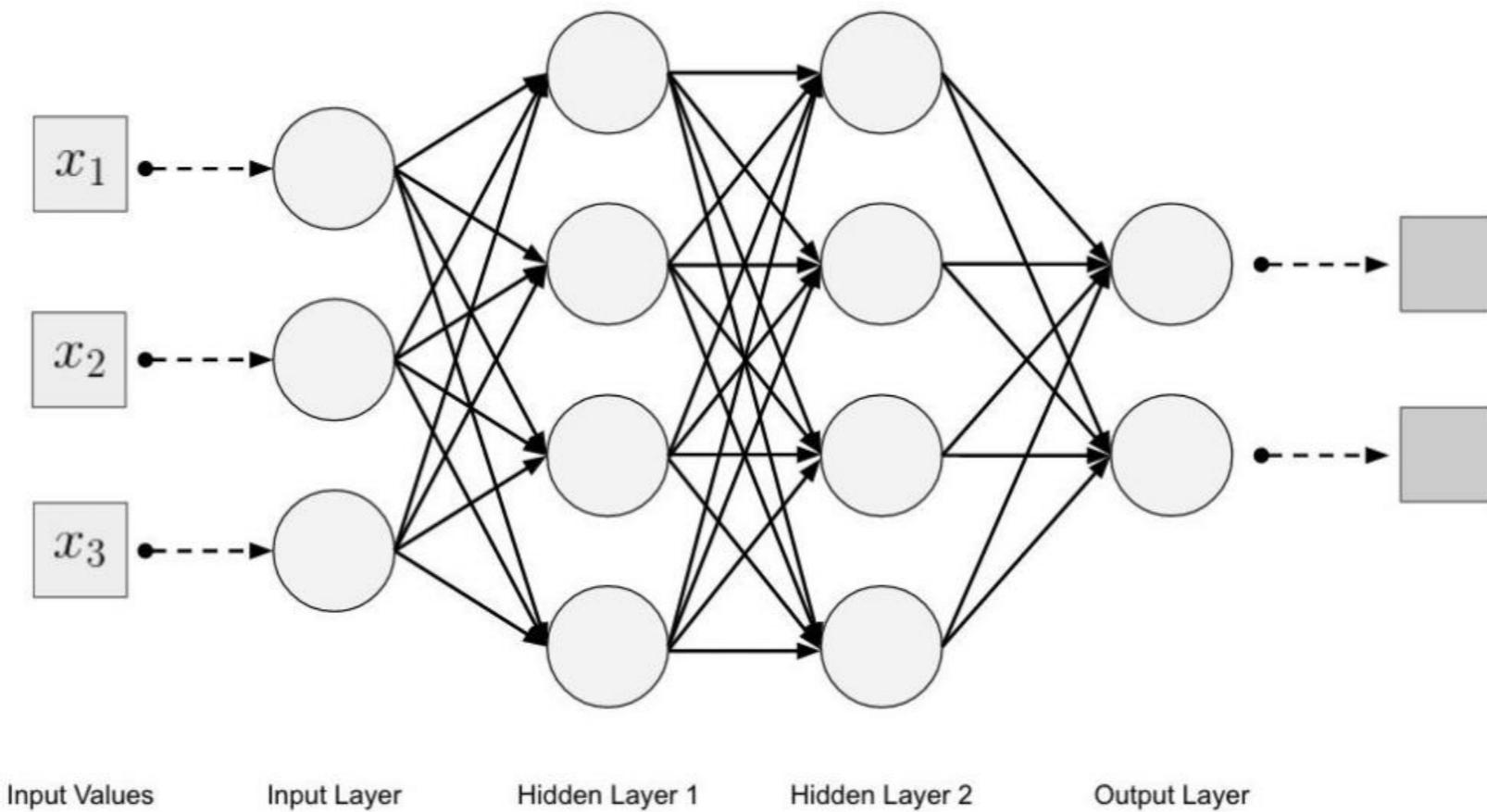
**Maxout**  
 $\max(w_1^T x + b_1, w_2^T x + b_2)$

**ELU**  
 $\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

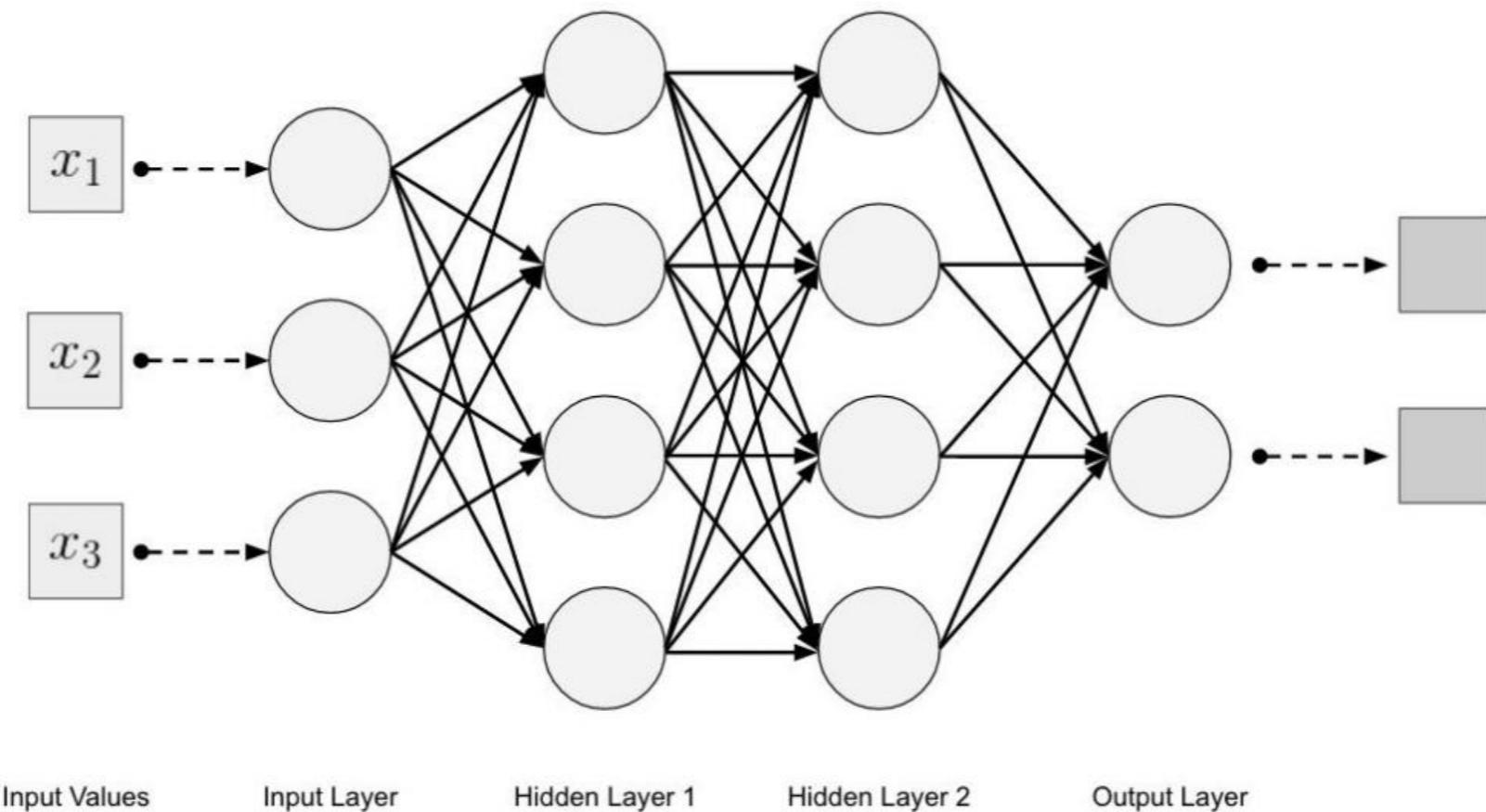


- Первое правило Хебба – *Если сигнал персептрона неверен и равен нулю, то необходимо увеличить веса тех входов, на которые была подана единица.*
- Второе правило Хебба – *Если сигнал персептрона неверен и равен единице, то необходимо уменьшить веса тех входов, на которые была подана единица.*

# Нейронные сети



# Нейронные сети



## Утверждения:

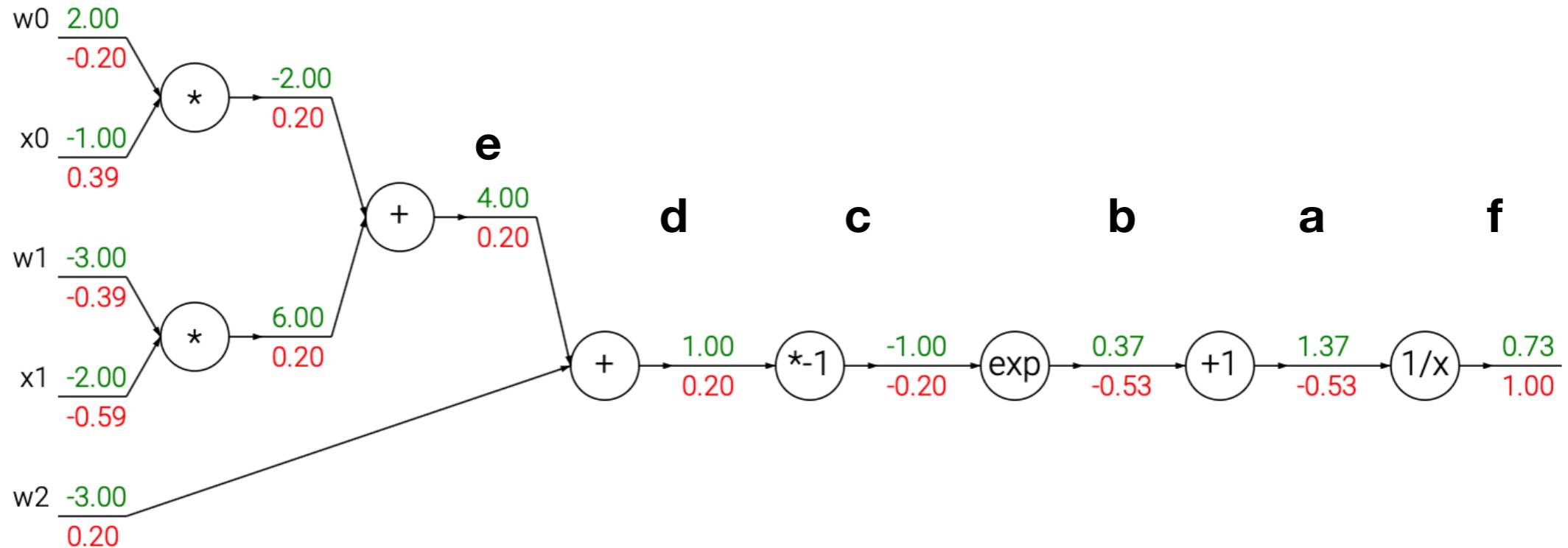
Любая булева функция представима в виде нейронной сети с одним скрытым слоем с нелинейной функцией активации нейрона (но может потребоваться экспоненциально много нейронов в скрытом слое).

Любая непрерывная и ограниченная функция может быть сколь угодно точно аппроксимирована нейронной сетью с одним скрытым слоем с нелинейной функцией активации нейрона.

Любая функция может быть сколь угодно точно аппроксимирована нейронной сетью с двумя скрытыми слоями с нелинейной функцией активации нейрона.

# Обучение

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



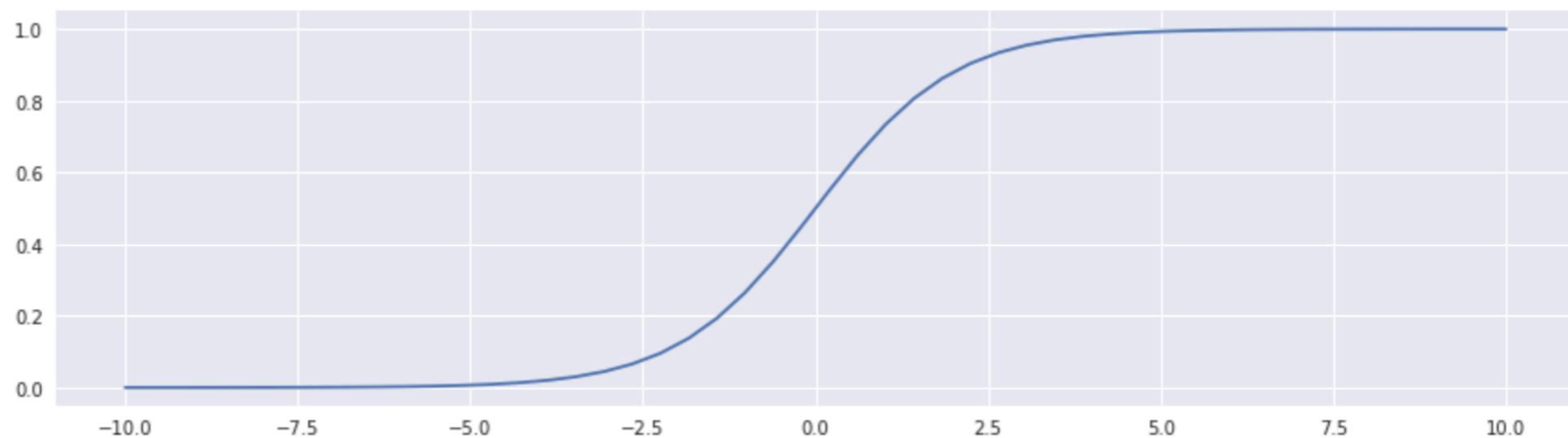
**f**       $f(x) = \frac{1}{x}$        $\rightarrow$        $\frac{df}{dx} = -1/x^2$

**a**       $f_c(x) = c + x$        $\rightarrow$        $\frac{df}{dx} = 1$

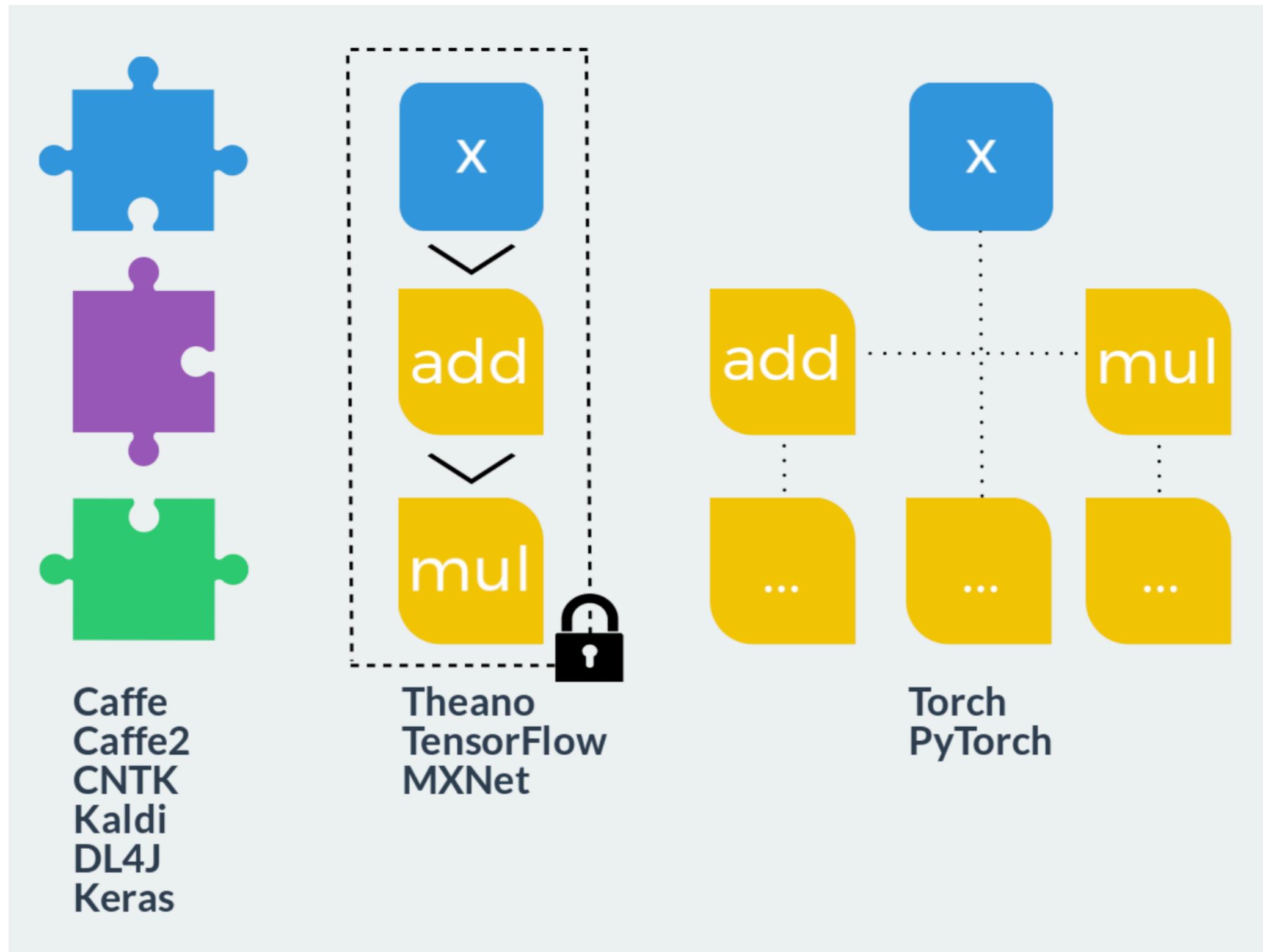
**b**       $f(x) = e^x$        $\rightarrow$        $\frac{df}{dx} = e^x$

**c**       $f_a(x) = ax$        $\rightarrow$        $\frac{df}{dx} = a$

# Проблема затухающих градиентов

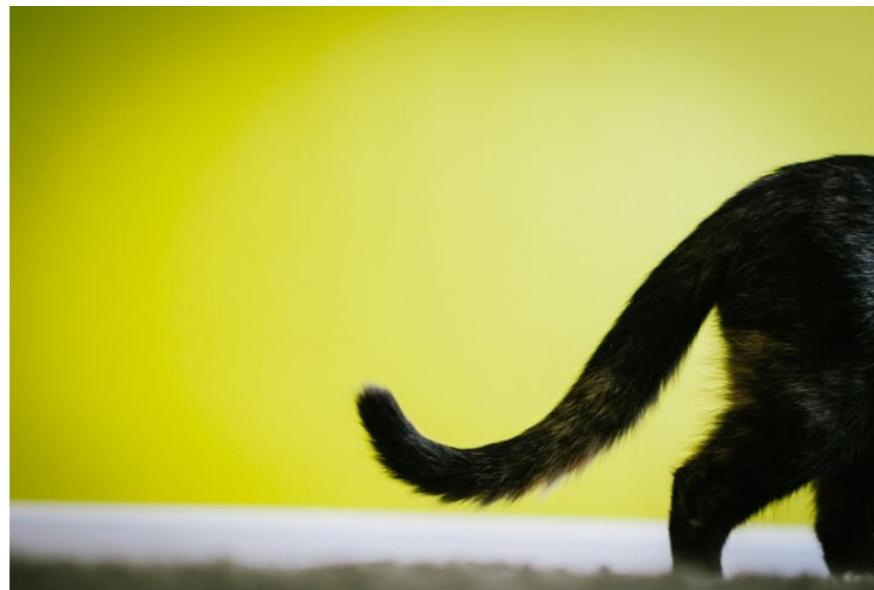


# Сравнение библиотек



# Реализация в python

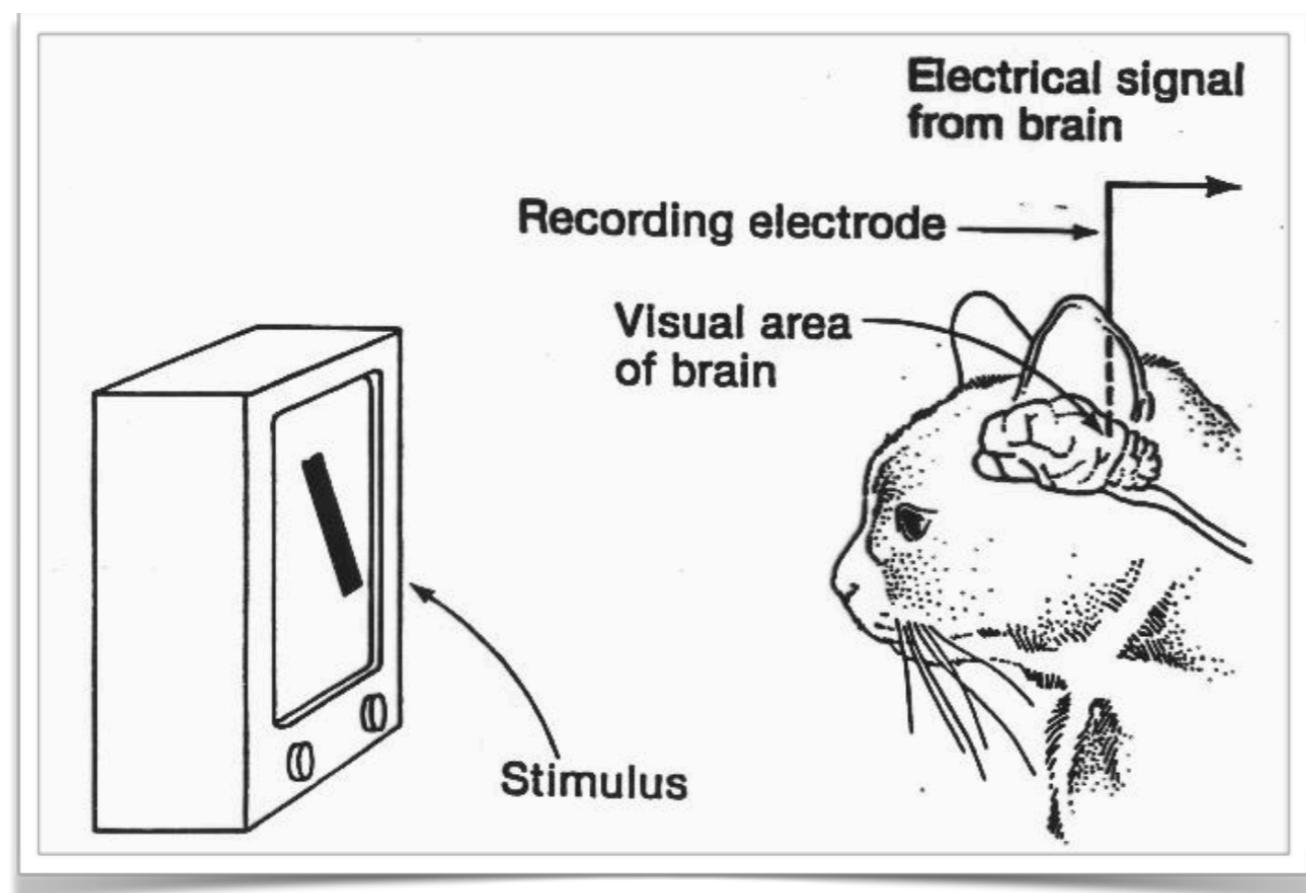
# Распознавание образов



# Биологическая предпосылка

The Nobel Prize in Physiology or Medicine 1981

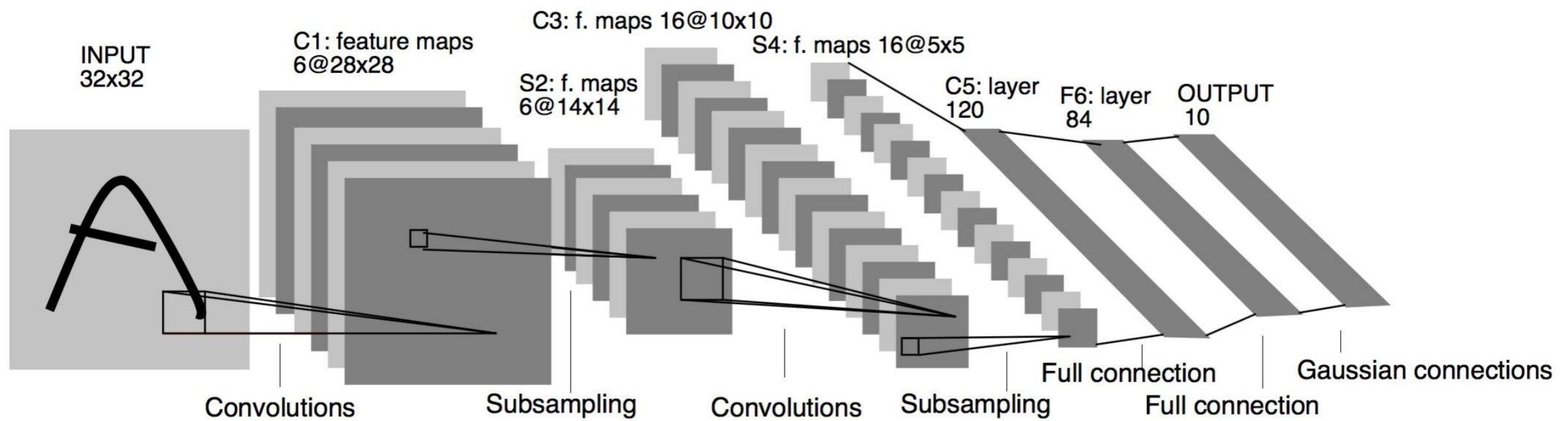
специализированные нейроны, которые реагируют только на определенную сенсорную информацию.



- соседние нейроны обрабатывают сигналы с соседних областей сетчатки;
- нейроны образуют иерархическую структуру (изображение ниже), где каждый следующий уровень выделяет все более и более высокоуровневые признаки;
- нейроны организованы в так называемые колонки – вычислительные блоки, которые трансформируют и передают информацию от уровня к уровню.

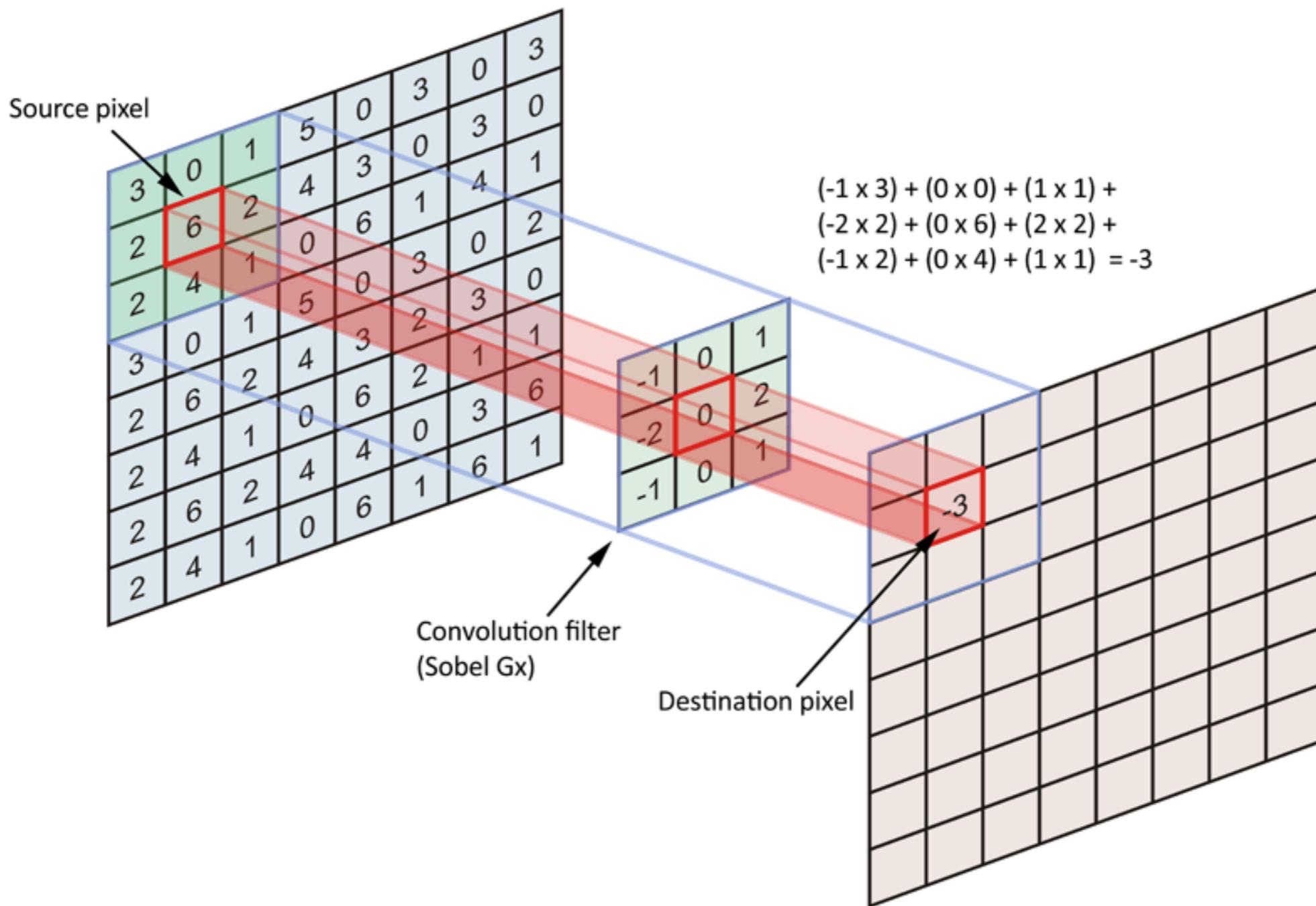
<https://www.youtube.com/watch?v=IOHayh06LJ4>

# Первая сверточная нейронная сеть

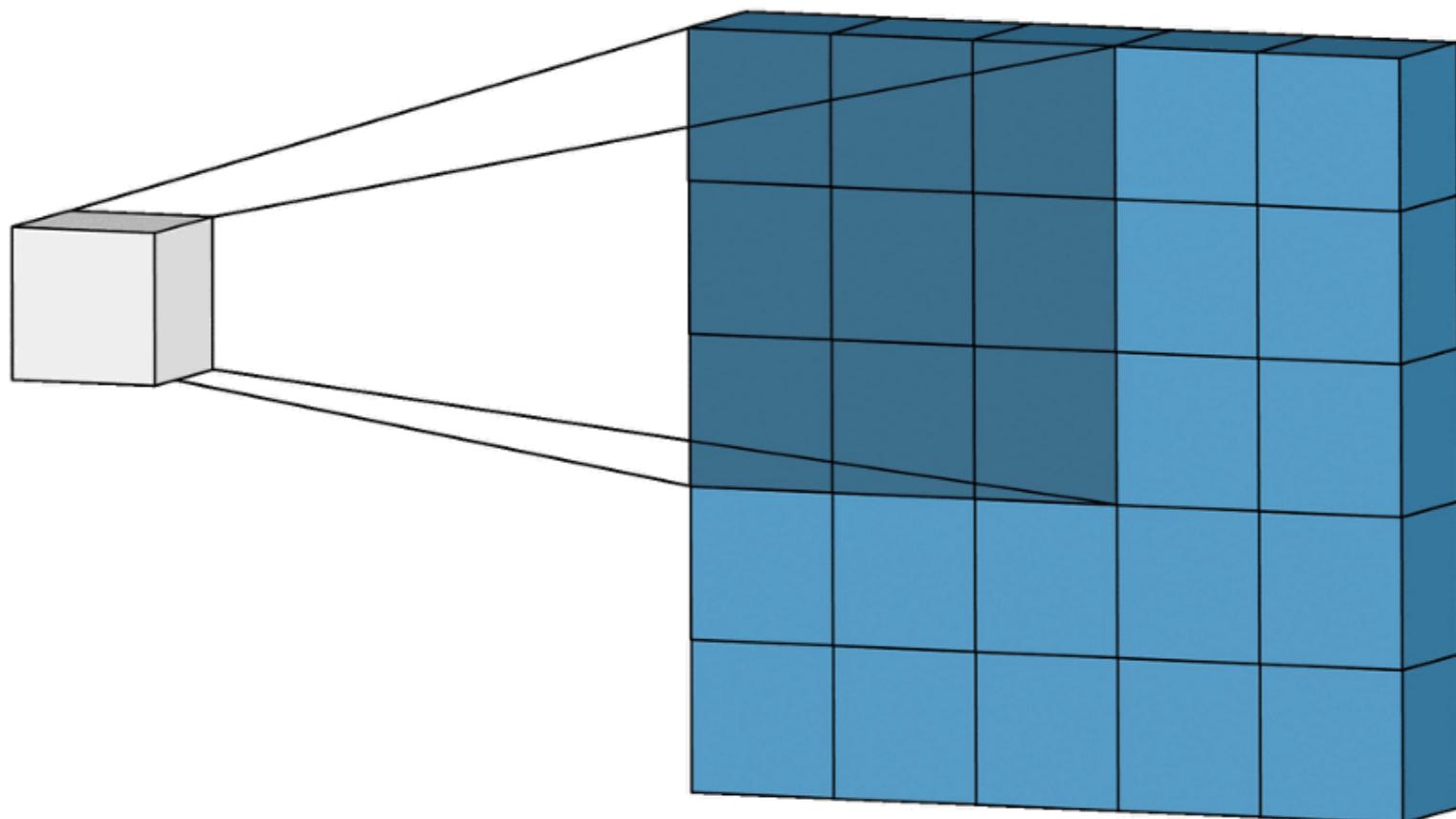


- Свертки  $5 \times 5$  со сдвигом 1 и пулинг  $2 \times 2$  со сдвигом 2
- 60 тысяч параметров
- Практическое применение: с помощью этой сети на почте США распознавали рукописные индексы

# Операция Свертки



# Операция Свертки



<https://proglib.io/wp-content/uploads/2018/06/1.gif>

# Операция Свертки

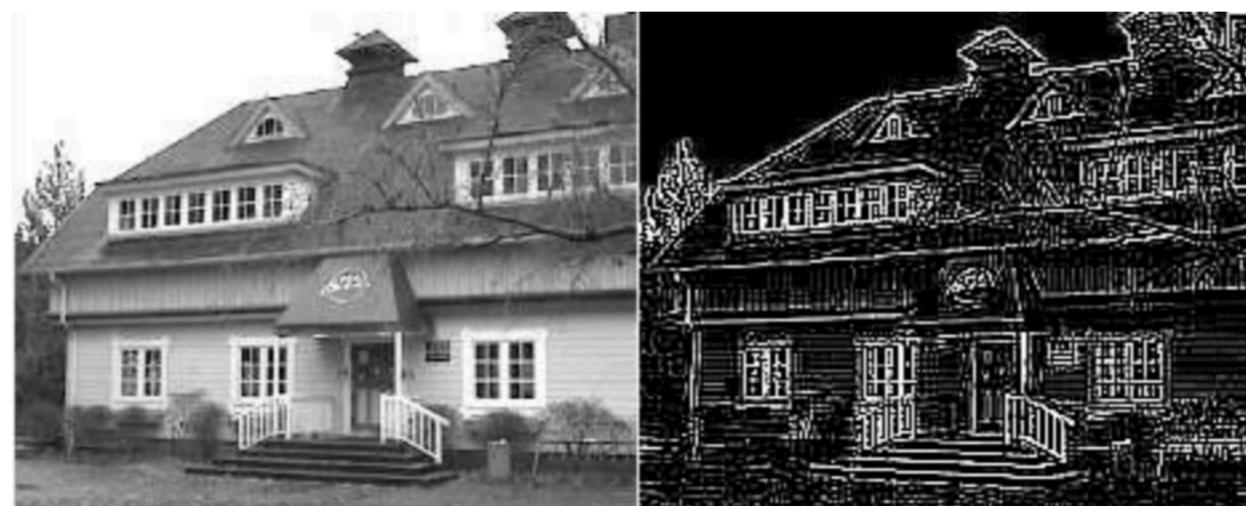
Blur

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



Edge detection

-1	-1	-1
-1	8	-1
-1	-1	-1



<http://aishack.in/tutorials/image-convolution-examples/>

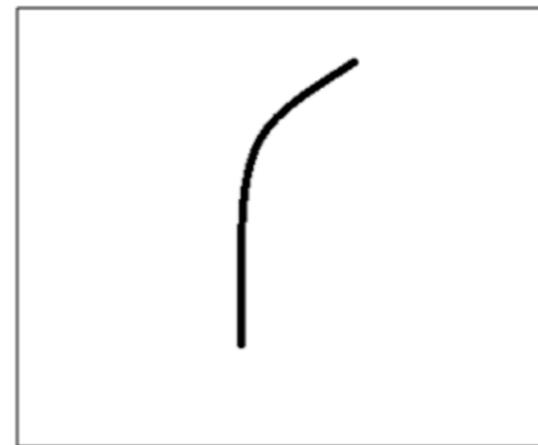
<https://habr.com/post/142818/>

# Операция Свертки

## Фильтры

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

# Операция Свертки



Original image



Visualization of the filter on the image



Visualization of the receptive field

0	0	0	0	0	0	30	
0	0	0	0	50	50	50	
0	0	0	20	50	0	0	
0	0	0	50	50	0	0	
0	0	0	50	50	0	0	
0	0	0	50	50	0	0	
0	0	0	50	50	0	0	

Pixel representation of the receptive field

\*

0	0	0	0	0	0	30	0
0	0	0	0	0	30	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation =  $(50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600$  (A large number!)

# Операция Свертки



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

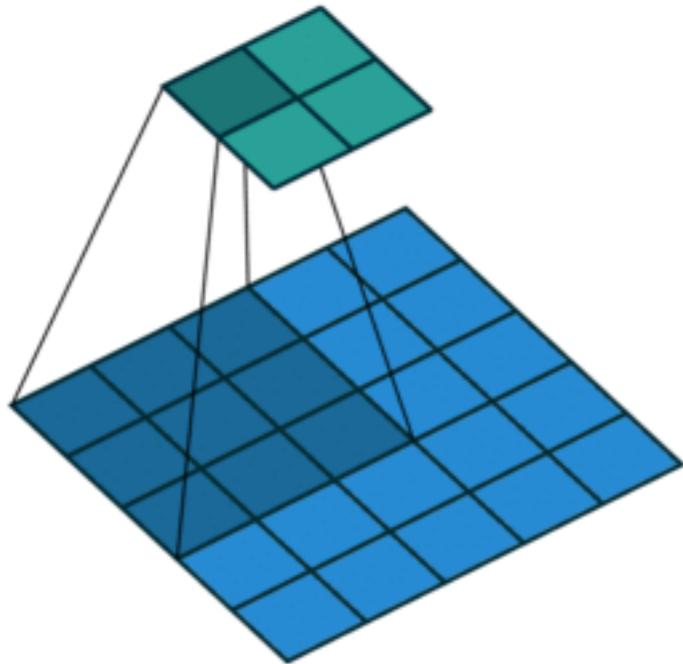
\*

0	0	0	0	0	0	30	0
0	0	0	0	0	30	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	0	0	0	0	0

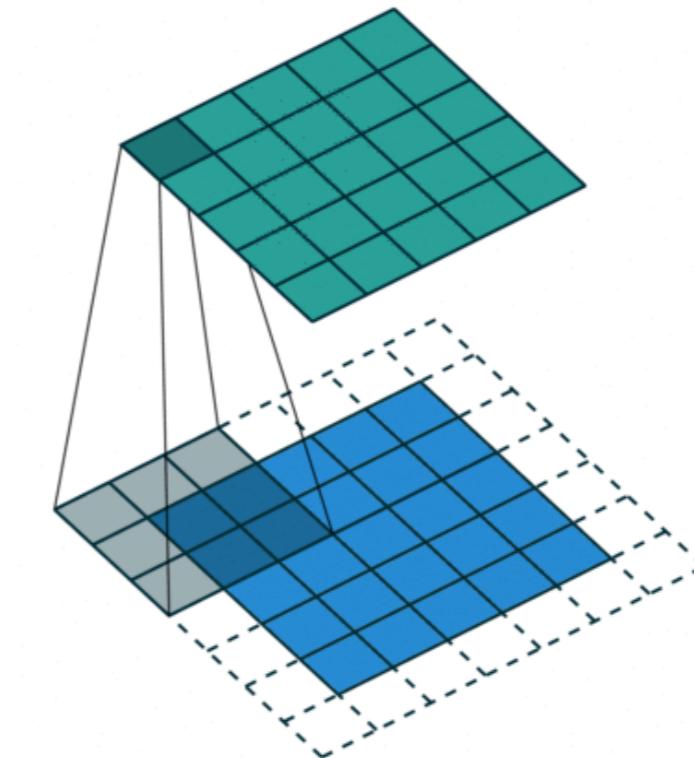
Pixel representation of filter

Multiplication and Summation = 0

# Операция Сдвига (stride) и дополнения (padding)

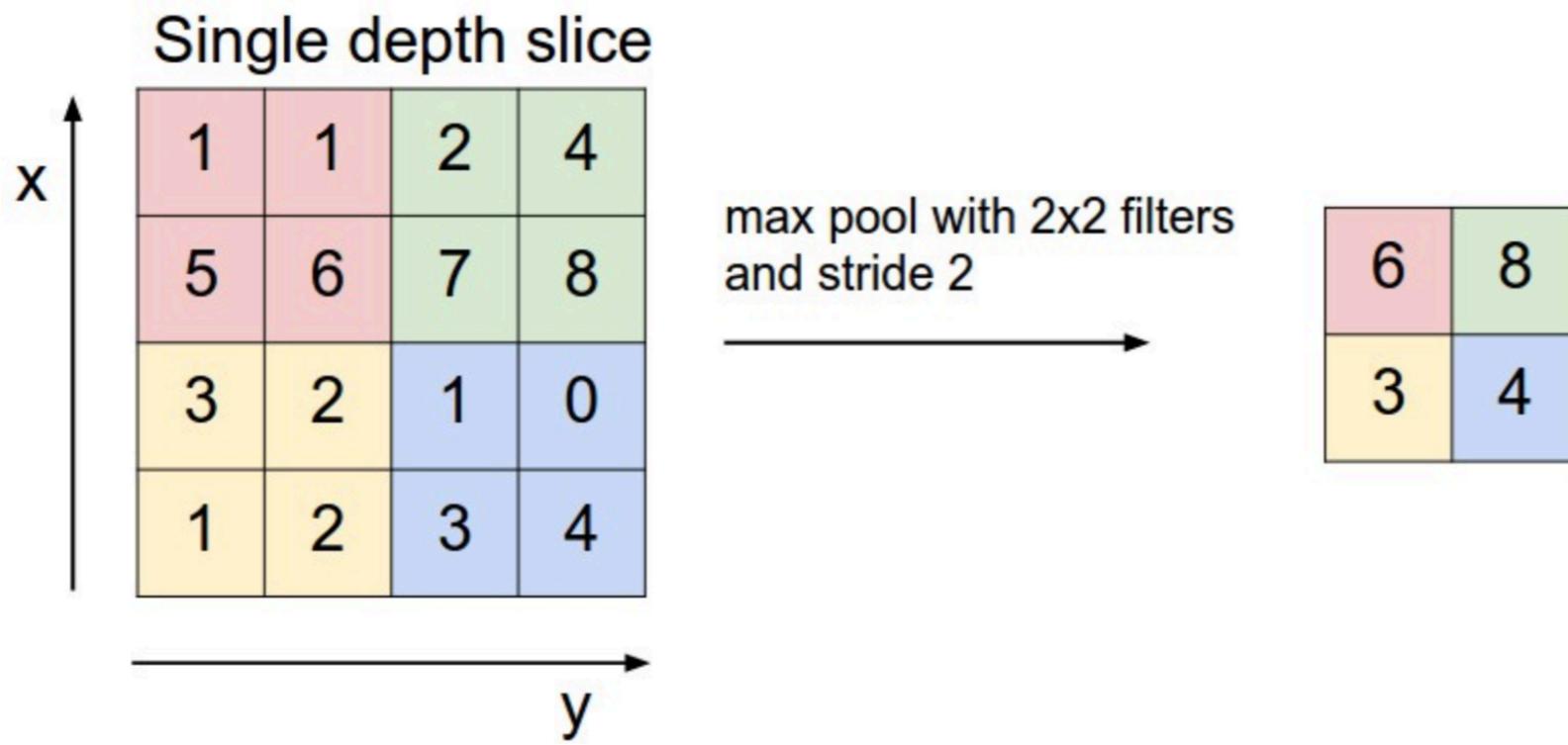


Stride снижает размерность (экономя тем самым вычислительные ресурсы), не теряя при этом много информации, поскольку изображения обладают свойством локальной коррелированности пикселей — соседние пиксели, как правило, не сильно отличаются друг от друга.

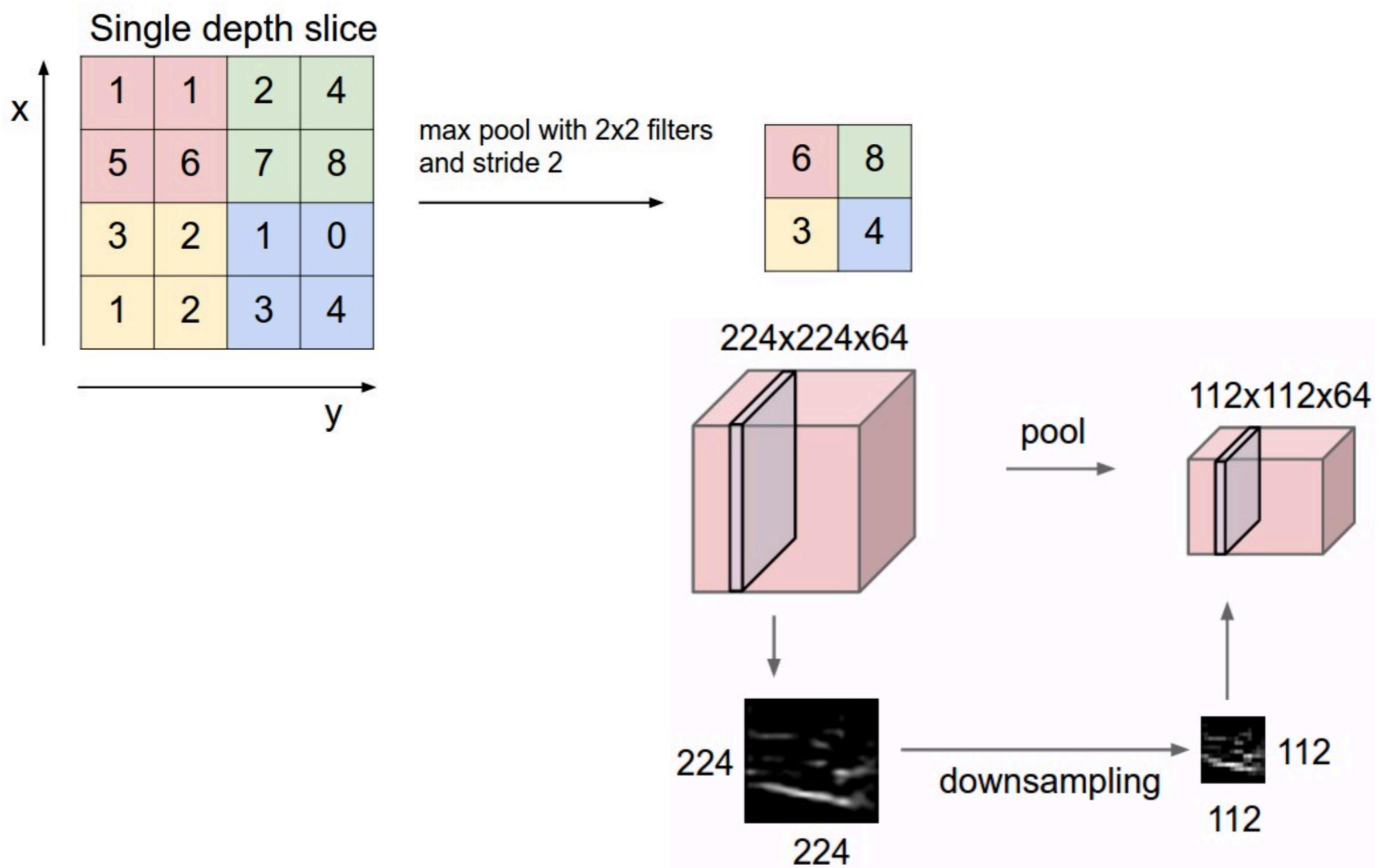


Padding используют, чтобы пространственная размерность картинки не уменьшалась после применения свертки или уменьшалась не так быстро

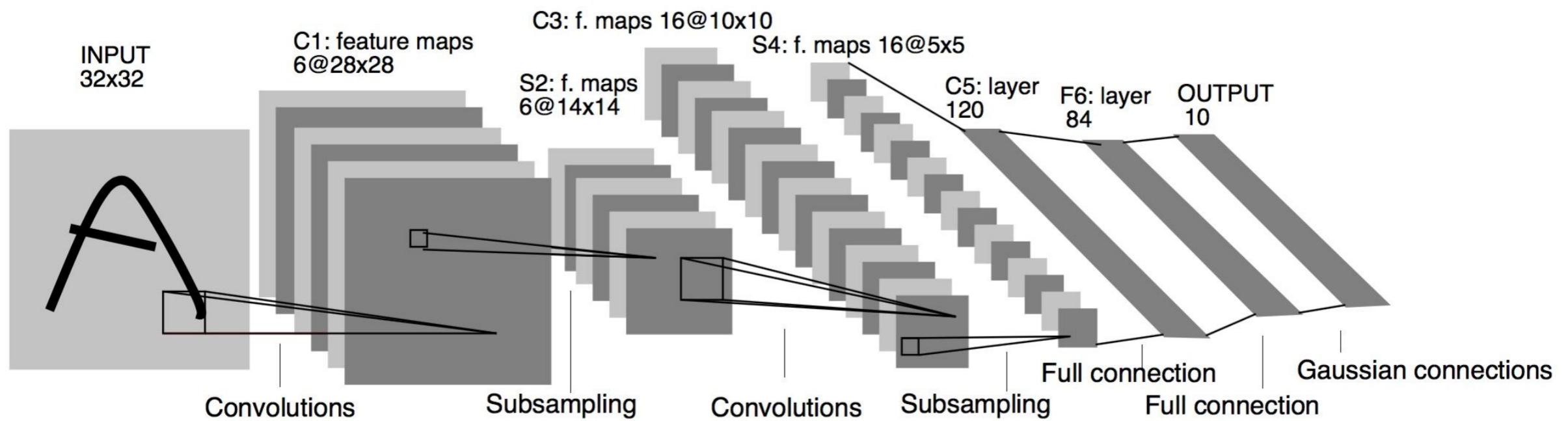
# Операция пулинга (pooling)



# Операция пулинга (pooling)



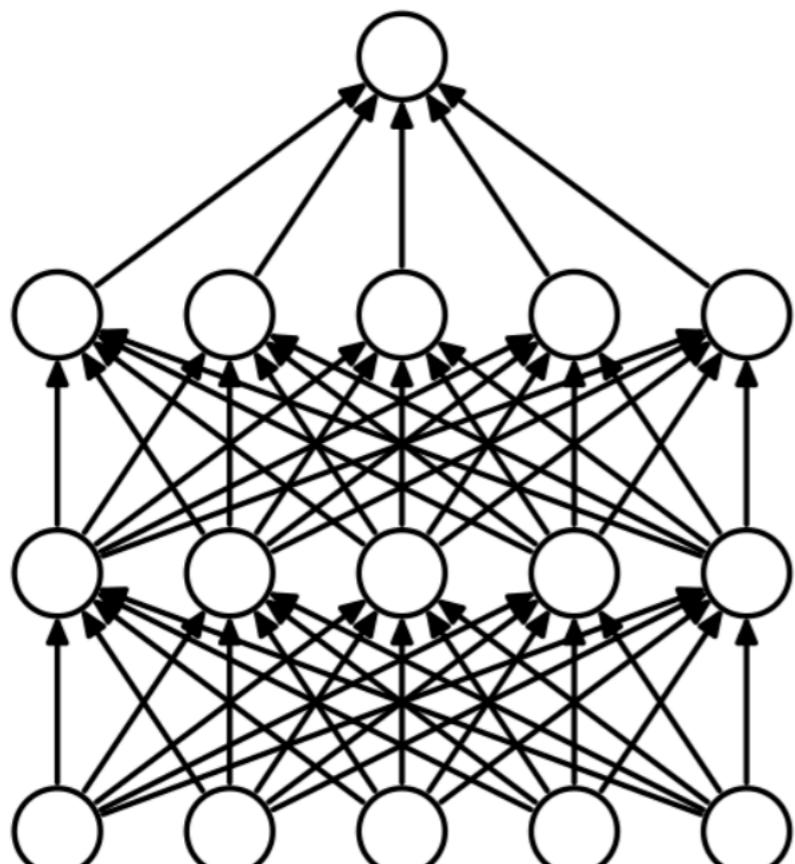
# Первая сверточная нейронная сеть



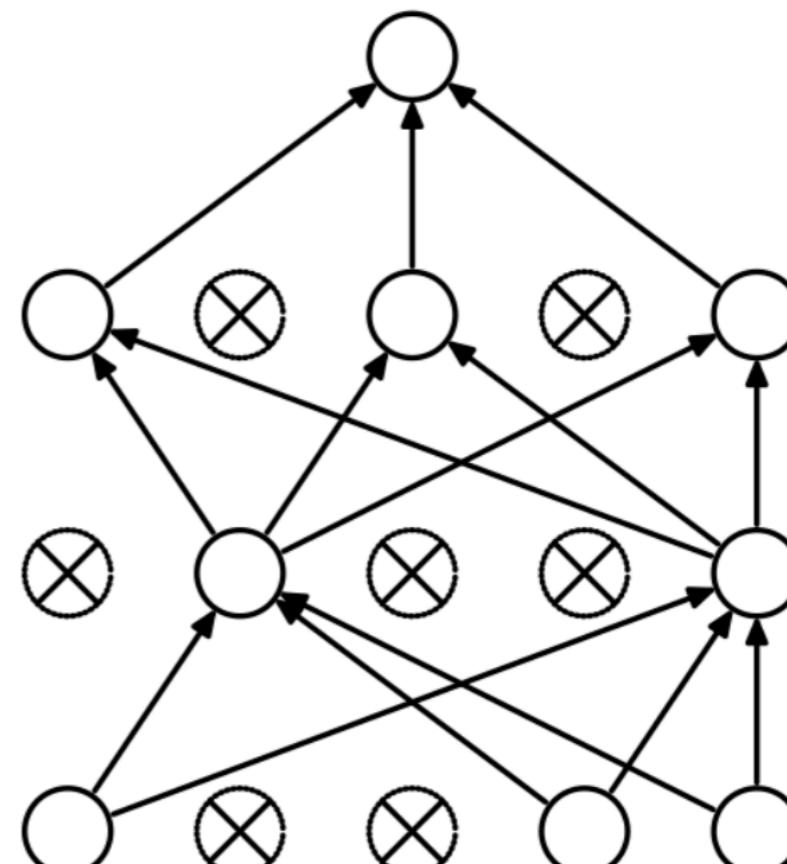
- Свертки  $5 \times 5$  со сдвигом 1 и пулинг  $2 \times 2$  со сдвигом 2
- 60 тысяч параметров
- Практическое применение: с помощью этой сети на почте США распознавали рукописные индексы

# Регуляризация

Dropout

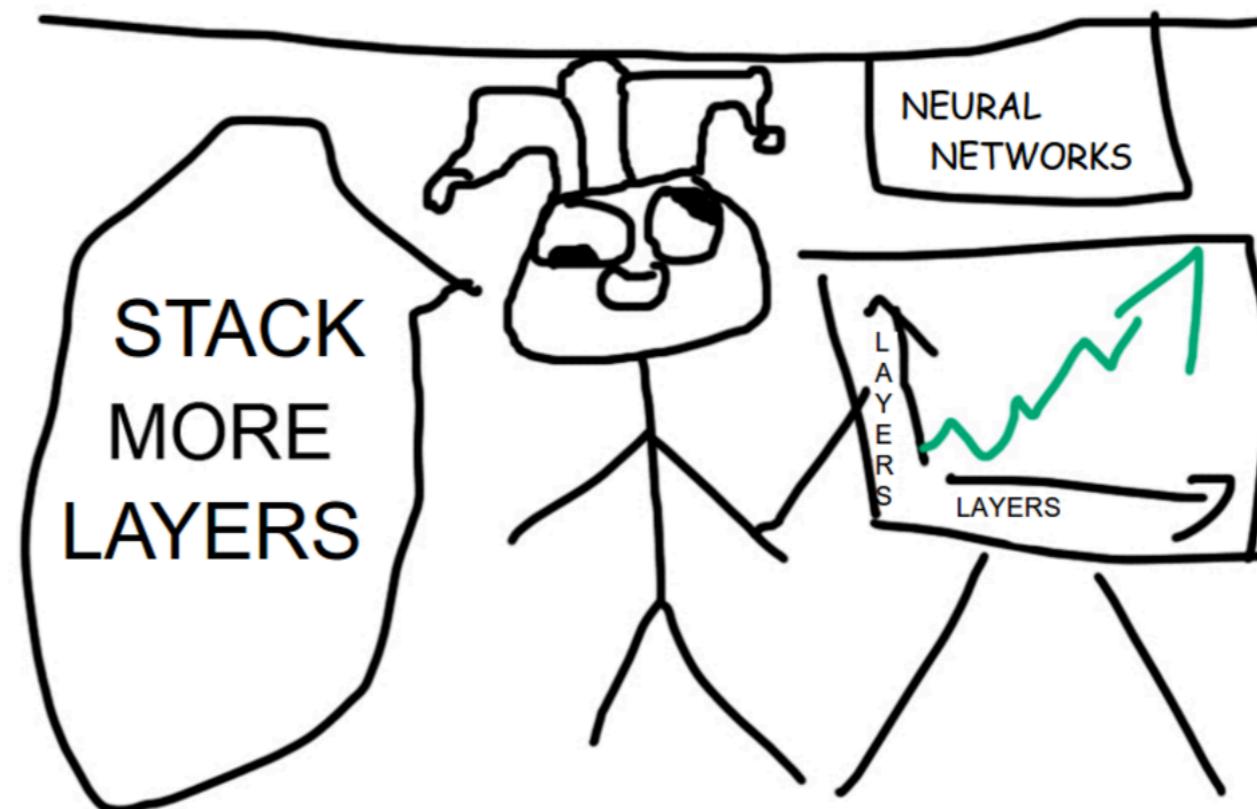
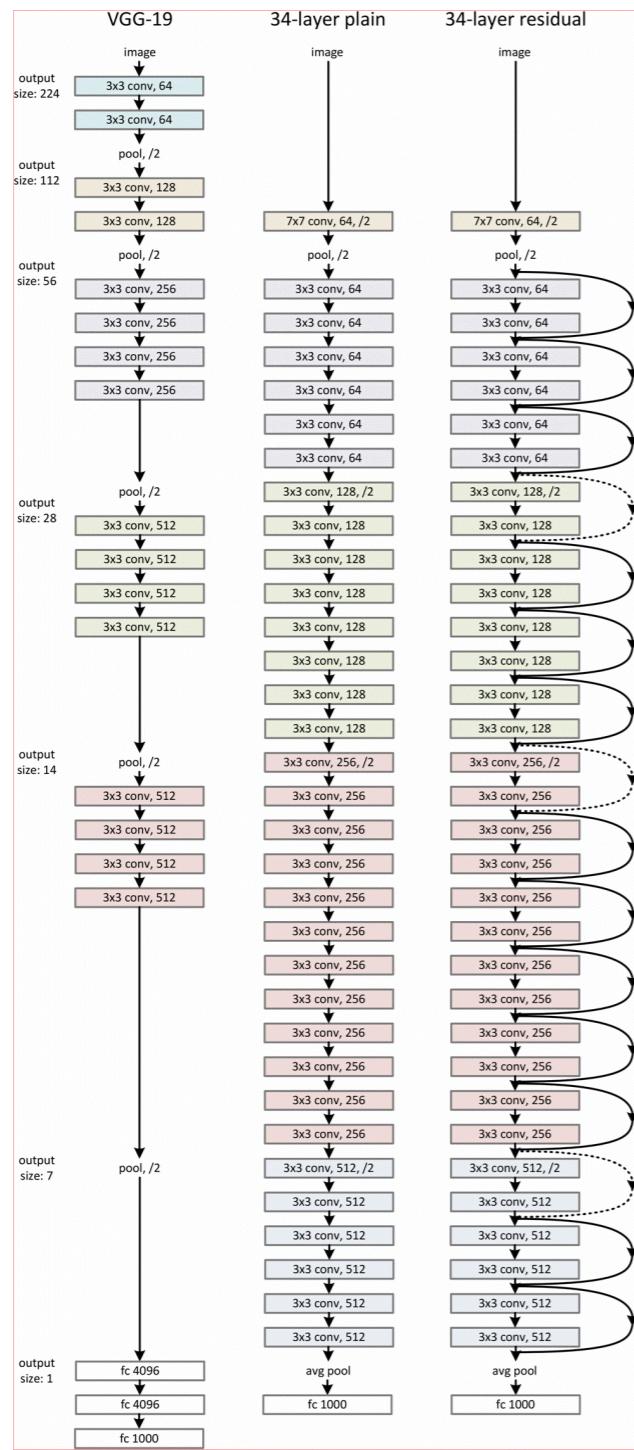


(a) Standard Neural Net



(b) After applying dropout.

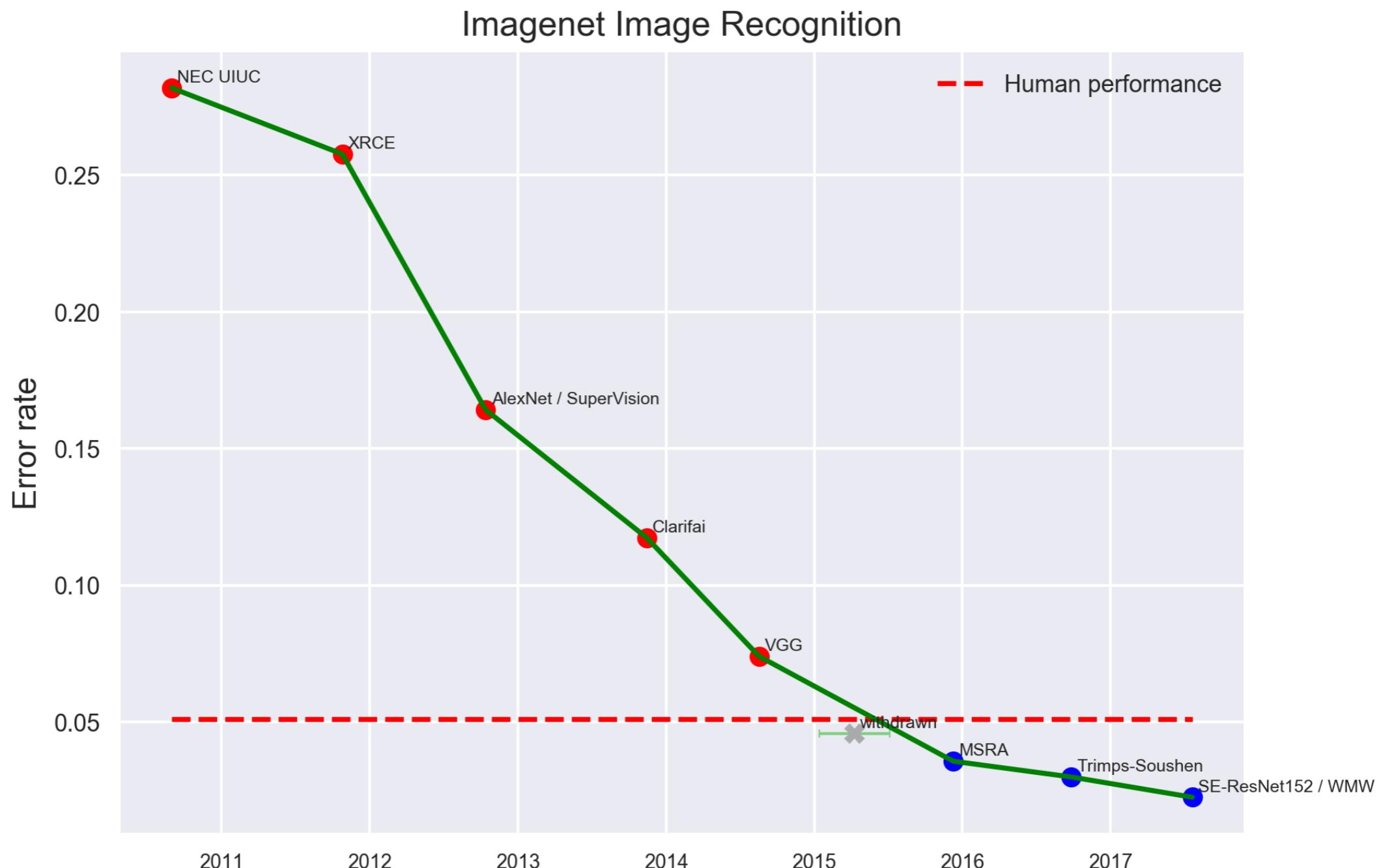
# Быстрее, выше, сильнее



Быстрее, выше, сильнее



# Превосходство NN над человеком



<https://www.eff.org/ai/metrics>

