Занятие 7
Введение в нейронные сети

Кантонистова Е.О.

ВШЭ, 2019

СХЕМА НЕЙРОНА В МОЗГЕ

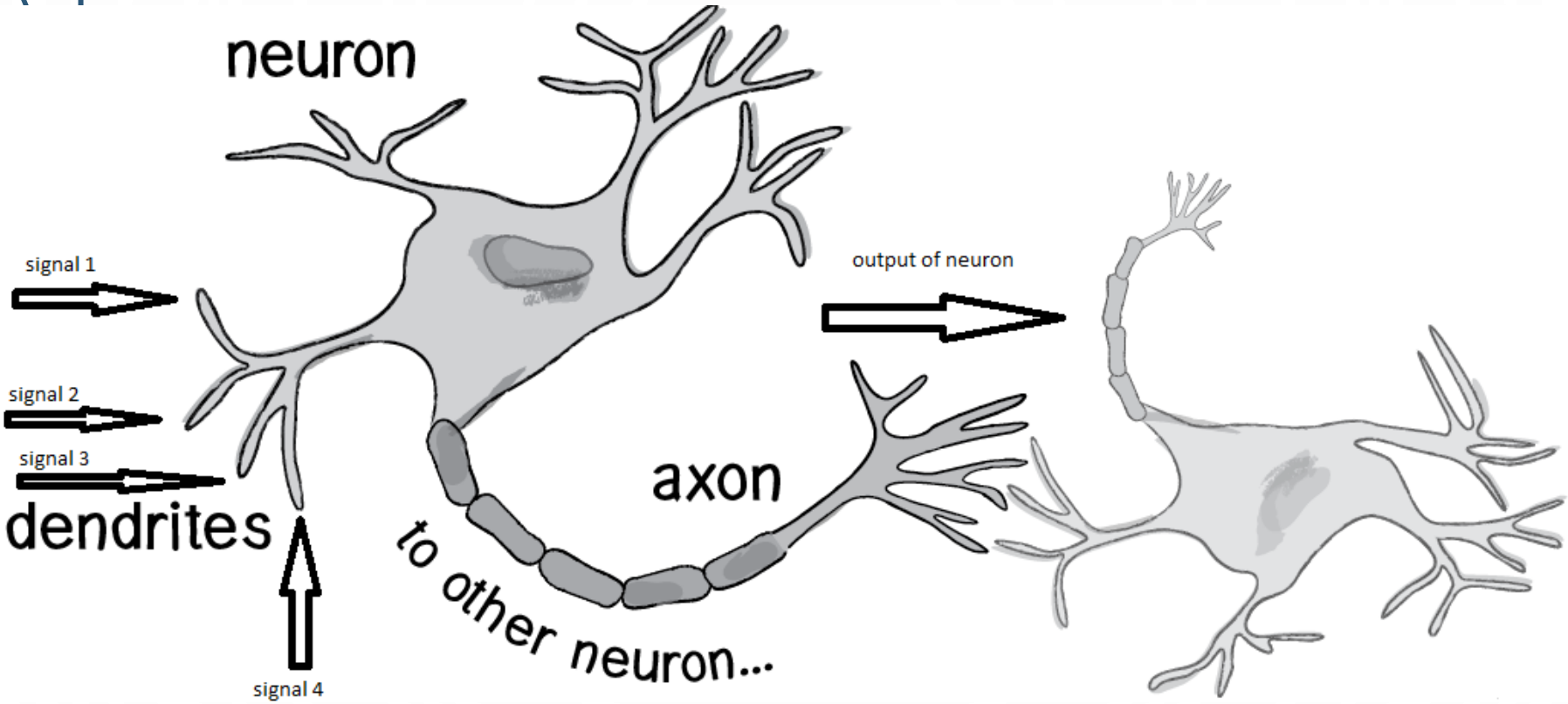
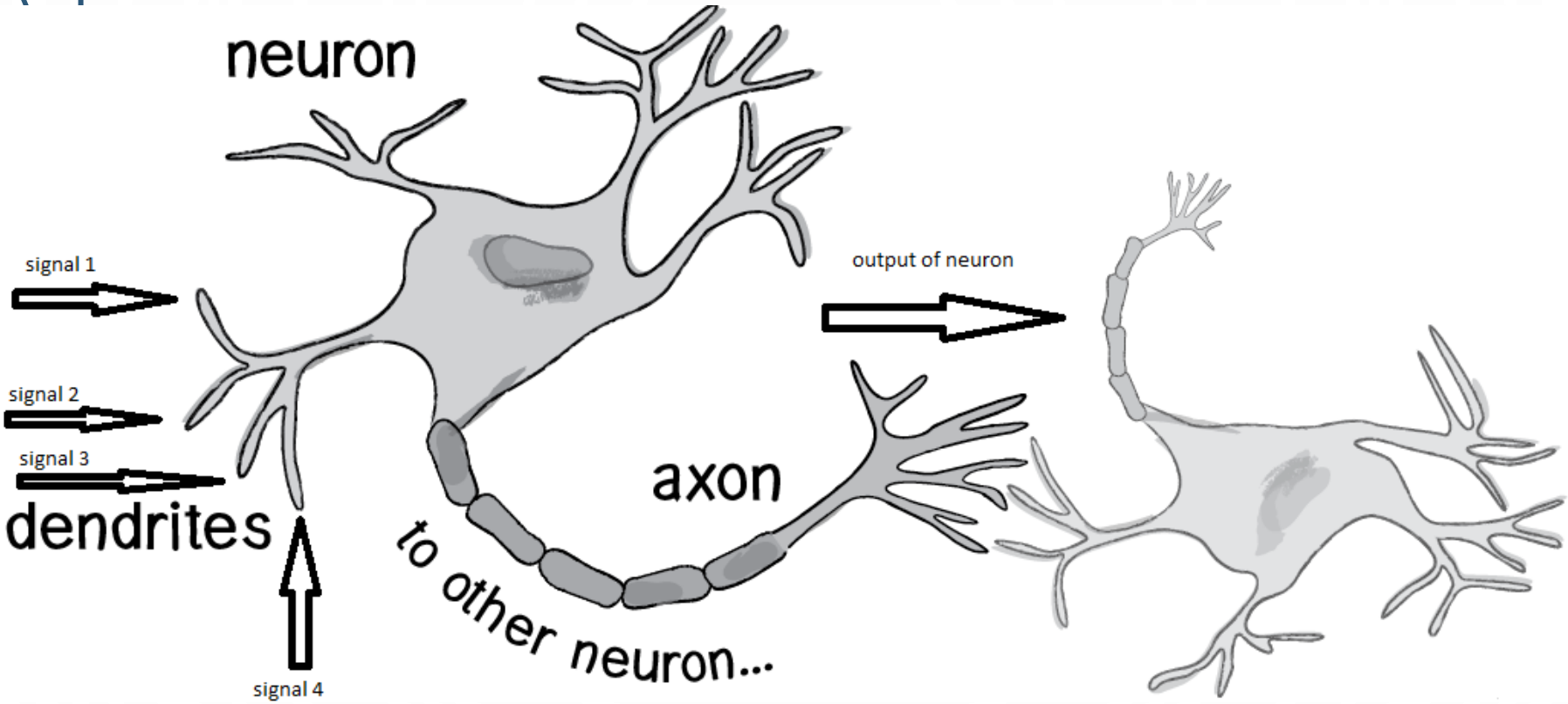


СХЕМА НЕЙРОНА В МОЗГЕ



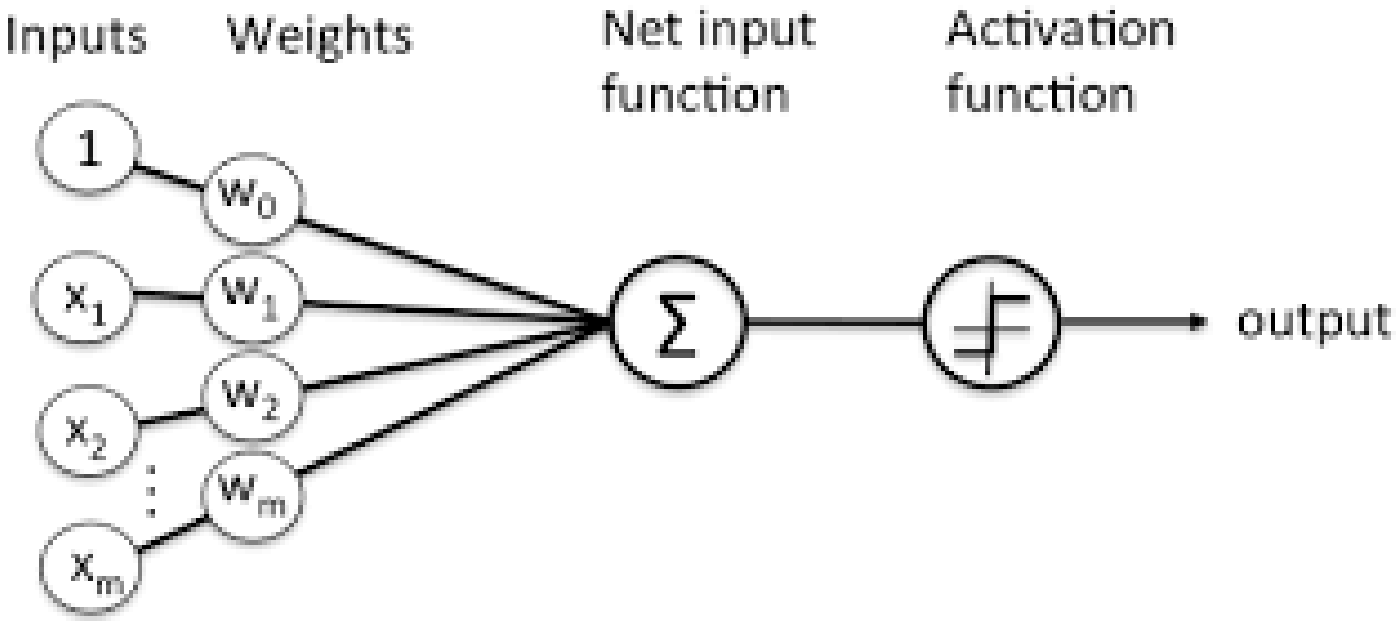
Линейная модель:

$$a(x, w) = \sigma(w_0 + \sum_{j=1}^{n} w_j x_j),$$

σ – функция активации

МОДЕЛЬ НЕЙРОНА

$$a(x,w) = \sigma(w_0 + \sum_{j=1}^{n} w_j x_j)$$



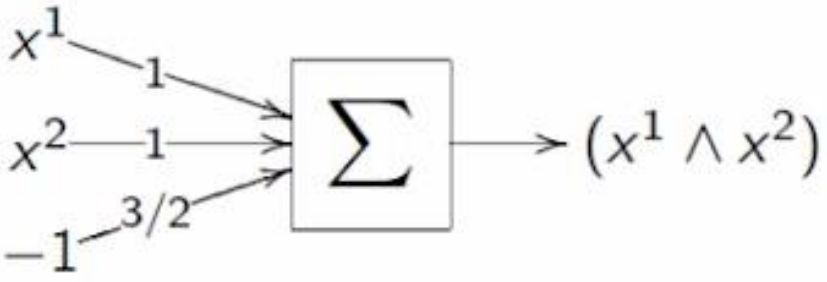
Функции активации:

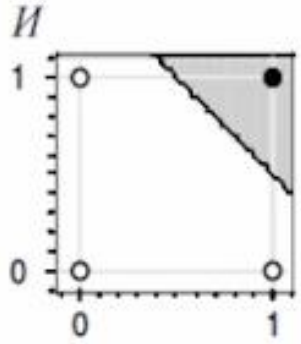
- $\sigma(z) = sign(z) \Rightarrow a(x,w) = sign[w_0 + \sum_{j=1}^{n} w_j x_j]$
- $\sigma(z) = \frac{1}{1+\exp(-z)} \Rightarrow a(x,w) = \frac{1}{1+\exp(-\left(w_0+\sum_{j=1}^{n} w_j x_j\right))}$

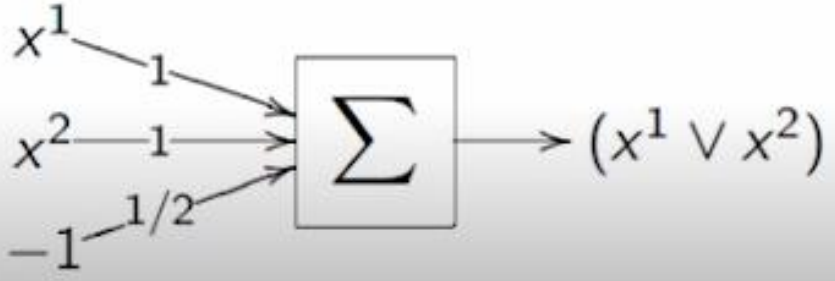
КАКИЕ ФУНКЦИИ РЕАЛИЗУЮТСЯ НЕЙРОНОМ?

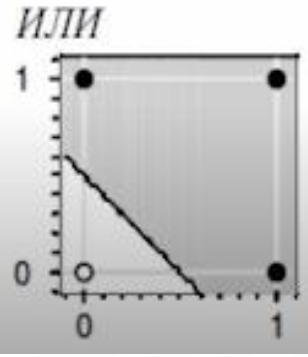
Функции И, ИЛИ, НЕ от бинарных переменных x^1 и x^2:

$$x^1 \wedge x^2 = \left[x^1 + x^2 - \tfrac{3}{2} > 0\right];$$
$$x^1 \vee x^2 = \left[x^1 + x^2 - \tfrac{1}{2} > 0\right];$$
$$\neg x^1 = \left[-x^1 + \tfrac{1}{2} > 0\right];$$







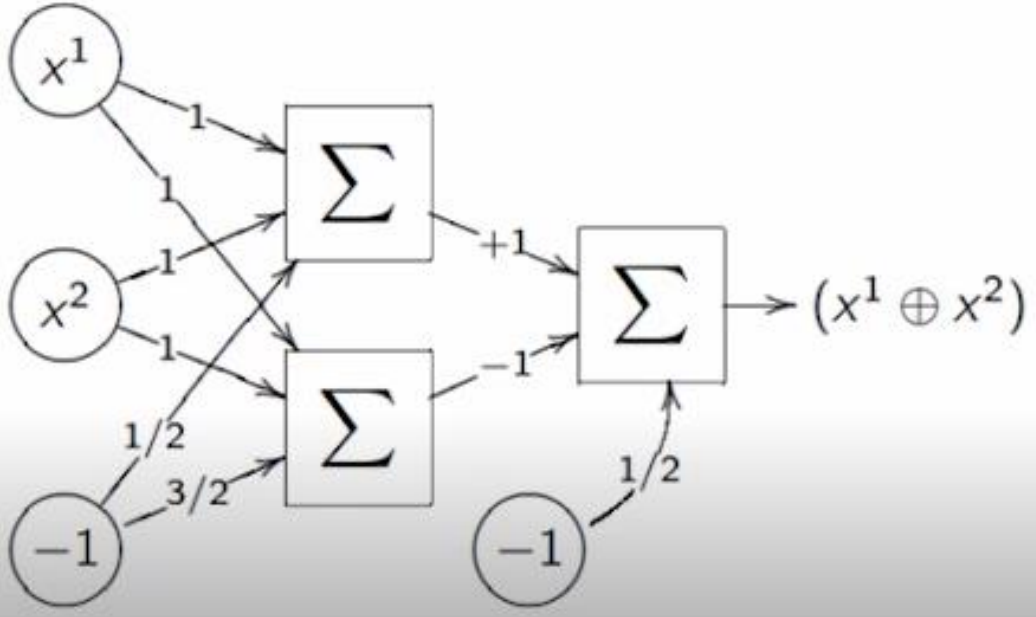


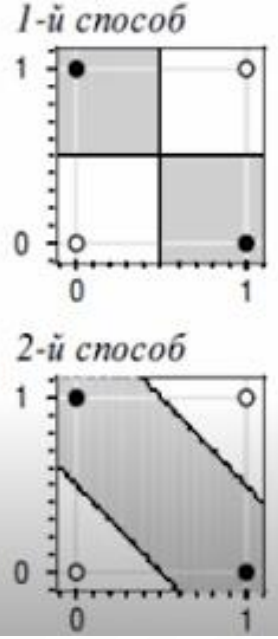
ПРИМЕР ФУНКЦИИ, НЕ РЕАЛИЗУЕМОЙ НЕЙРОНОМ

Функция $x^1 \oplus x^2 = [x^1 \neq x^2]$ не реализуема одним нейроном.

Два способа реализации:

- Добавлением нелинейного признака:
  $x^1 \oplus x^2 = \left[x^1 + x^2 - 2x^1x^2 - \frac{1}{2} > 0\right]$;
- Сетью (двухслойной суперпозицией) функций И, ИЛИ, НЕ:
  $x^1 \oplus x^2 = \left[(x^1 \vee x^2) - (x^1 \wedge x^2) - \frac{1}{2} > 0\right]$.



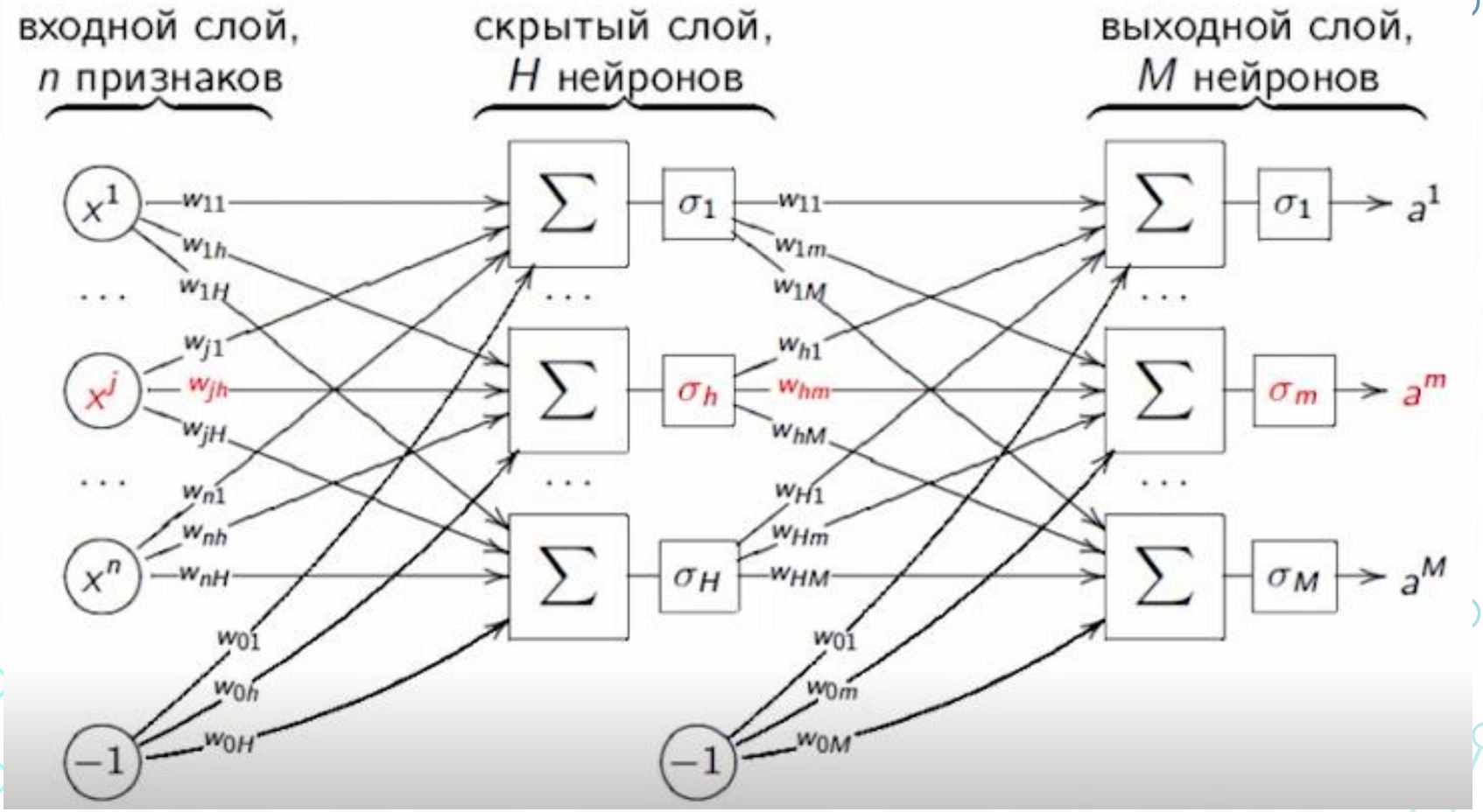


ДВУХСЛОЙНАЯ НЕЙРОННАЯ СЕТЬ

- **Теорема.** Любая булева функция может быть приведена к дизъюнктивной нормальной форме (т.е. к дизъюнкциям конъюнкций).
- **Теорема.** С помощью линейных операций и одной нелинейной функции активации можно приблизить любую непрерывную функцию с любой заданной точностью.

ПОЛНОСВЯЗНАЯ НЕЙРОННАЯ СЕТЬ

Рассмотрим двухслойную полносвязную нейронную сеть для задачи регрессии ($y \in \mathbb{R}^m$).



Вектор параметров модели: $w = (w_{jh}, w_{hm}) \in \mathbb{R}^{H(n+M+1)}$.

ЗАДАЧА ОПТИМИЗАЦИИ

Решаем задачу (L – функция потерь на объекте)

$$Q(w) = \sum_{i=1}^{l} L(w; x_i; y_i) \to \min_w$$

Градиентный спуск:

- Инициализируем w: $w^{(0)}$
- На N-м шаге: $w^{(N)} := w^{(N-1)} - \eta \nabla L(w^{(N-1)}; x; y)$

ЗАДАЧА ОПТИМИЗАЦИИ

Решаем задачу (L – функция потерь на объекте)

$$Q(w) = \sum_{i=1}^{l} L(w; x_i; y_i) \to \min_w$$

Градиентный спуск:

- Инициализируем w: $w^{(0)}$
- На N-м шаге: $w^{(N)} := w^{(N-1)} - \eta \nabla L(w^{(N-1)}; x; y)$

Если вычислять градиент обычным способом, то сложность составит квадрат от числа параметров.

BACKPROPAGATION

Теорема (метод обратного распространения ошибки).
Градиент функции потерь можно вычислить за линейное количество операций (по числу параметров).

BACKPROPAGATION

Теорема (метод обратного распространения ошибки).
Градиент функции потерь можно вычислить за линейное количество операций (по числу параметров).

Идея доказательства:

$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm} u^h(x_i)\right), \quad u^h(x_i) = \sigma_h\left(\sum_{j=0}^{n} w_{jh} x_j\right)$$

- Сначала вычислим $\varepsilon_i^m = \frac{\partial L_i(w)}{\partial a^m}$ и $\varepsilon_i^h = \frac{\partial L_i(w)}{\partial u^h}$.
- Ошибка на скрытом слое $\varepsilon_i^h = \sum_m \varepsilon_i^m \sigma_m' w_{hm}$

Тогда компоненты градиента:

- $\frac{\partial L_i(w)}{\partial w_{hm}} = \varepsilon_i^m \sigma_m' u^h(x_i)$, $\frac{\partial L_i(w)}{\partial w_{jh}} = \varepsilon_i^h \sigma_h' x_j$

АЛГОРИТМ BACKPROPAGATION

- инициализируем веса w_{jh}, w_{hm}
- повторяем до сходимости:

1. Выбираем объект $x_i \in X$

2. Прямой ход:

$$u_i^h := \sigma_h\left(\sum_{j=0}^{n} w_{jh} x_i^j\right), \; a_i^m := \sigma_m\left(\sum_{h=0}^{H} w_{hm} u_i^h\right),$$

$$\varepsilon_i^m := a_i^m - y_i^m, \; L_i := \sum_{m=1}^{M} (\varepsilon_i^m)^2.$$

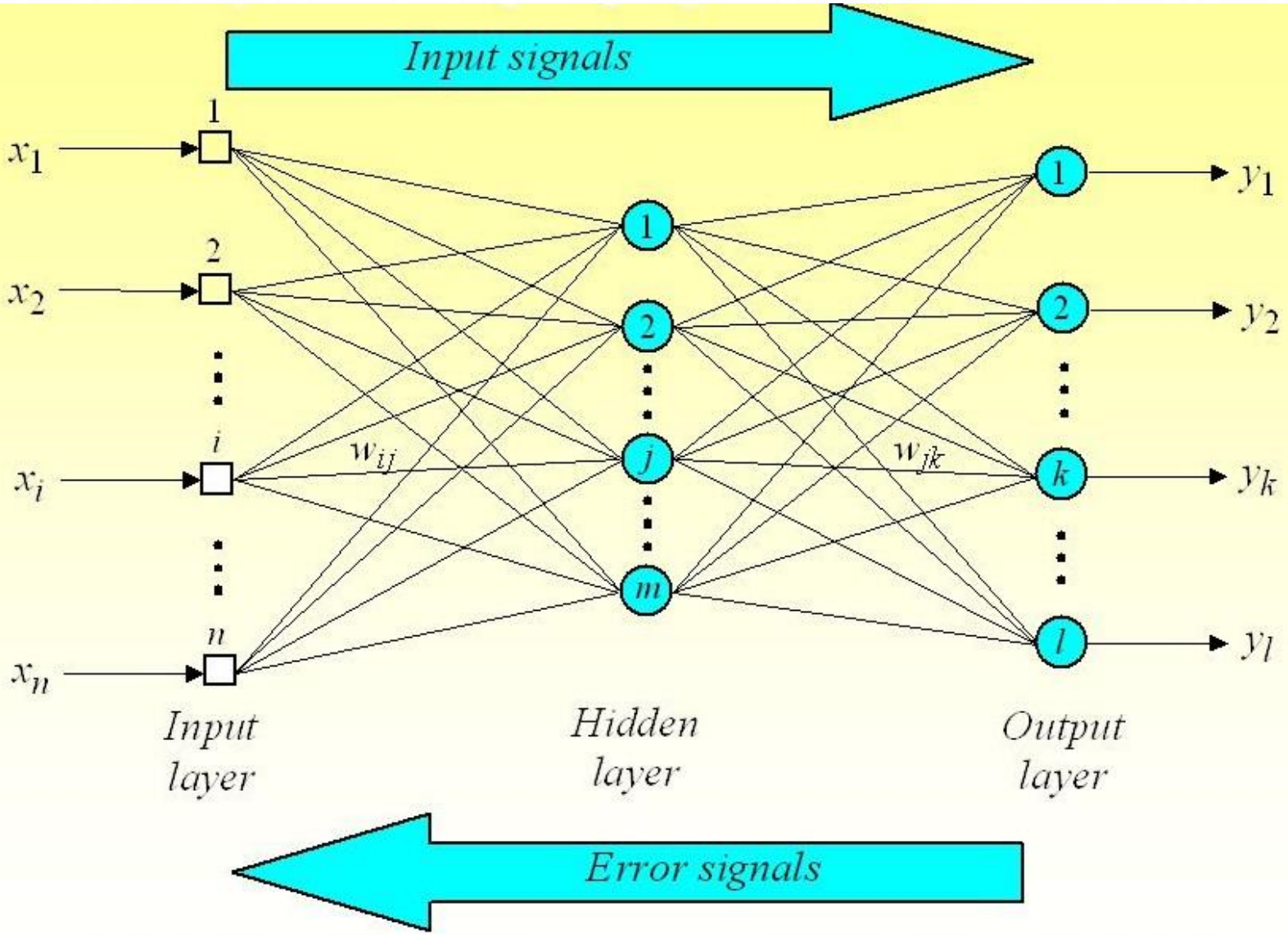
3. Обратный ход:

$$\varepsilon_i^h := \sum_{m=1}^{M} \varepsilon_i^m \sigma_m' u_i^h.$$

4. Градиентный шаг:

$$w_{hm} := w_{hm} - \eta \varepsilon_i^m \sigma_m' u_i^h, \; w_{jh} := w_{jh} - \eta \varepsilon_i^h \sigma_h' x_i^j.$$

BACKPROPAGATION



ПЛЮСЫ И МИНУСЫ BACKPROPAGATION

Плюсы:

- быстрое вычисление градиента
- метод обобщается на любые функции σ, L
- возможно динамическое обучение и распараллеливание

Минусы:

- возможна медленная сходимость
- застревание в локальных минимумах
- “паралич сети” (горизонтальные асимптоты σ)
- переобучение

ВАРИАНТЫ РЕШЕНИЯ ПРОБЛЕМ

- стандартные подходы градиентного спуска (инициализация весов, поиск величины градиентного шага, регуляризация)
- выбор функции активации в каждом нейроне
- выбор числа слоёв и числа нейронов
- выбор значимых связей

ФУНКЦИЯ ПОТЕРЬ В ЗАДАЧЕ КЛАССИФИКАЦИИ

Одна из часто используемых функций потерь – ***кросс-энтропия (log-loss)***.

Пусть алгоритм a предсказывает вероятности принадлежности к классам.

- Случай двух классов:

$$L(w; x; y) = -\sum_{i=1}^{l} y_i \log a(x_i) + (1 - y_i) \log(1 - a(x_i))$$

- Общий случай:

$$L(w; x; y) = -\sum_{i=1}^{l} \sum_{k=1}^{K} [y_i = k] \cdot \log(P(y_i = k | x_i, w))$$

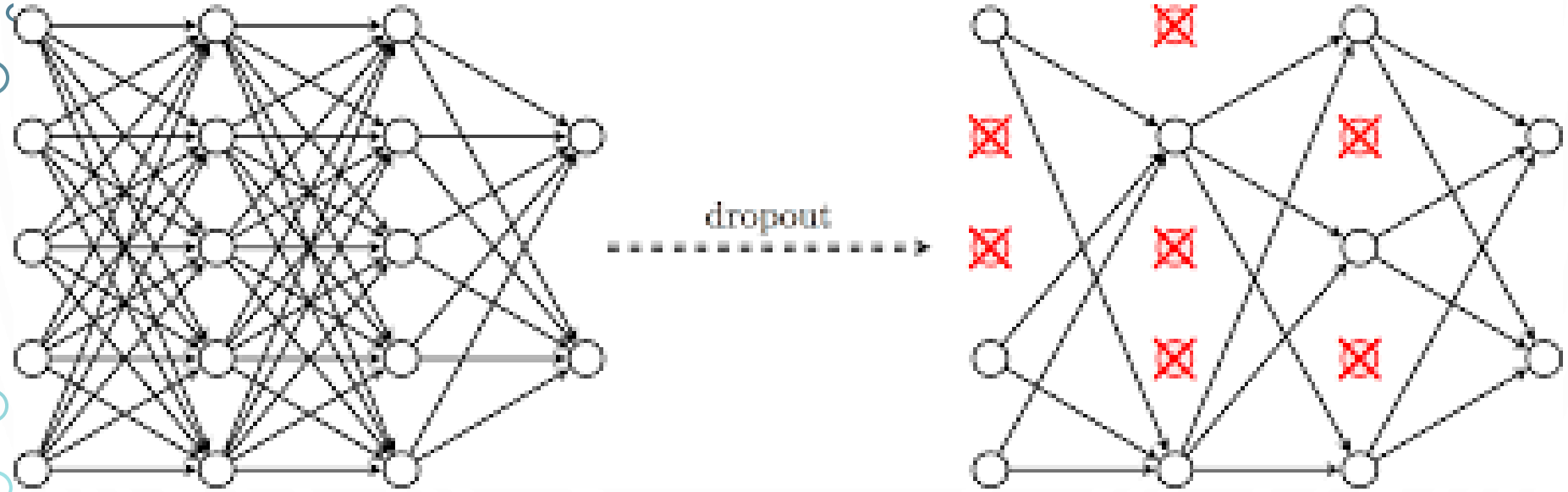
DROPOUT

Идея: *на каждом шаге обучения из нейронной сети выбрасываются нейроны (каждый нейрон выбрасывается с вероятностью p).*

Почему это работает:

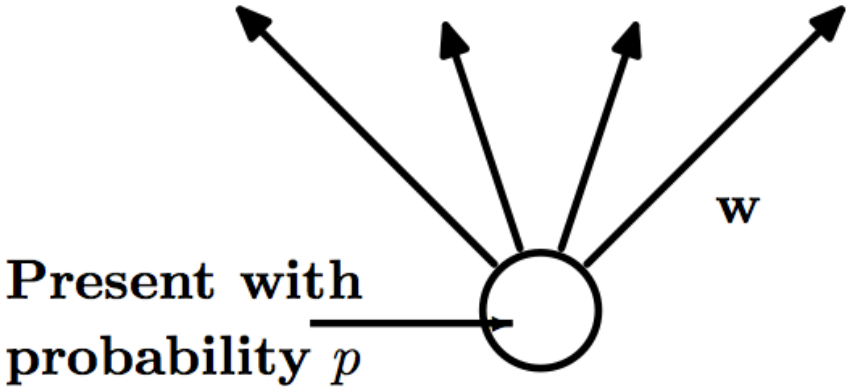
- производная, полученная каждым параметром, сообщает ему, как он должен измениться, чтобы, учитывая деятельность остальных блоков, минимизировать функцию конечных потерь
- это может привести к чрезмерной совместной адаптации (co-adaptation), что, в свою очередь, приводит к переобучению
- dropout предотвращает совместную адаптацию (дропаут – своего рода регуляризация)

DROPOUT

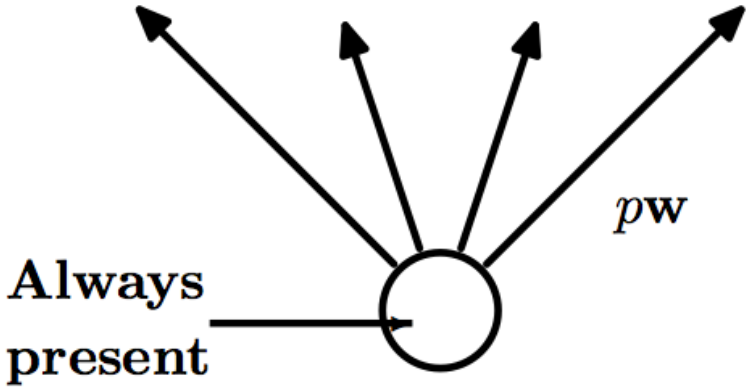


DROPOUT

Dropout на этапе обучения и на этапе применения нейронной сети:



(a) At training time

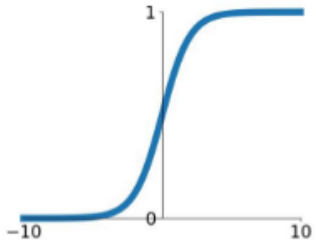


(b) At test time

ФУНКЦИИ АКТИВАЦИИ

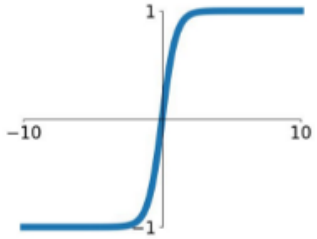
Sigmoid

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



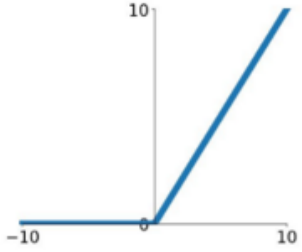
tanh

$$\tanh(x)$$



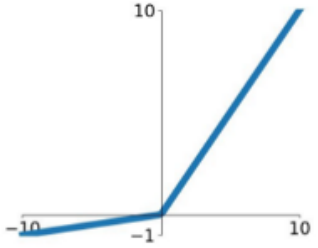
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

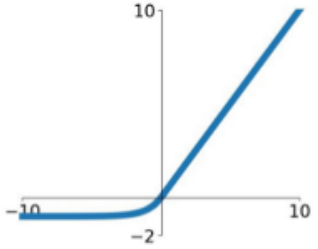


Maxout

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

ELU

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



НЕЙРОННЫЕ СЕТИ В КОМПЬЮТЕРНОМ ЗРЕНИИ

What We See

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

What Computers See

НЕЙРОННЫЕ СЕТИ В КОМПЬЮТЕРНОМ ЗРЕНИИ

What We See

What Computers See

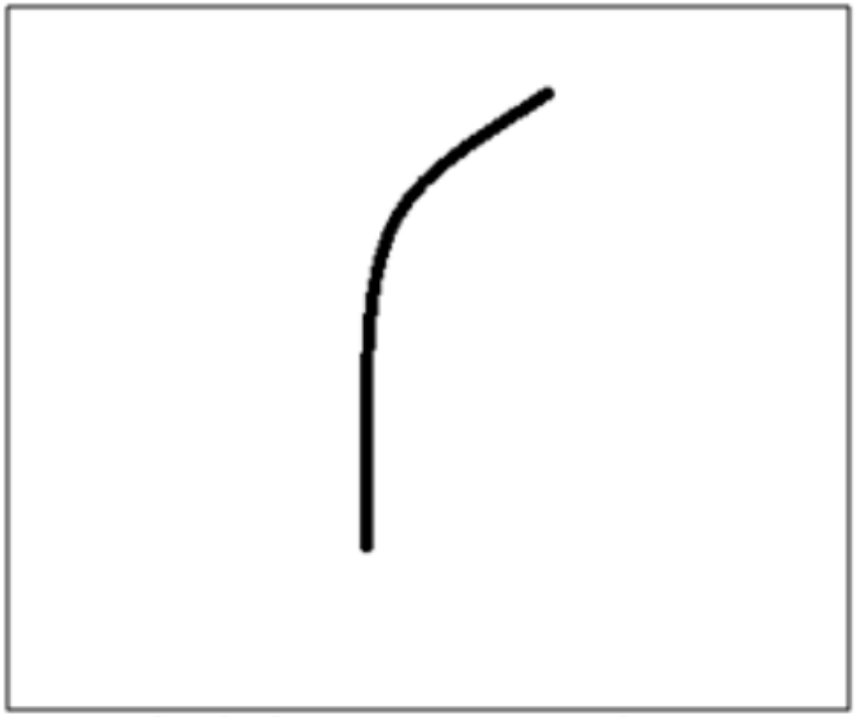
- Мозг человека идентифицирует изображение как набор некоторых форм (набор кривых). Эти формы инвариантны относительно сдвигов и поворотов.

СВЁРТКА

- Свёртка – это фильтр, накладываемый на изображение. Фильтр помогает детектировать участки изображения с некоторыми свойствами.

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

СВЁРТКА

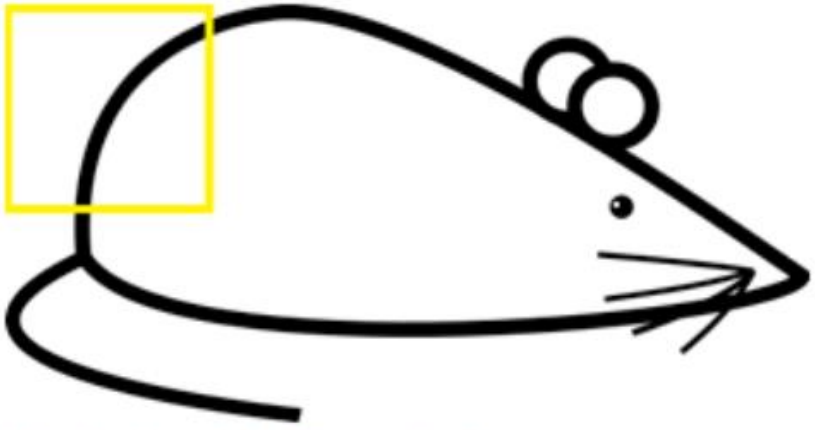
- Пусть свёртка задана матрицей G на сетке $(-s, s) \times (-t, t)$.
- Пусть изображение задано матрицей F.

Тогда результатом свёртки в точке (i, j) изображения будет:

$$f(i,j) = \sum_{k=-s}^{s} \sum_{l=-t}^{t} F(i+k, j+l) \cdot G(k,l)$$

ПРИМЕР

Original image



Visualization of the filter on the image

https://habr.com/post/309508/

ПРИМЕР

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	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

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

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	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

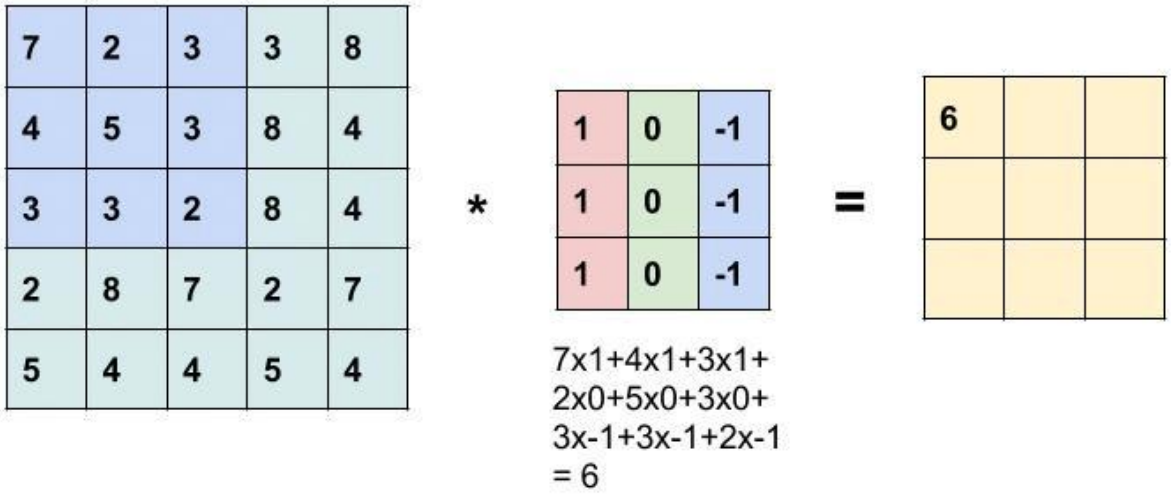
Pixel representation of filter

Multiplication and Summation = 0

СВЁРТОЧНЫЙ СЛОЙ

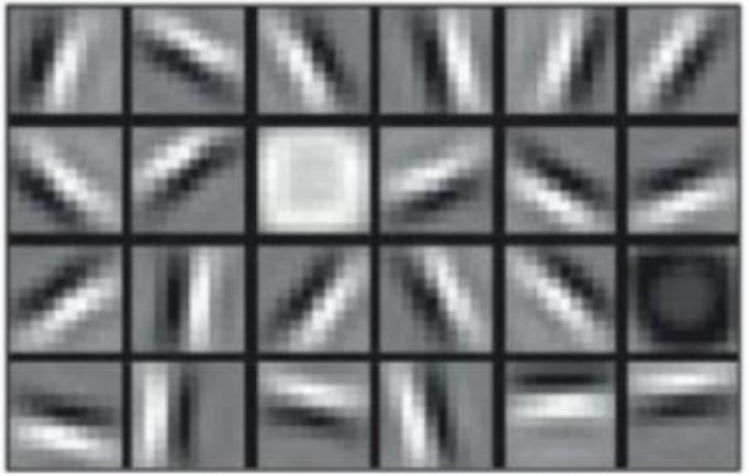
- Свёрточный слой – это фильтр. Он применяется к каждому участку изображения.
- Коэффициенты фильтра (значения в матрице фильтра) являются параметрами и подбираются в процессе обучения.

СВЁРТКА (ПРИМЕР)



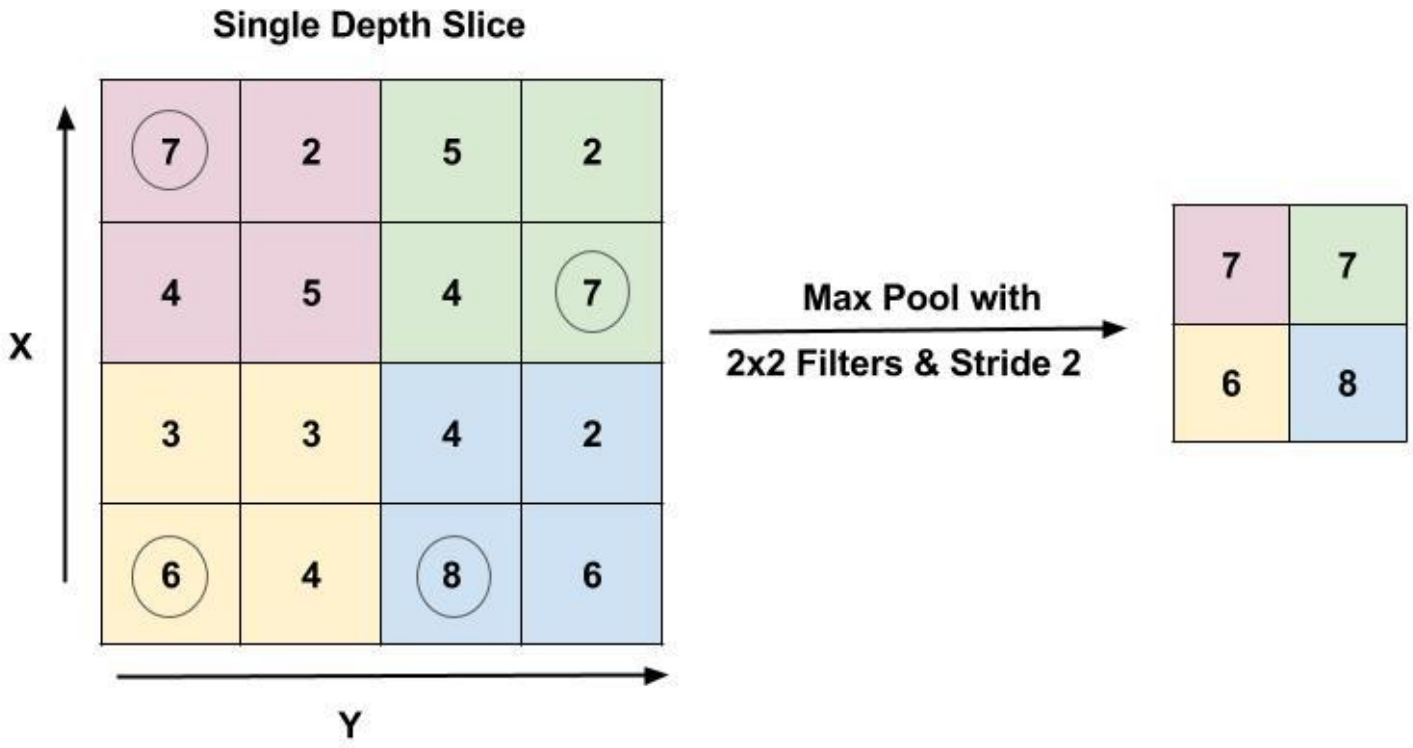
СВЁРТОЧНАЯ НЕЙРОННАЯ СЕТЬ

Свёрточные слои учат иерархические признаки для изображений.



MAX POOLING СЛОЙ

- используется для уменьшения количества параметров

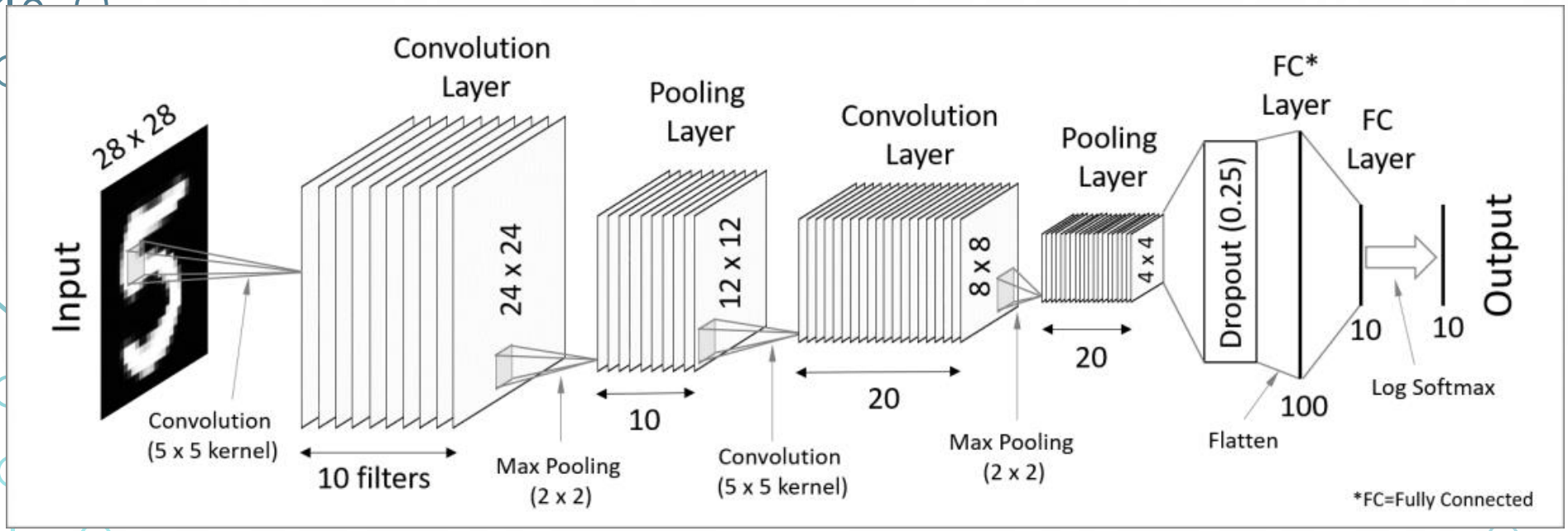


ЗАДАЧА КЛАССИФИКАЦИИ ЦИФР

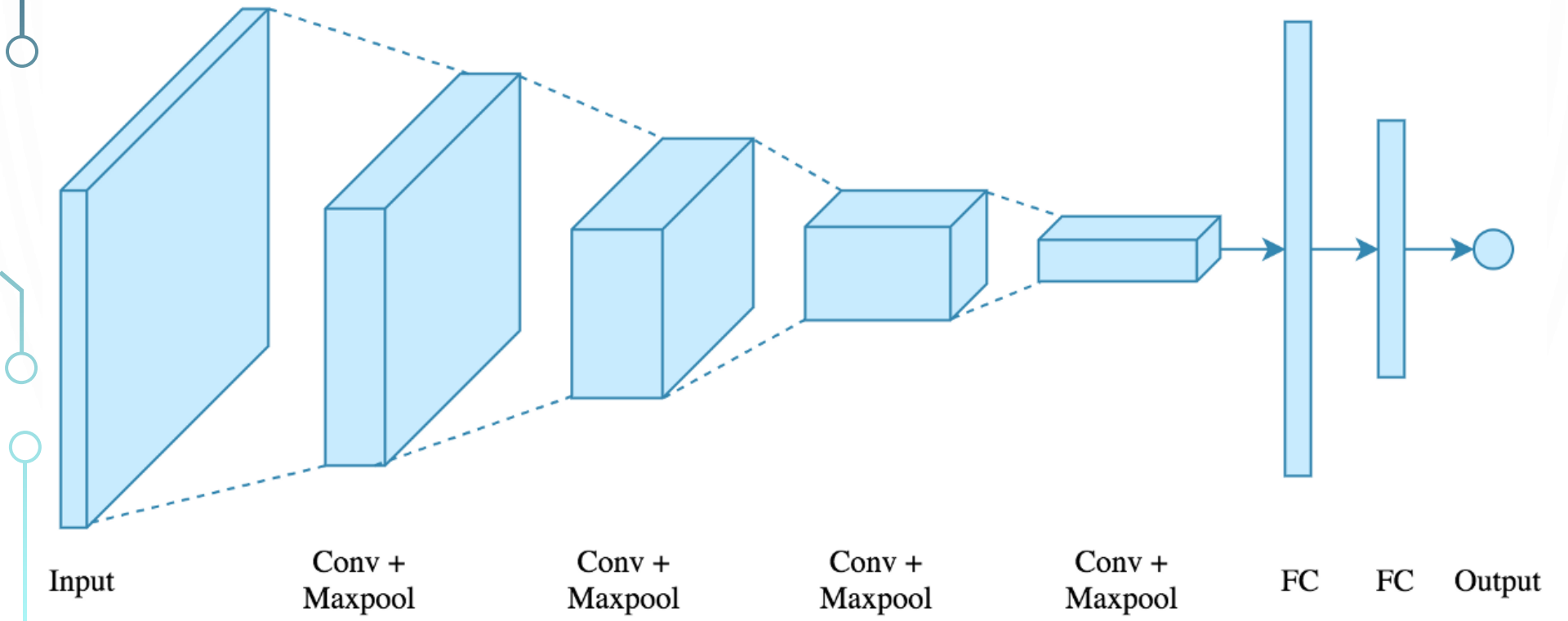
- Датасет: mnist

ЗАДАЧА КЛАССИФИКАЦИИ ЦИФР

Архитектура нейронной сети, решающей данную задачу:



СТАНДАРТНАЯ АРХИТЕКТУРА СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ



VGG

Архитектура нейронной сети для задачи классификации изображений. Обучена на ImageNet.