



Deep Learning School

# Лекция 7

Введение в нейронные сети

# План лекции

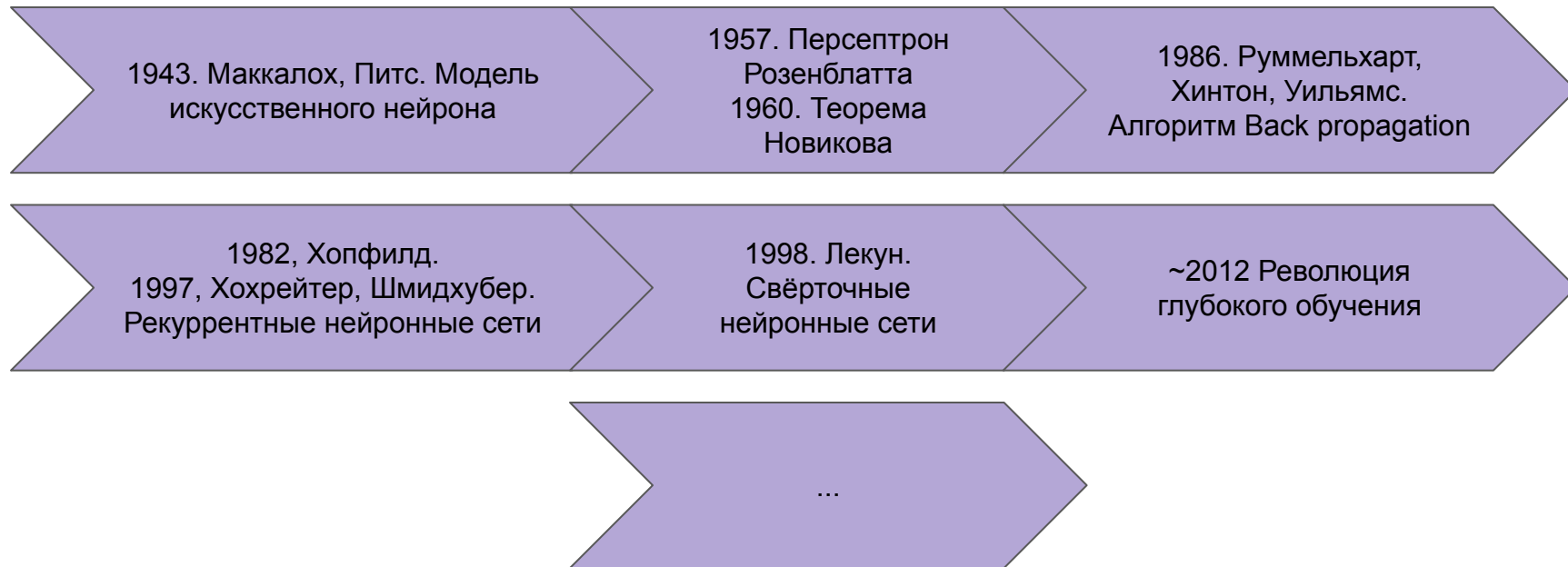


- Немного истории
- Модель нейрона
- Полносвязная нейронная сеть
- Обучение нейронных сетей
  - Стохастический градиентный спуск
  - Алгоритм Back-Propagation
- Фреймворки Deep Learning

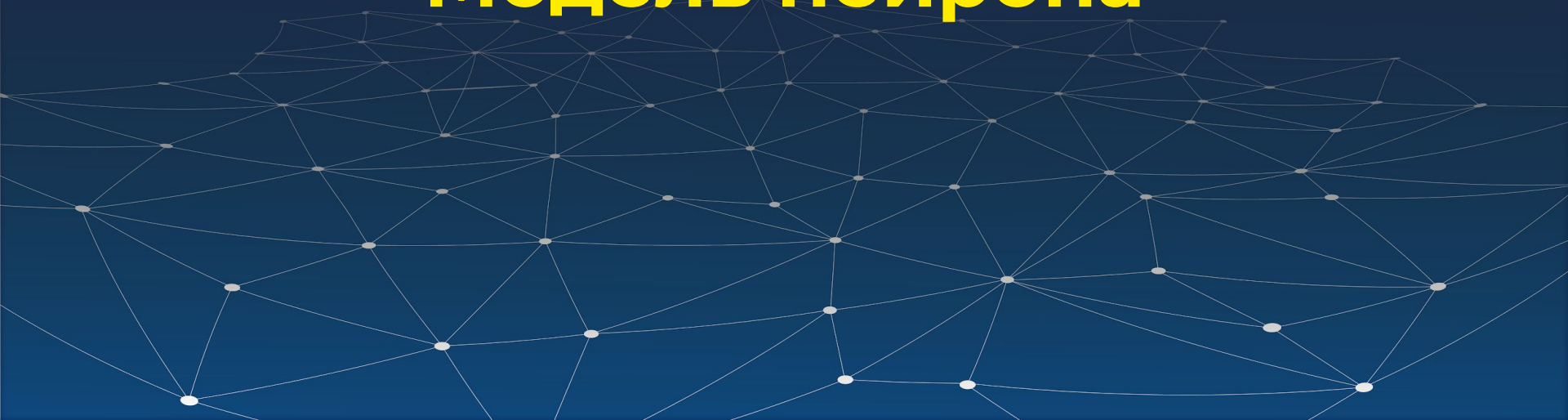
# История нейронных сетей



# История развития нейронных сетей

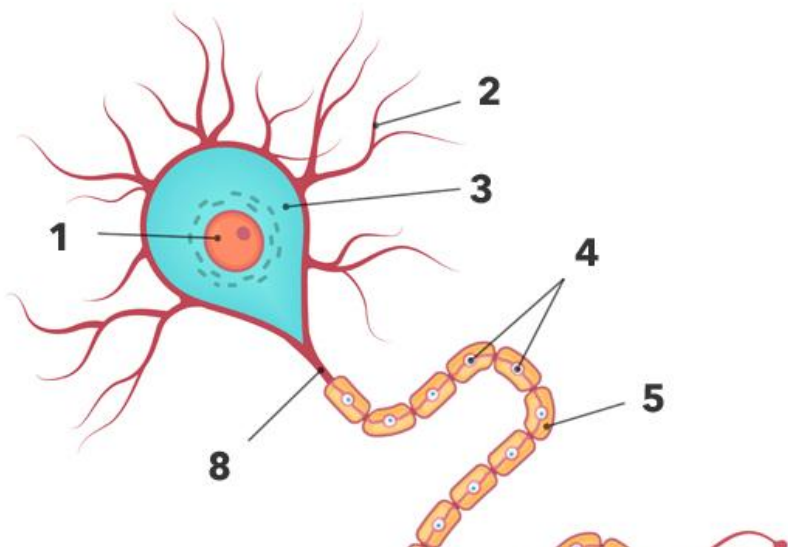


# Модель нейрона



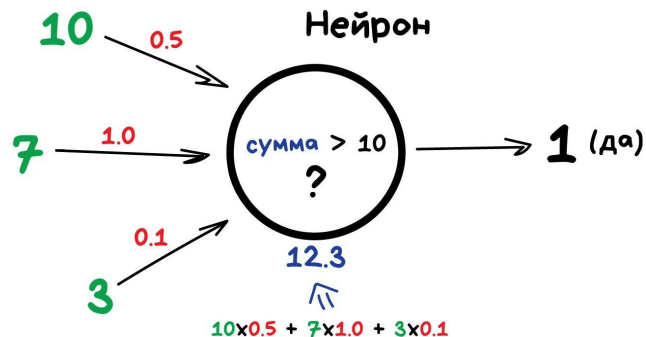
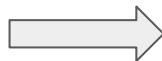
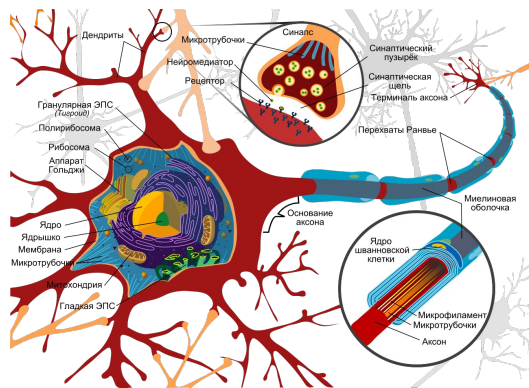
# Модель человеческого нейрона

- Внутри мозга информация передаётся по связям между *нейронами* химическим веществом — нейротрансмиттером
- Нейротрансмиттер поступает в тело нейрона по специальным отросткам — *дендритам*
- нейротрансмиттера нейрон  
ту другим нейронам

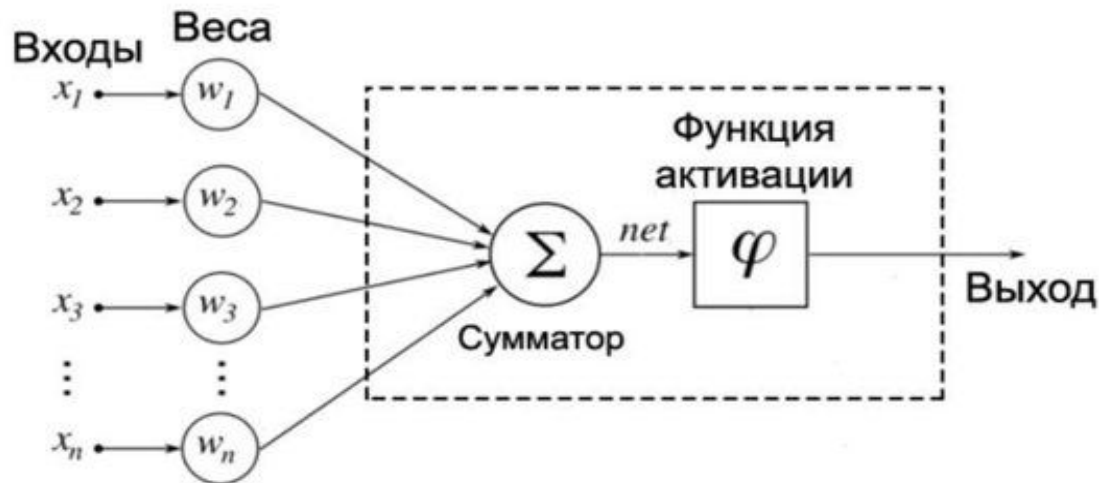


# Модель человеческого нейрона

- Внутри нейронной сети информация передаётся в виде чисел
- Каждый нейрон имеет несколько взвешенных рёбер-входов. Сигнал от каждого из входных нейронов домножается на вес ребра. Сигналы от входных нейронов складываются. При накоплении определённого значения происходит “пробой”
- В случае пробоя сигнал передаётся другим нейронам



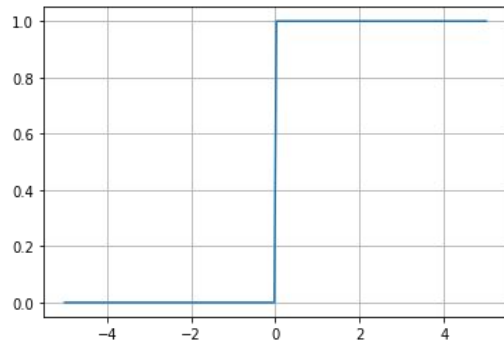
# Модель нейрона



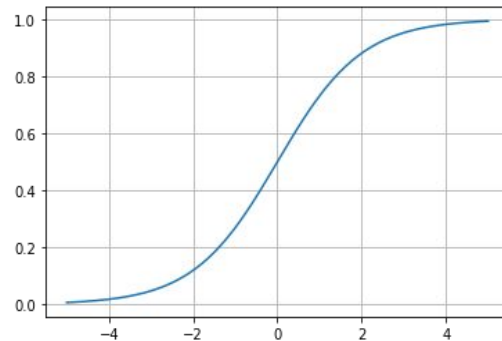
Нейрон определяет функцию  $y_w(x) = \varphi(w^\top x (+b))$



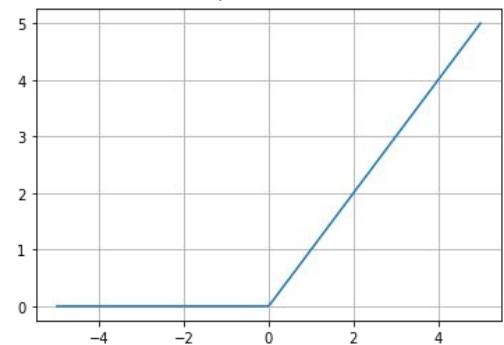
# Функции активации



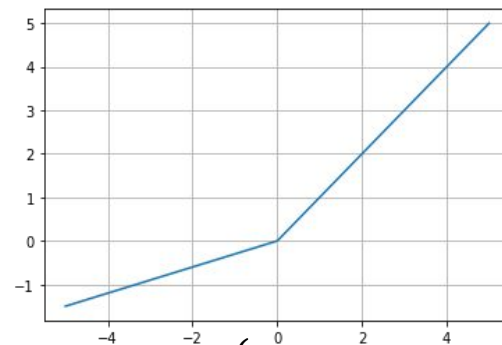
$$\varphi(t) = \begin{cases} 1, & t \geq 0 \\ 0, & \text{otherwise} \end{cases}$$



$$\varphi(t) = \frac{1}{1+e^{-t}}$$

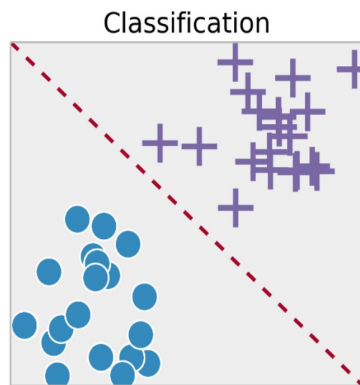


$$\varphi(t) = \max(t, 0)$$

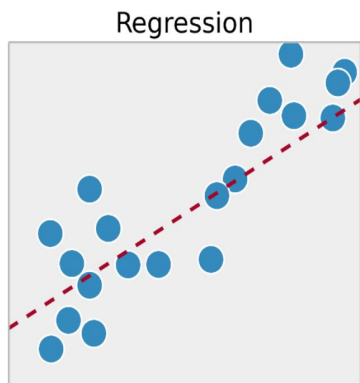
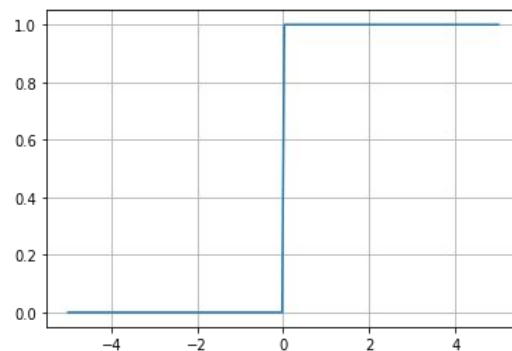
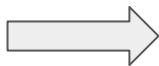


$$\varphi(t) = \begin{cases} \alpha t, & t \geq 0 \\ \beta t, & t < 0 \end{cases}$$

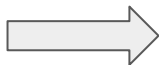
# Линейные алгоритмы



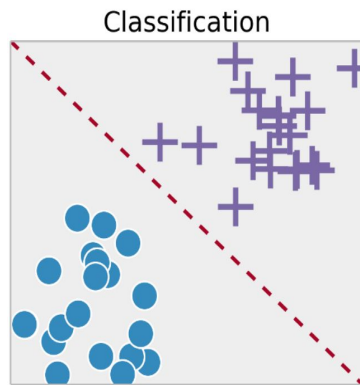
$$y(x) = \text{sign}(\langle w, x \rangle)$$



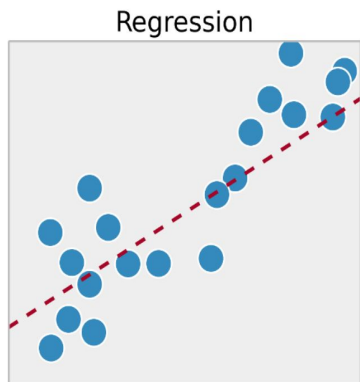
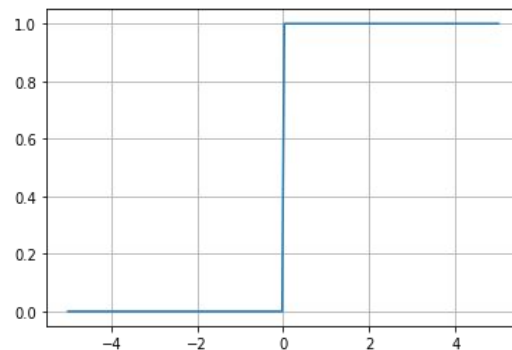
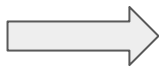
$$y(x) = \langle w, x \rangle$$



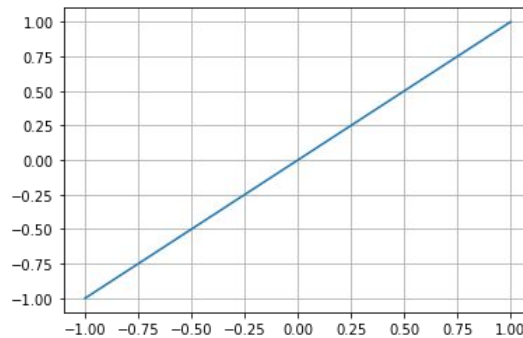
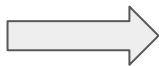
# Линейные алгоритмы



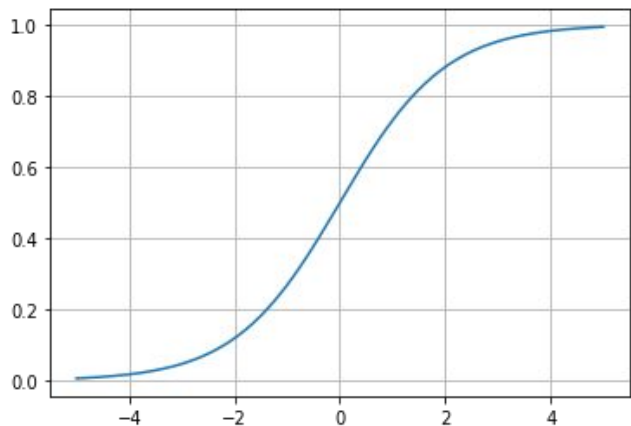
$$y(x) = \text{sign}(\langle w, x \rangle)$$



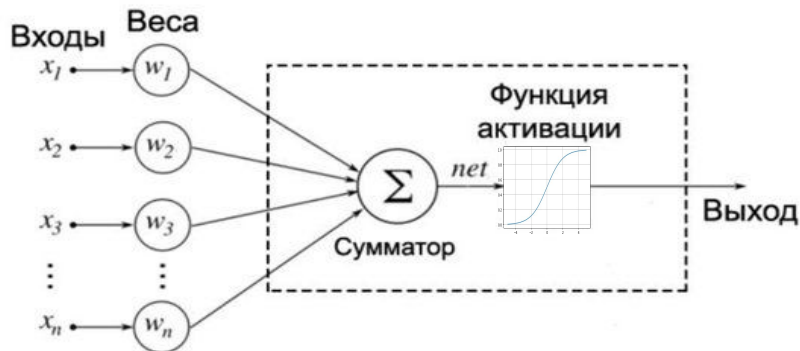
$$y(x) = \langle w, x \rangle$$



# Как изобразить логистическую регрессию?

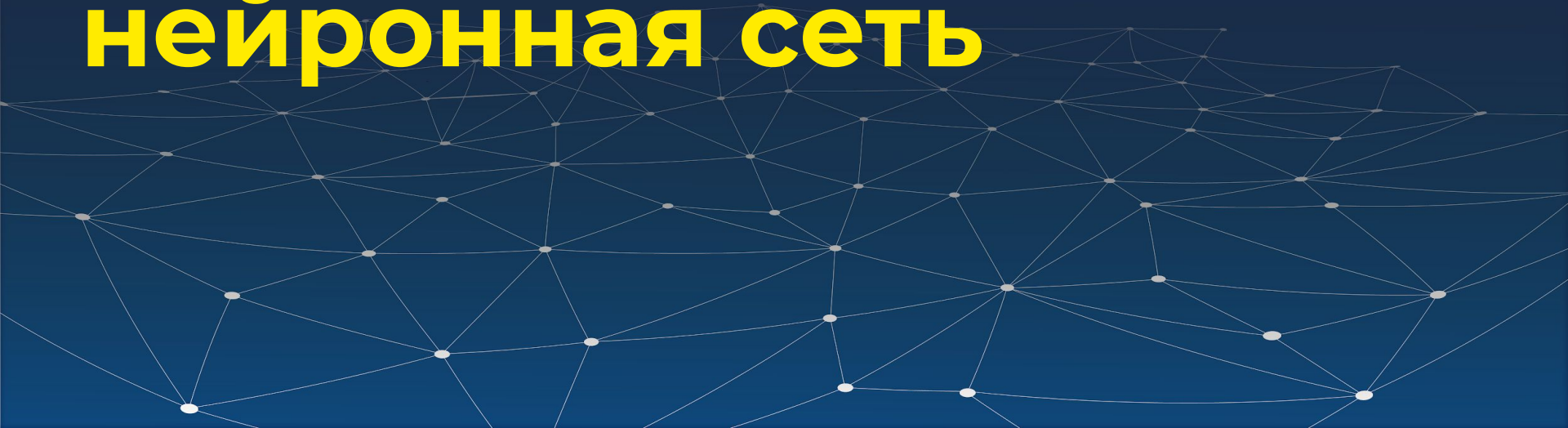


$$y(x) = \sigma(w^T x)$$



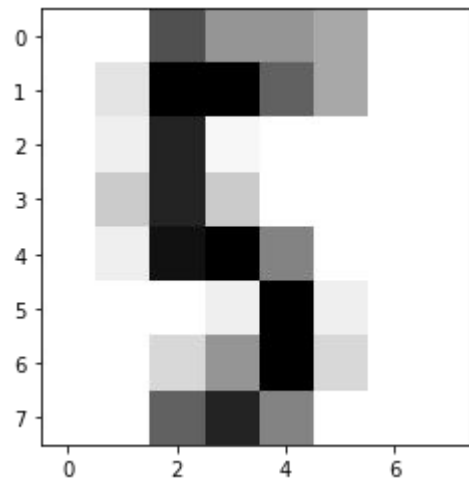
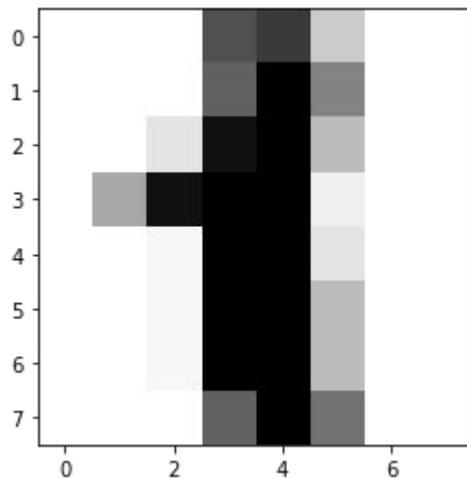
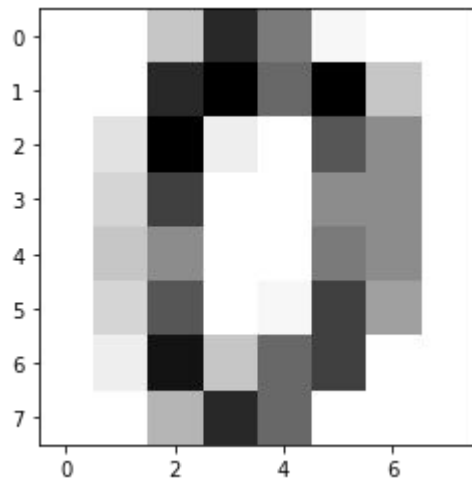
Как обучать?

# Полносвязная нейронная сеть



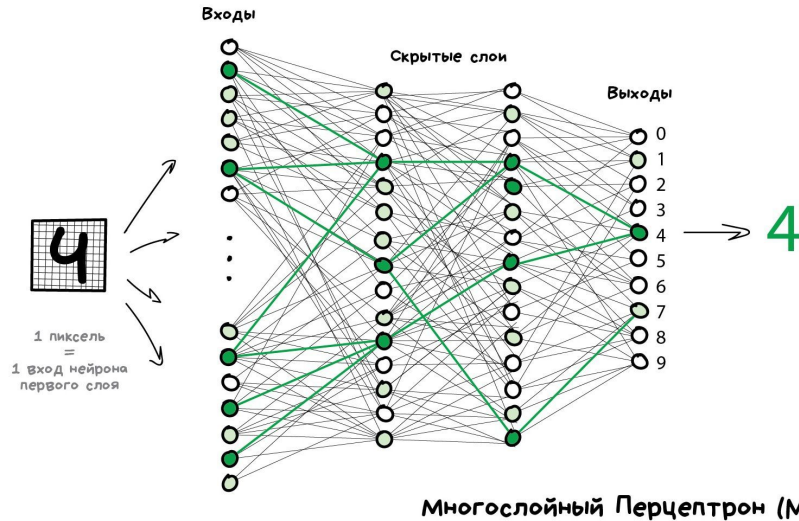
# Задача распознавания рукописных цифр

- Дано: чёрно-белое изображение 8x8
- Определить: какая из цифр нарисована
- Имеется: обучающая выборка размеченных изображений

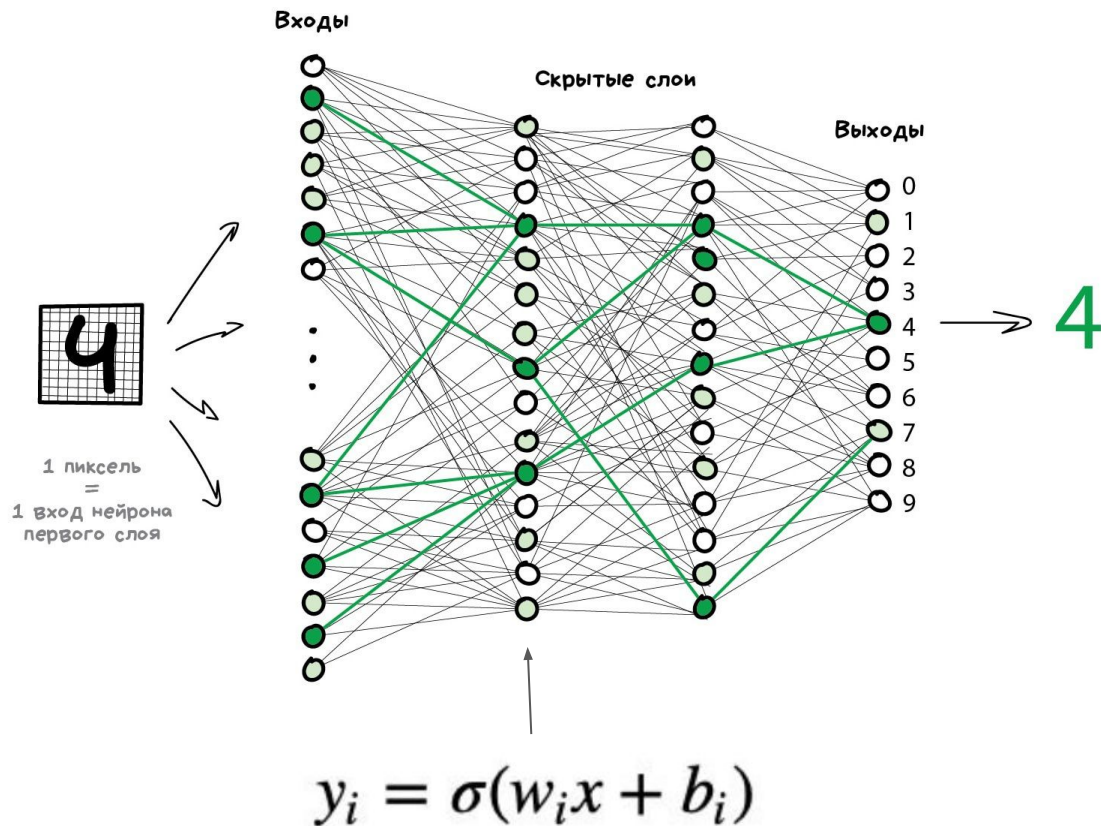


# Многослойный перцептрон

- Многослойный перцептрон — простейшая архитектура нейронной сети, каждый слой нейронов связан со всем нейронами с предыдущего слоя
- Десять выходных нейронов соответствуют классам изображений



# Многослойный перцептрон





# Последний слой в задаче классификации

- Решаем задачу классификации на 10 классов
- Предпоследний слой содержит 10 линейных нейронов — меры принадлежности объекта каждому классу
- Пусть  $y_0, y_1, \dots, y_9$  — выходы предпоследнего слоя


# Последний слой в задаче классификации

- Решаем задачу классификации на 10 классов
- Предпоследний слой содержит 10 линейных нейронов — меры принадлежности объекта каждому классу
- Пусть  $y_0, y_1, \dots, y_9$  — выходы предпоследнего слоя
- На последнем слое применяется softmax-преобразование:

$$y_0, y_1, \dots, y_9$$

# Последний слой в задаче классификации

- Решаем задачу классификации на 10 классов
- Предпоследний слой содержит 10 линейных нейронов — меры принадлежности объекта каждому классу
- Пусть  $y_0, y_1, \dots, y_9$  — выходы предпоследнего слоя
- На последнем слое применяется softmax-преобразование:

$$y_0, y_1, \dots, y_9$$

$$e^{y_0}, e^{y_1}, \dots, e^{y_9}$$

# Последний слой в задаче классификации

- Решаем задачу классификации на 10 классов
- Предпоследний слой содержит 10 линейных нейронов — меры принадлежности объекта каждому классу
- Пусть  $y_0, y_1, \dots, y_9$  — выходы предпоследнего слоя
- На последнем слое применяется softmax-преобразование:

$$\begin{array}{c} y_0, y_1, \dots, y_9 \\ \downarrow \\ e^{y_0}, e^{y_1}, \dots, e^{y_9} \\ \downarrow \\ \frac{e^{y_0}}{e^{y_0} + e^{y_1} + \dots + e^{y_9}}, \frac{e^{y_1}}{e^{y_0} + e^{y_1} + \dots + e^{y_9}}, \dots, \frac{e^{y_9}}{e^{y_0} + e^{y_1} + \dots + e^{y_9}} \end{array}$$

# Обучение нейронных сетей



# Оптимизация функции потерь

- Пусть  $(x, y)$  — элемент обучающей выборки,  $\theta$  — параметры нейросети

# Оптимизация функции потерь

- Пусть  $(x, y)$  — элемент обучающей выборки,  $\theta$  — параметры нейросети
- На последнем слое нейронной сети — вероятности классов

$$p = (p_0, p_1, \dots, p_9)^\top$$

# Оптимизация функции потерь

- Пусть  $(x, y)$  — элемент обучающей выборки,  $\theta$  — параметры нейросети
- На последнем слое нейронной сети — вероятности классов

$$p = (p_0, p_1, \dots, p_9)^\top$$

- Как и в случае логистической регрессии, оптимизируем  $\text{logloss}(y, p)$ :

$$\begin{aligned} \log p_y(x) &\rightarrow \max \\ -\log p_y(x) &\rightarrow \min \end{aligned}$$



# Оптимизация функции потерь

- Пусть  $(x, y)$  — элемент обучающей выборки,  $\theta$  — параметры нейросети
- На последнем слое нейронной сети — вероятности классов

$$p = (p_0, p_1, \dots, p_9)^\top$$

- Как и в случае логистической регрессии, оптимизируем  $\text{logloss}(y, p)$ :

$$\begin{aligned}\log p_y(x) &\rightarrow \max \\ -\log p_y(x) &\rightarrow \min\end{aligned}$$

- Итоговая задача оптимизации:

$$\frac{1}{\ell} \sum_{(x,y) \in X_{train}} \text{logloss}(y, p(x, \theta)) \rightarrow \min_{\theta}$$

# Оптимизация функции потерь

Для бинарной классификации:

$$\text{logloss} = -\frac{1}{n} \sum_{i=0}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Для многоклассовой классификации:

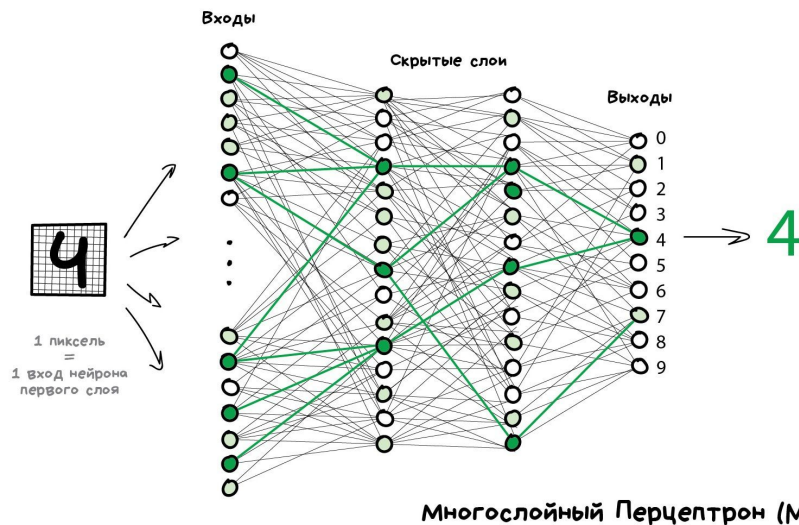
$$\text{cross-entropy} = - \sum_x p(x) \log q(x)$$

# Оптимизация в общем случае

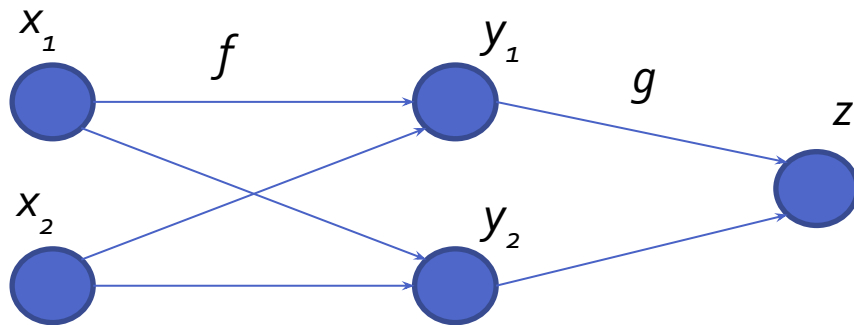
Можно оптимизировать и другие функции потерь. Например, MSE в случае задачи регрессии

# Стохастический градиентный спуск

- Выбираем пример из обучающей выборки
- Вычисляем производную функции потерь по всем весам нейросети
- Обновляем веса в направлении антиградиента



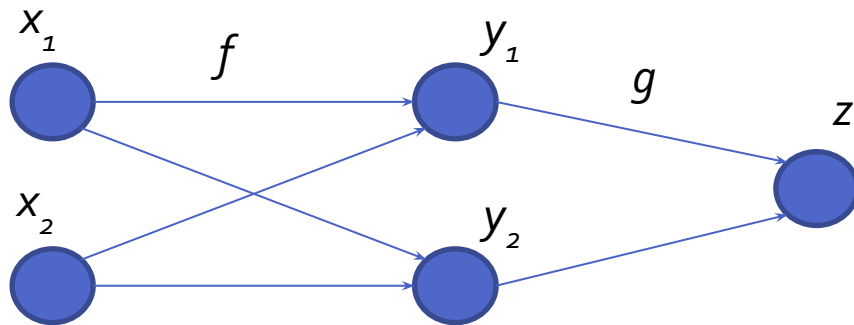
# Напоминание: правило дифференцирования



$$\frac{dz}{dy} = \begin{bmatrix} \frac{\partial z}{\partial y_1}(y_1, y_2) \\ \frac{\partial z}{\partial y_2}(y_1, y_2) \end{bmatrix}$$

$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_1} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_1}$$

# Напоминание: правило дифференцирования



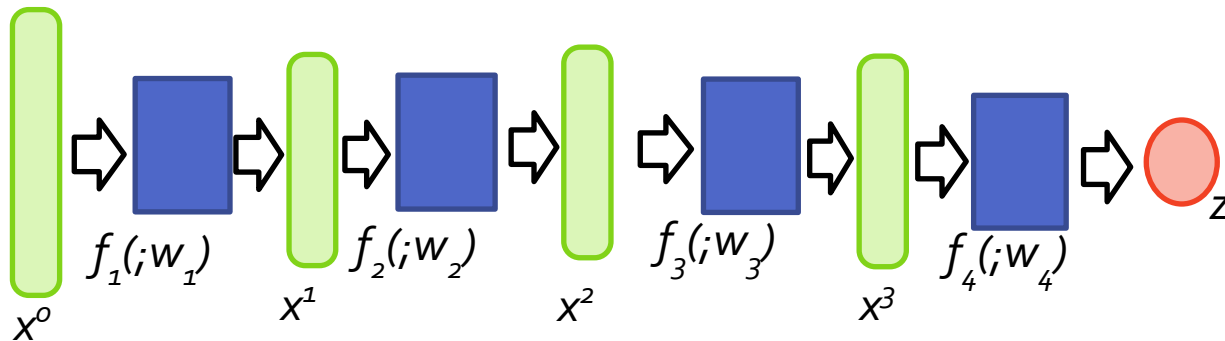
$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_1} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_1}$$

$$\frac{\partial z}{\partial x_2} = \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_2} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_2}$$

$$\frac{dz}{dx} = \begin{bmatrix} \frac{\partial z}{\partial x_1} \\ \frac{\partial z}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial z}{\partial y_1} \\ \frac{\partial z}{\partial y_2} \end{bmatrix}$$

$$\frac{dz}{dx} = \left( \frac{dy}{dx} \right)^T \frac{dz}{dy}$$

# Вычисление глубоких производных



$$z = f_4(f_3(f_2(f_1(x; w_1); w_2); w_3); w_4)$$

# Вычисление глубоких производных

$\frac{dz}{dx^3}, \frac{dz}{dw_4}$  МОЖНО ВЫЧИСЛИТЬ

$$\frac{dz}{dw_3} = \frac{dx^3{}^T}{dw_3} \cdot \frac{dz}{dx^3}$$

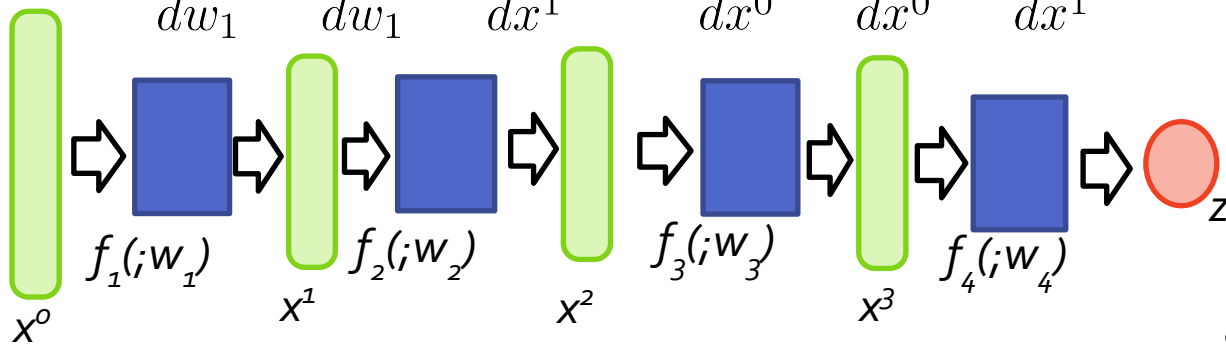
$$\frac{dz}{dx^2} = \frac{dx^3{}^T}{dx^2} \cdot \frac{dz}{dx^3}$$

$$\frac{dz}{dw_2} = \frac{dx^2{}^T}{dw_2} \cdot \frac{dz}{dx^2}$$

$$\frac{dz}{dx^1} = \frac{dx^2{}^T}{dx^1} \cdot \frac{dz}{dx^2}$$

$$\frac{dz}{dw_1} = \frac{dx^1{}^T}{dw_1} \cdot \frac{dz}{dx^1}$$

$$\frac{dz}{dx^0} = \frac{dx^1{}^T}{dx^0} \cdot \frac{dz}{dx^1}$$





# Слой нейронной сети

Чтобы определить слой, необходимо задать:

- forward performance:  $y = f(x; w)$
- backward performance:  $z(x) = z(f(x; w))$

В случае, если слой реализует простую функцию, то для backward пользуемся правилом

$$\frac{dz}{dx} = \left( \frac{dy}{dx} \right)^T \frac{dz}{dy}$$

# Пример: полносвязный (линейный) слой

```
In [ ]: class Linear(Module):
        """
        A module which applies a linear transformation
        A common name is fully-connected layer

        The module should work with 2D input of shape (n_samples, n_feature).
        """

        def forward(self, input):
            self.output = np.dot(input, self.W.T) + self.b
            return self.output

        def updateGradInput(self, input, gradOutput):
            self.gradInput = np.dot(gradOutput, self.W)

            self.gradW = np.dot(gradOutput.T, input)
            self.gradb = np.sum(gradOutput, axis=0)

            return self.gradInput
```

# Фреймворки Deep Learning



# Фреймворки Deep Learning



theano



# Фреймворки Deep Learning



theano



# The End

