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

1943. Маккалох, Питс. Модель искусственного нейрона

1957. Персептрон Розенблатта
1960. Теорема Новикова

1986. Руммельхарт, Хинтон, Уильямс. Алгоритм Back propagation

1982, Хопфилд.
1997, Хохрейтер, Шмидхубер.
Рекуррентные нейронные сети

1998. Лекун. Свёрточные нейронные сети

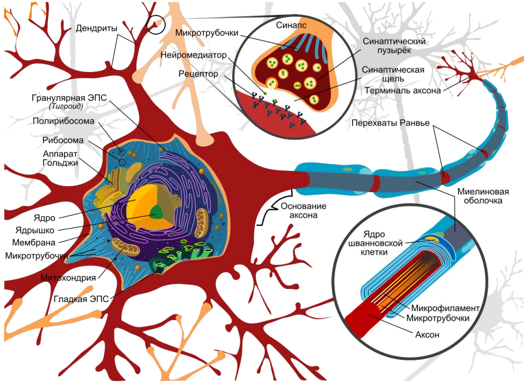
~2012 Революция глубокого обучения

...

Модель нейрона

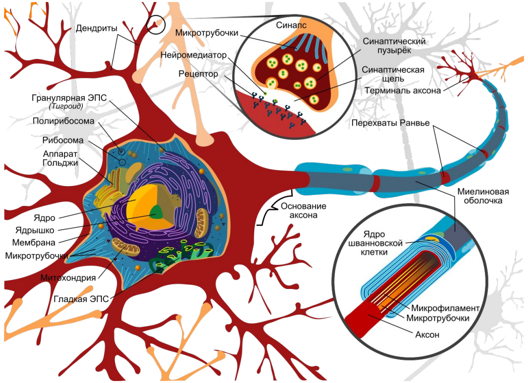
Модель человеческого нейрона

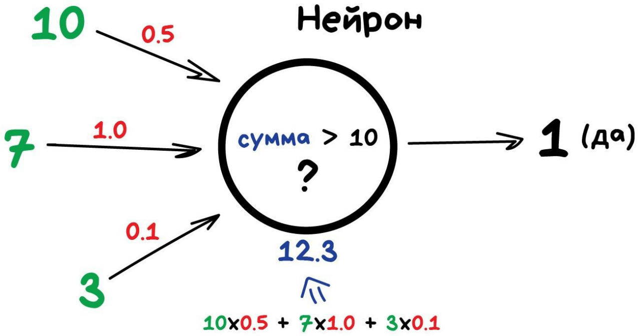
- Внутри мозга информация передаётся по связям между *нейронами* химическим веществом — нейротрансмиттером
- Нейротрансмиттер поступает в тело нейрона по специальным отросткам — *дендритам*
- При накоплении определённого количества нейротрансмиттера нейрон “пробивает” и вещество передаётся по *аксону* другим нейронам



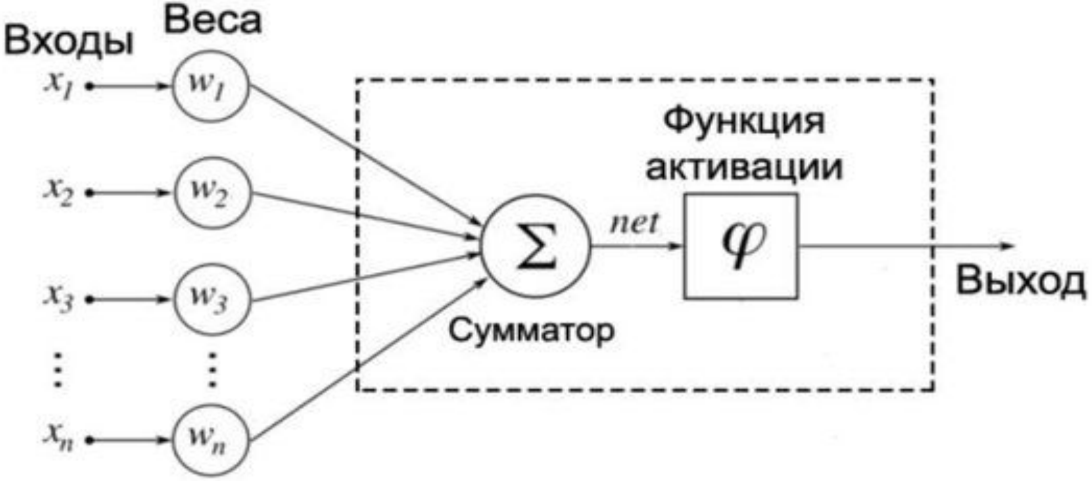
Модель человеческого нейрона

- 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



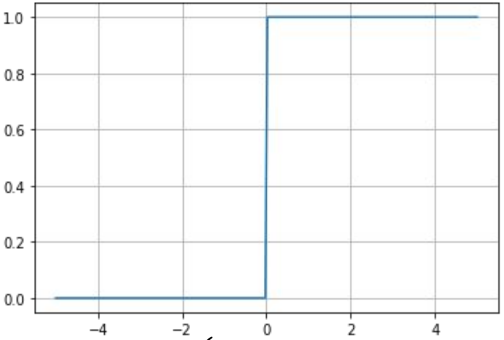


Модель нейрона

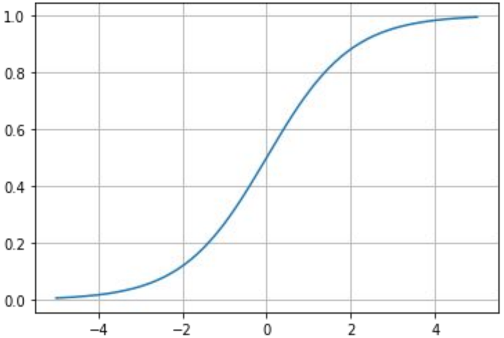


Нейрон определяет функцию $y_w(x) = \varphi(w^\top x\,(+b))$

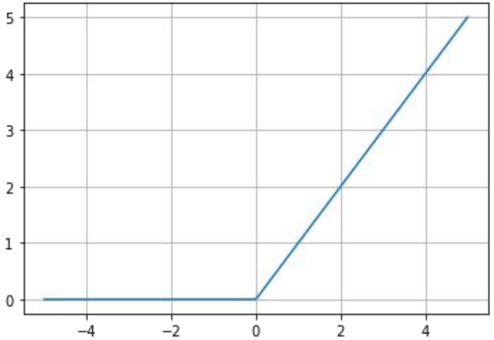
Функции активации



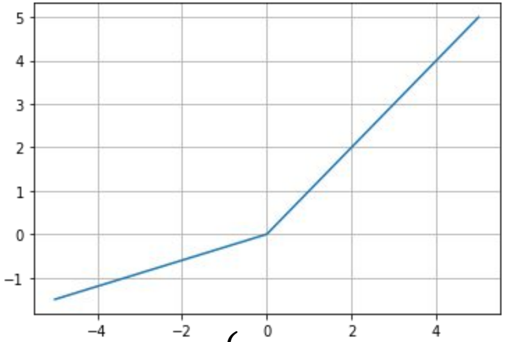
$$\varphi(t) = \begin{cases} 1, & t \geqslant 0 \\ 0, & otherwise \end{cases}$$



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

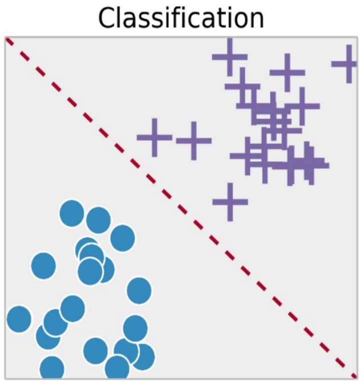


$$\varphi(t) = \max(t, 0)$$

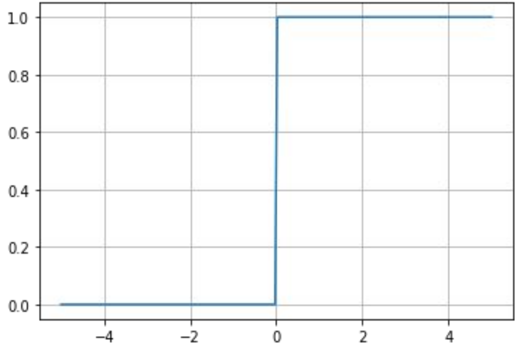


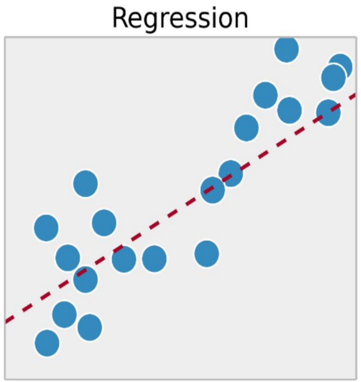
$$\varphi(t) = \begin{cases} \alpha t, & t \geqslant 0 \\ \beta t, & t < 0 \end{cases}$$

Линейные алгоритмы



$y(x) = \mathrm{sign}(<w, x>)$

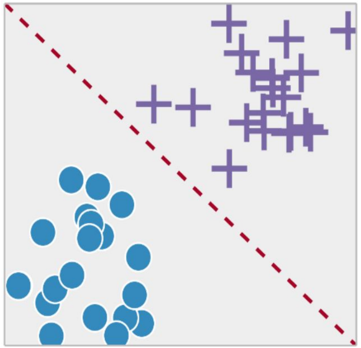




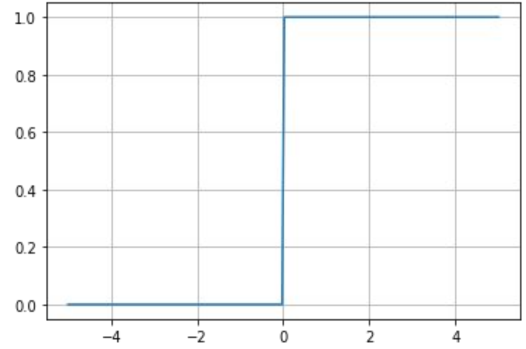
$y(x) = <w, x>$

Линейные алгоритмы

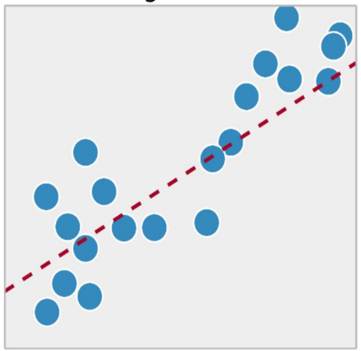
Classification



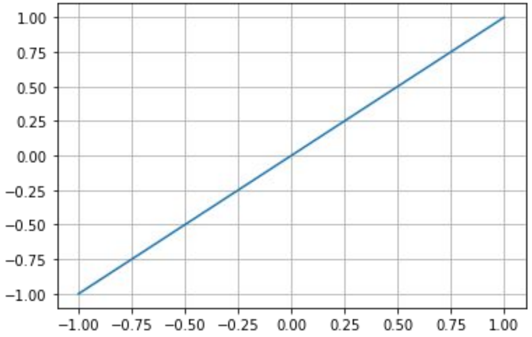
$y(x) = \text{sign}(\langle w, x \rangle)$



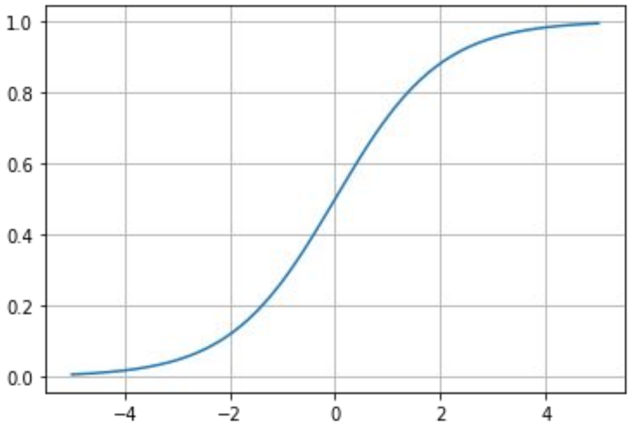
Regression

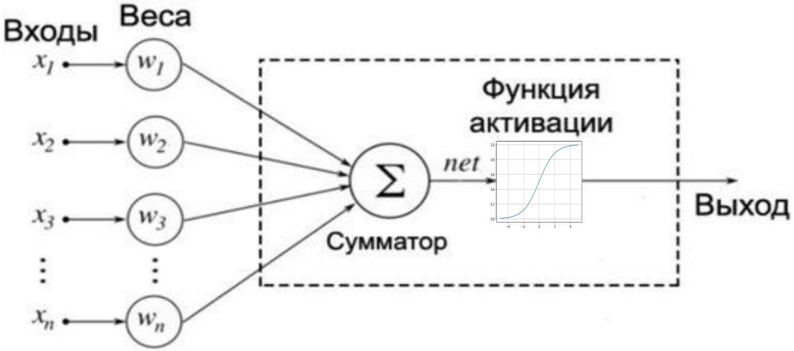


$y(x) = \langle w, x \rangle$



Как изобразить логистическую регрессию?





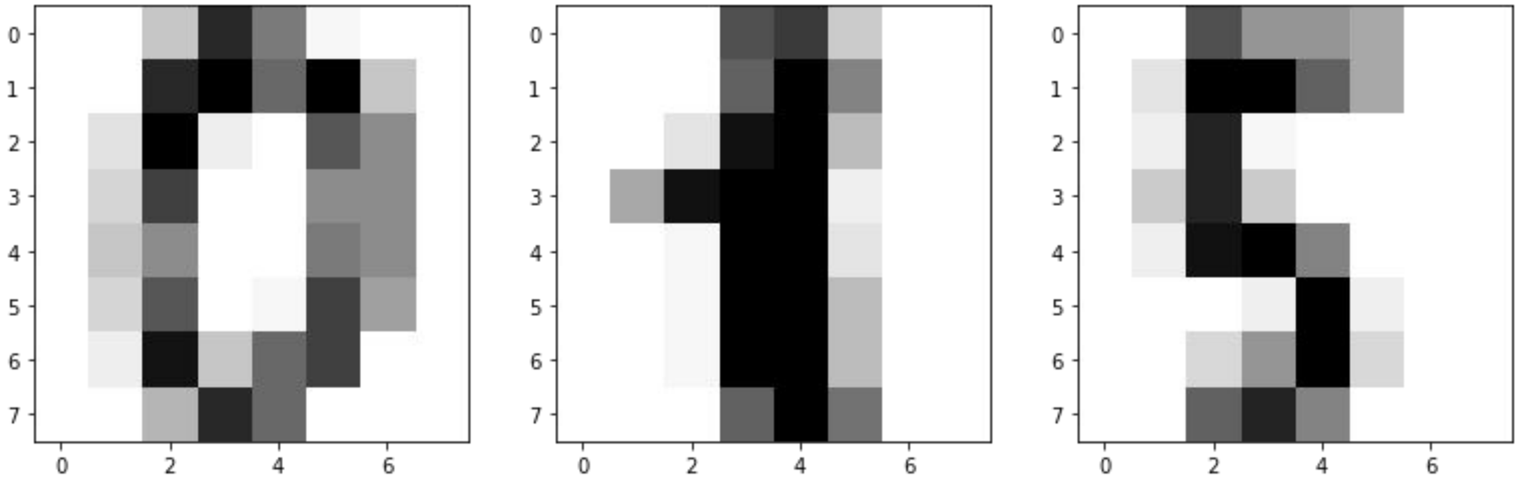
$$y(x) = \sigma(w^\top x)$$

Как обучать?

Полносвязная нейронная сеть

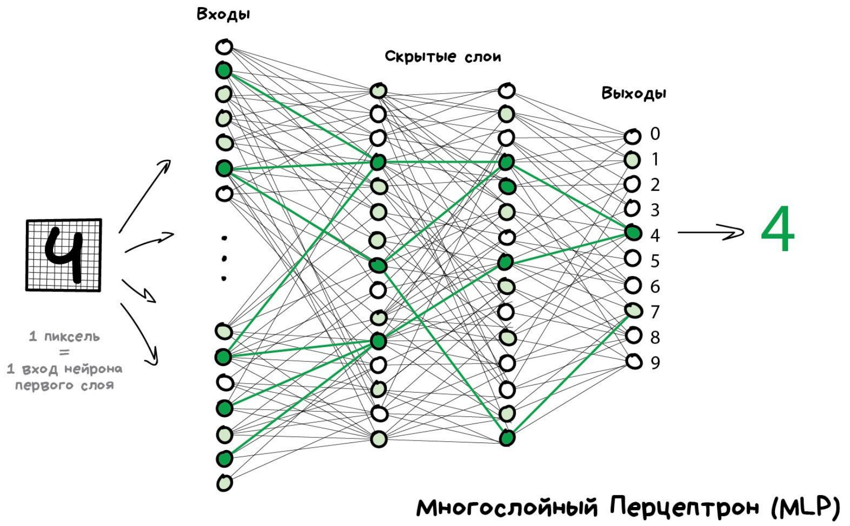
Задача распознавания рукописных цифр

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

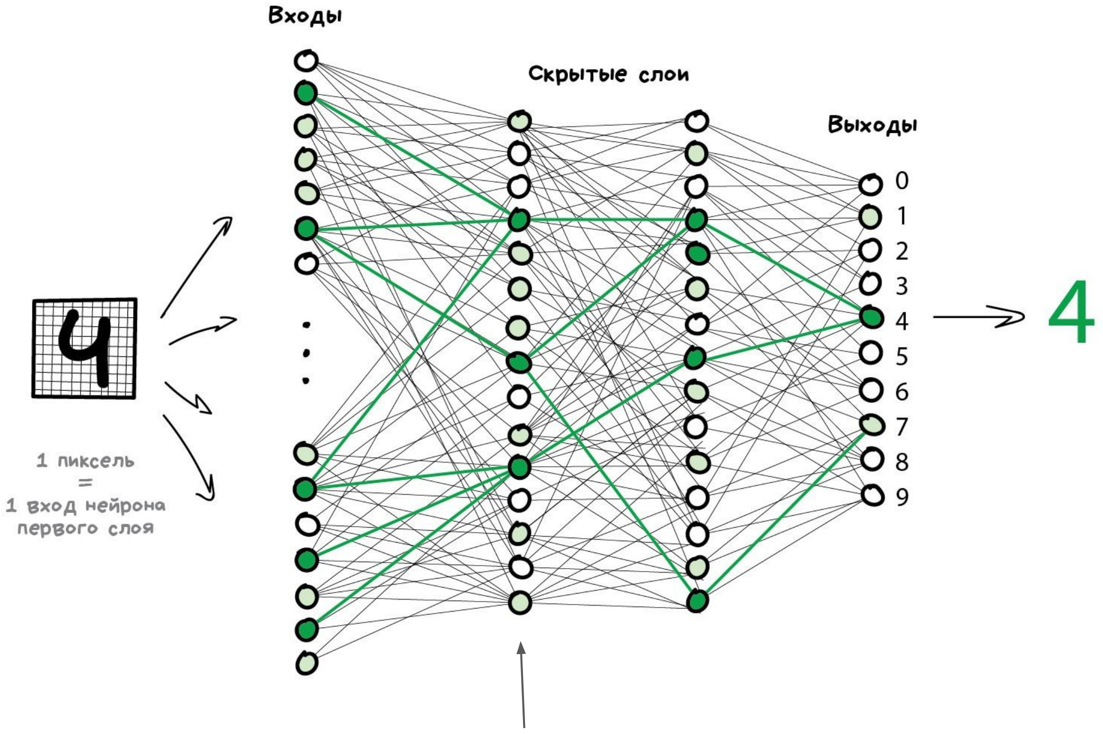


Многослойный перцептрон

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



Многослойный перцептрон



$$y_i = \sigma(w_i x + b_i)$$

Последний слой в задаче классификации

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

$$y_0, y_1, \ldots, y_9$$

Последний слой в задаче классификации

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

$$y_0, y_1, \ldots, y_9$$

$$\Downarrow$$

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

Последний слой в задаче классификации

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

$$y_0, y_1, \ldots, y_9$$

$$\Downarrow$$

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

$$\Downarrow$$

$$\frac{e^{y_0}}{e^{y_0}+e^{y_1}+\ldots+e^{y_9}}, \frac{e^{y_1}}{e^{y_0}+e^{y_1}+\ldots+e^{y_9}}, \ldots, \frac{e^{y_9}}{e^{y_0}+e^{y_1}+\ldots+e^{y_9}}$$

Обучение нейронных сетей

Оптимизация функции потерь

- Пусть (x, y) — элемент обучающей выборки, θ — параметры нейросети

Оптимизация функции потерь

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

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

Оптимизация функции потерь

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

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

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

$$\log p_y(x) \to \max$$

$$-\log p_y(x) \to \min$$

Оптимизация функции потерь

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

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

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

$$\log p_y(x) \to \max$$
$$-\log p_y(x) \to \min$$

- Итоговая задача оптимизации:

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

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:

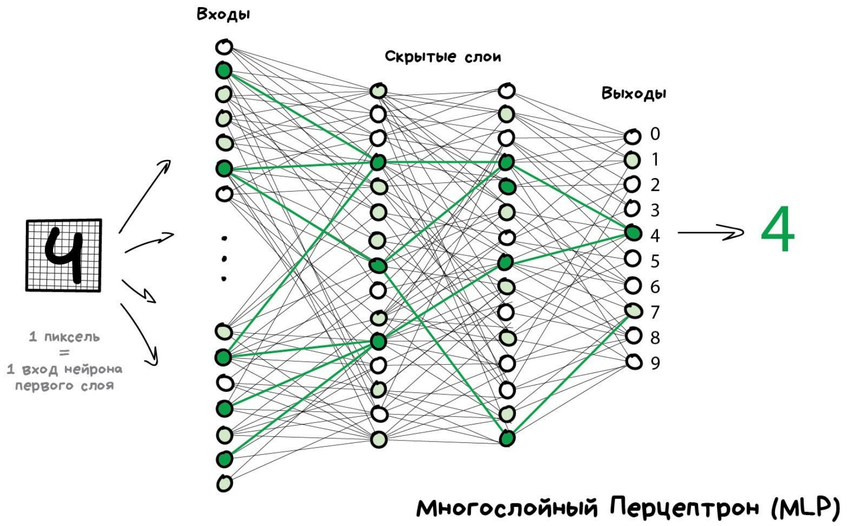
$$\text{cross-entropy} = -\sum_{x} p(x)\,\log q(x)$$

Оптимизация в общем случае

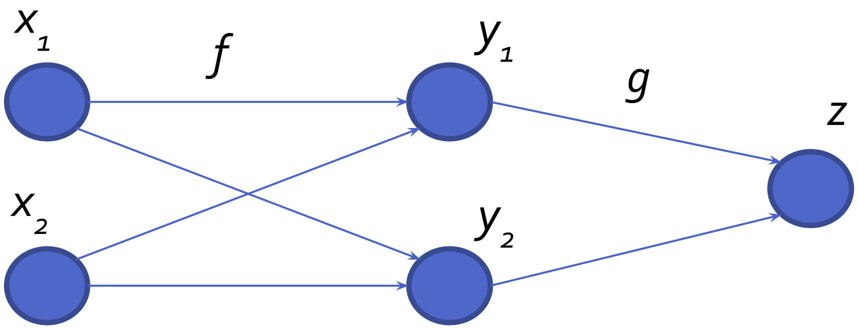
Можно оптимизировать и другие функции потерь. Например, MSE в случае задачи регрессии

Стохастический градиентный спуск

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



Напоминание: правило дифференцирования

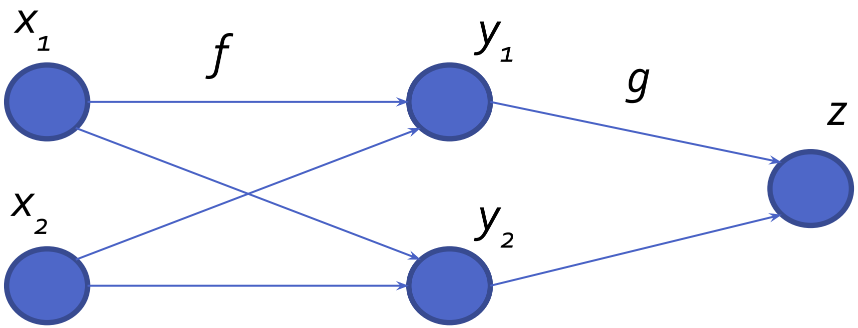


$$\frac{dz}{dy} = \begin{bmatrix} \frac{\partial z}{\partial y_1}(y_1, y_2) \\ \\ \frac{\partial z}{\partial y_2}(y_1, y_2) \end{bmatrix}$$

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

слайд В. Лемпицкого

Напоминание: правило дифференцирования



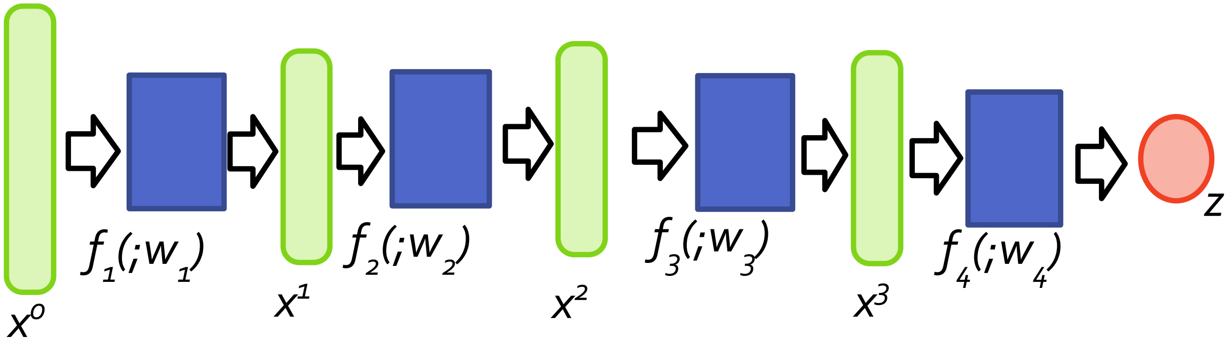
$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_1} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_1}$$

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

$$\frac{dz}{dx} = \begin{bmatrix} \frac{\partial z}{\partial x_1} \\ \\ \frac{\partial z}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} \\ \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial z}{\partial y_1} \\ \\ \frac{\partial z}{\partial y_2} \end{bmatrix} \qquad \frac{dz}{dx} = \left(\frac{dy}{dx}\right)^T \frac{dz}{dy}$$

слайд В. Лемпицкого

Вычисление глубоких производных



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

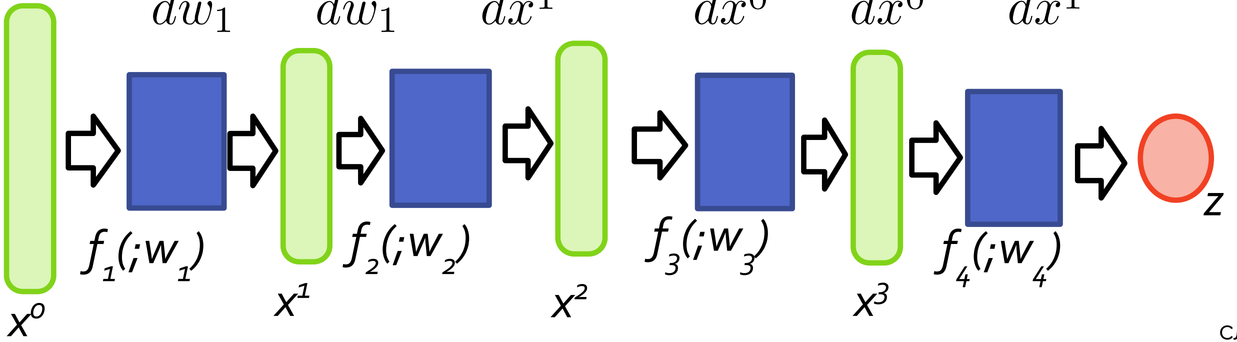
Вычисление глубоких производных

$\frac{dz}{dx^3}, \frac{dz}{dw_4}$ можно вычислить

$$\frac{dz}{dw_3} = \frac{dx^3}{dw_3}^T \cdot \frac{dz}{dx^3} \qquad \frac{dz}{dx^2} = \frac{dx^3}{dx^2}^T \cdot \frac{dz}{dx^3}$$

$$\frac{dz}{dw_2} = \frac{dx^2}{dw_2}^T \cdot \frac{dz}{dx^2} \qquad \frac{dz}{dx^1} = \frac{dx^2}{dx^1}^T \cdot \frac{dz}{dx^2}$$

$$\frac{dz}{dw_1} = \frac{dx^1}{dw_1}^T \cdot \frac{dz}{dx^1} \qquad \frac{dz}{dx^0} = \frac{dx^1}{dx^0}^T \cdot \frac{dz}{dx^1}$$



слайд В. Лемпицкого

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

mxnet

PyTorch

Фреймворки Deep Learning

theano

mxnet

PyTorch

The End