

Цифровая обработка изображений

4. Введение в нейронные сети

План занятия

- 4.1 Принципы работы нейронной сети
 - перцептрон
 - логические операции
- 4.2 Полносвязная нейронная сеть
 - структура полносвязной сети
 - функции активации и нелинейность
 - матричные операции
 - граф вычислений

План занятия

- 4.3 Обучение

- функции потерь - логит, софтмакс
- градиентный спуск и методы оптимизации
- обратное распространение градиента и граф вычислений, автоматическое дифференцирование
- регуляризация
- инициализация весов

Примеры применения нейронных сетей

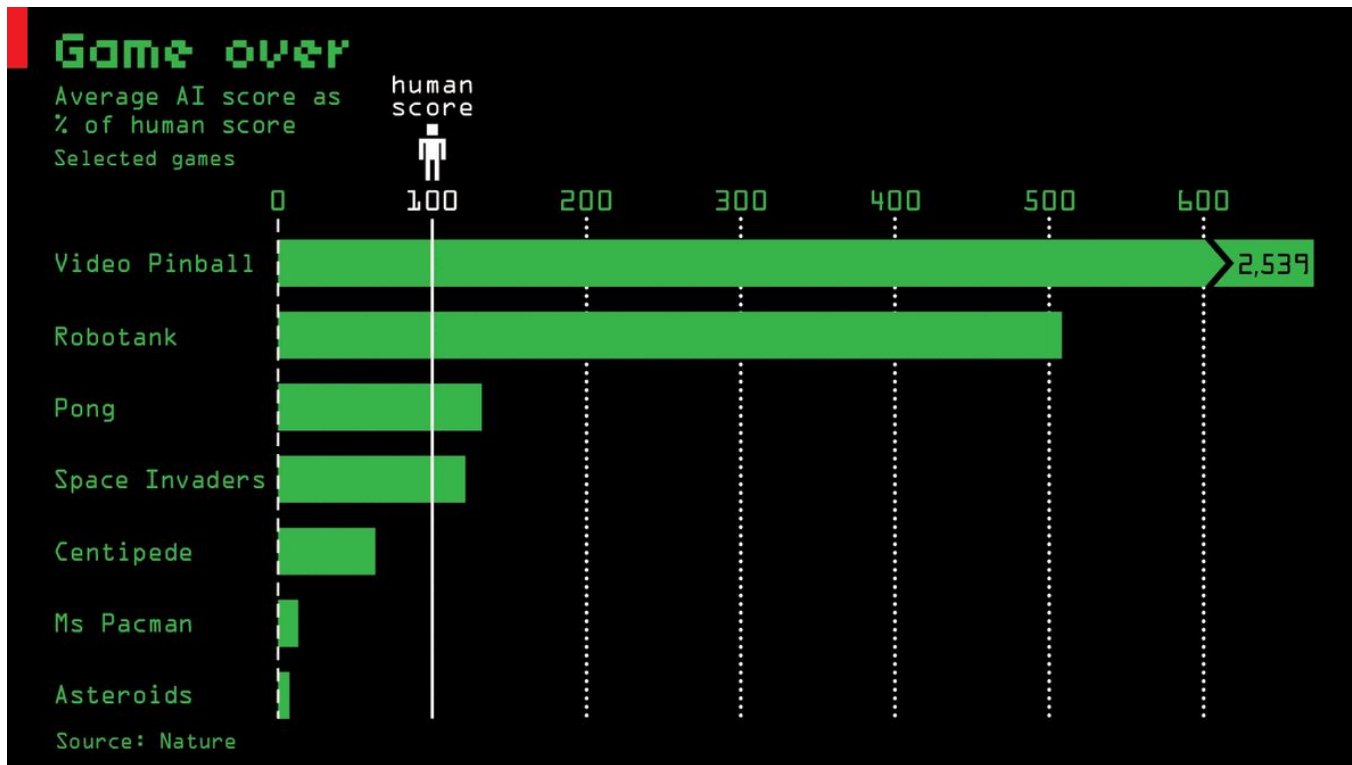
ALPHAGO



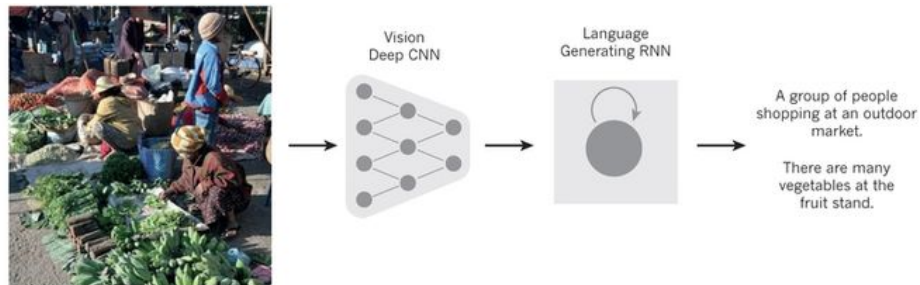
Консольные игры ATARI



Консольные игры ATARI



Автоматическая аннотация изображений



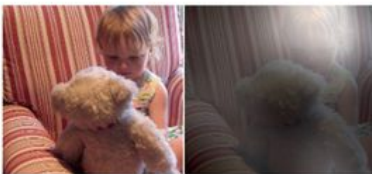
A woman is throwing a **frisbee** in a park.



A **dog** is standing on a hardwood floor.



A **stop** sign is on a road with a mountain in the background



A little **girl** sitting on a bed with a teddy bear.

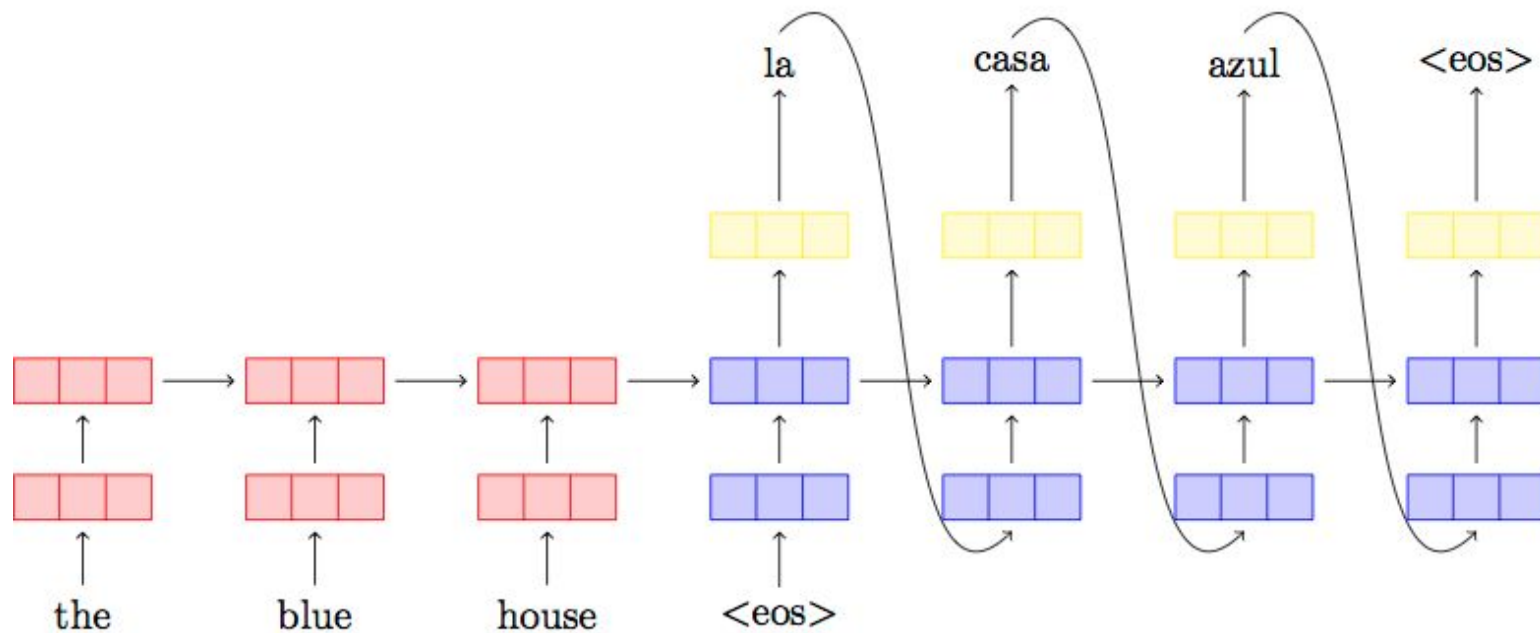


A group of **people** sitting on a boat in the water.



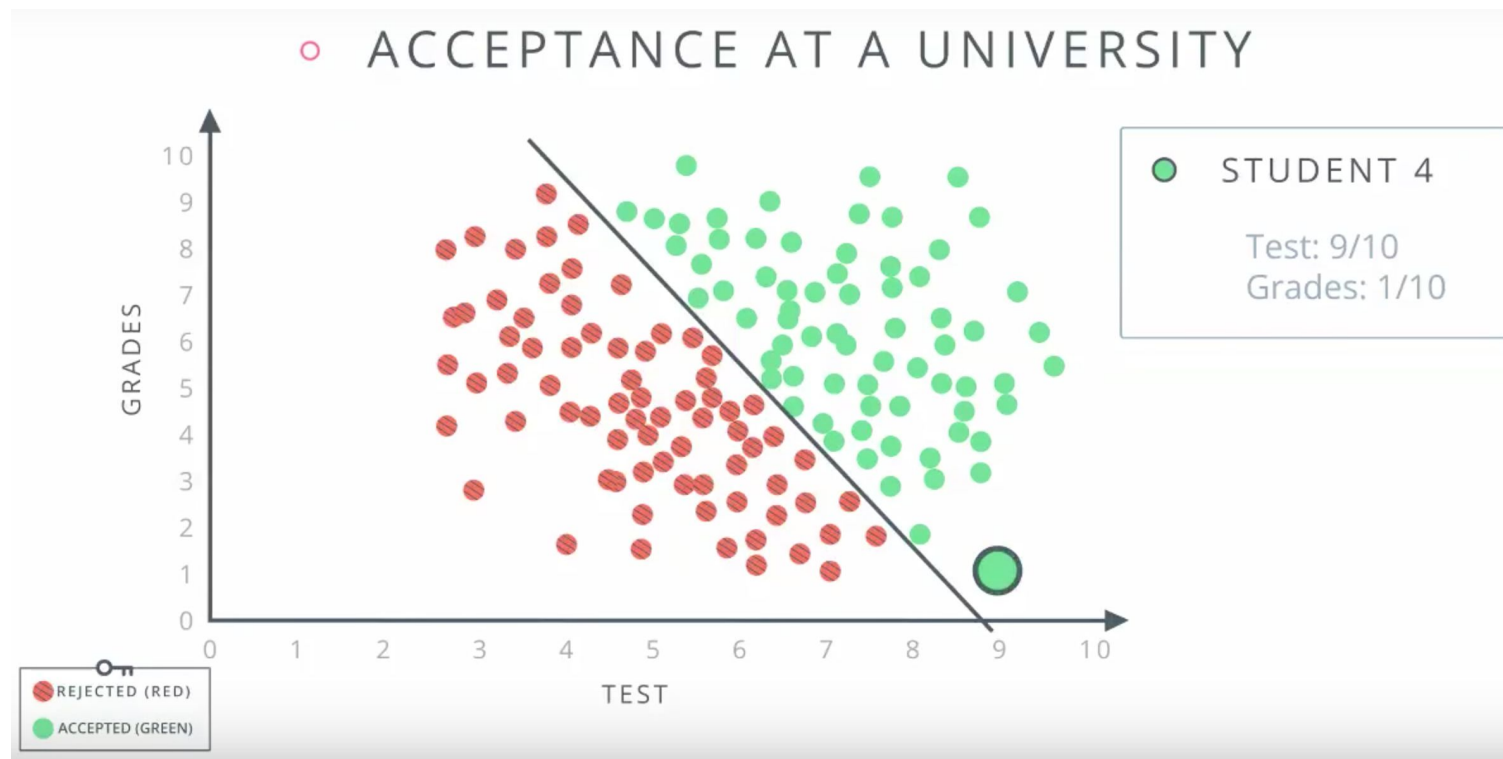
A giraffe standing in a forest with **trees** in the background.

Машинный перевод

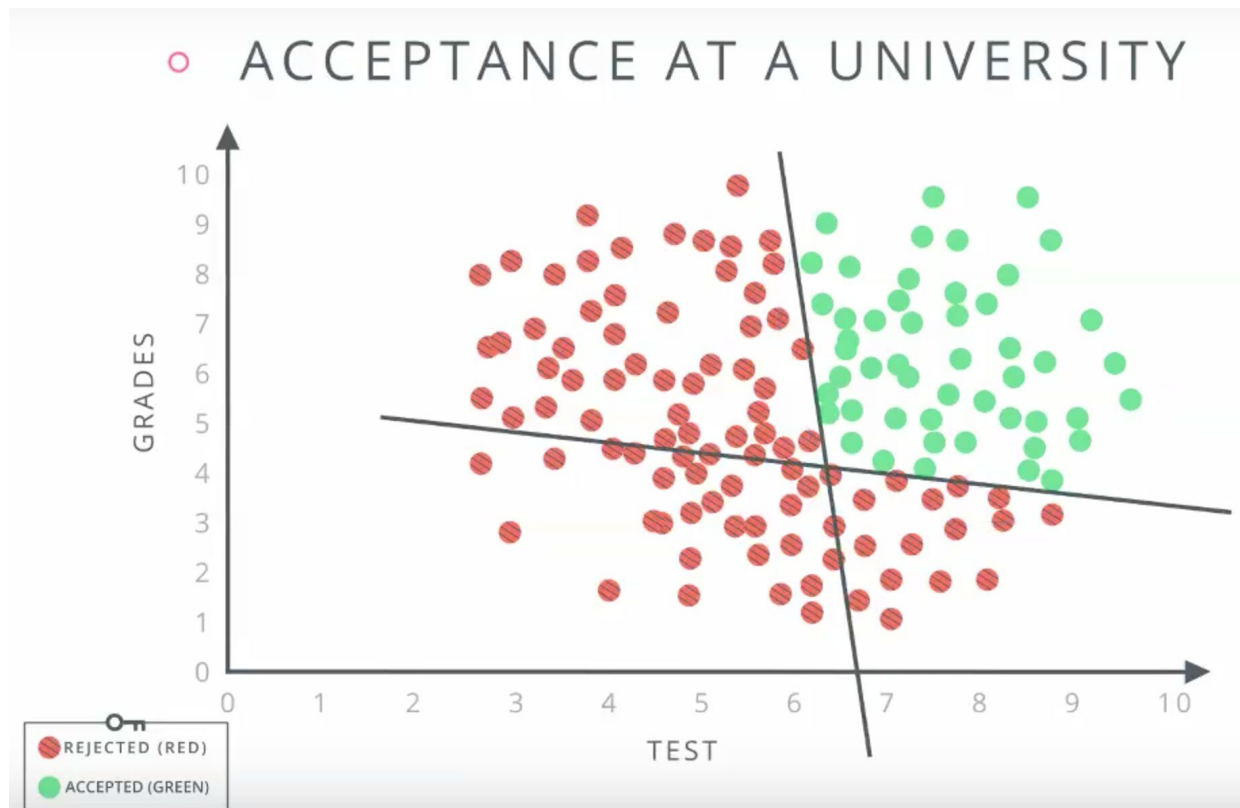


Принцип работы нейронной сети

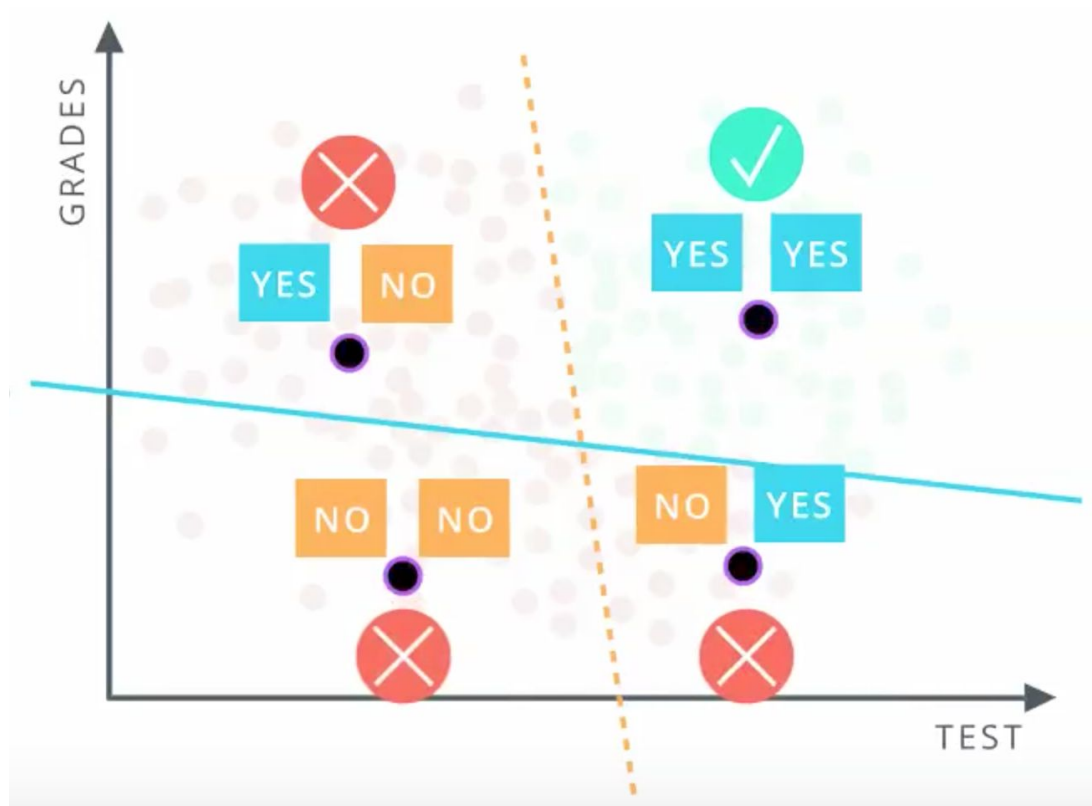
Задача. Предсказать поступление в университет по результатам теста и оценкам



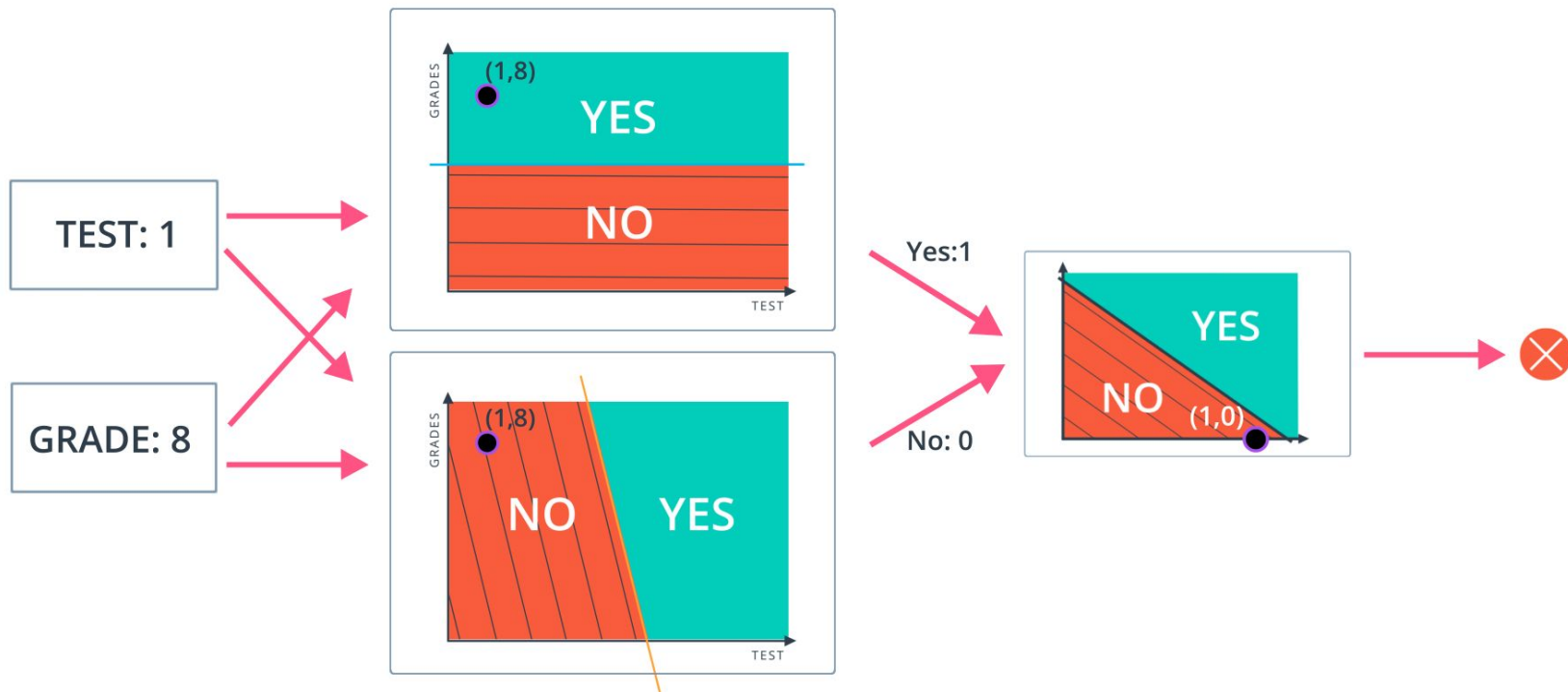
Линейная регрессия



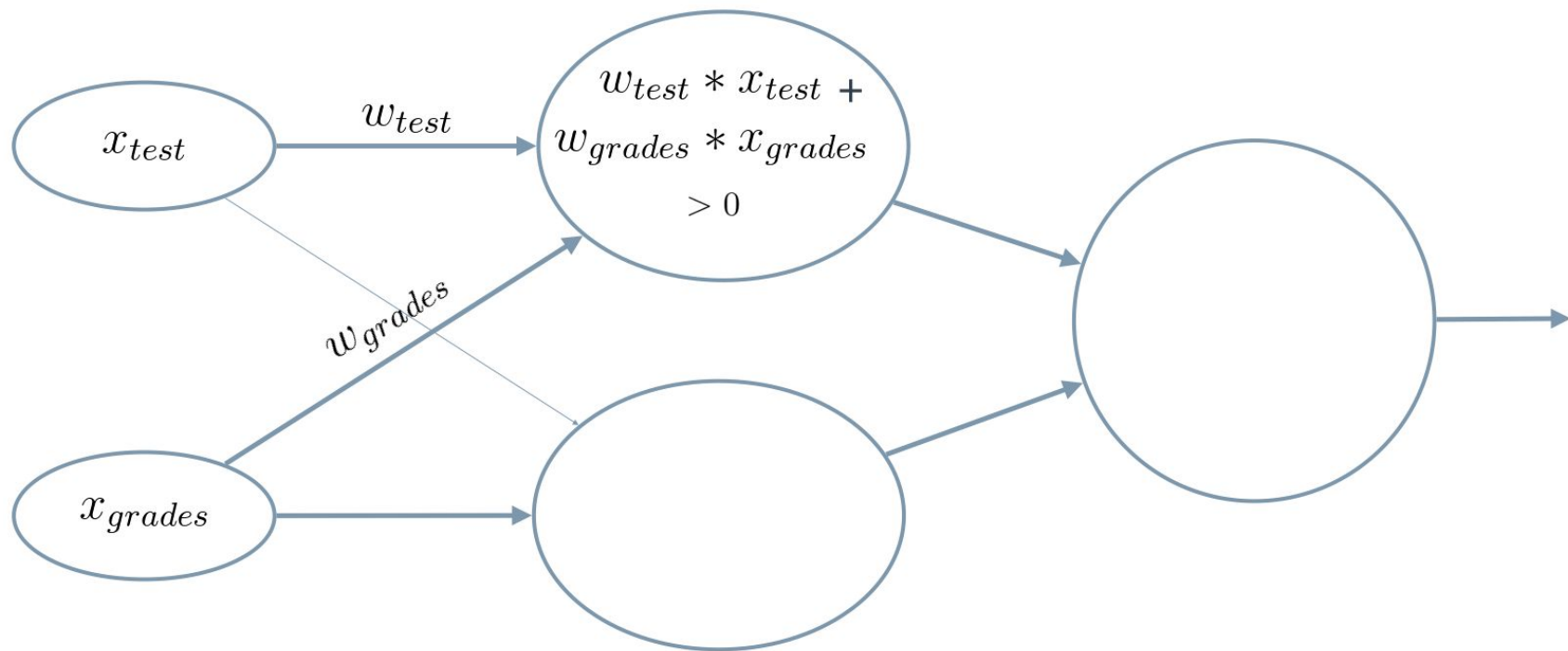
Объединяем линейные регрессии



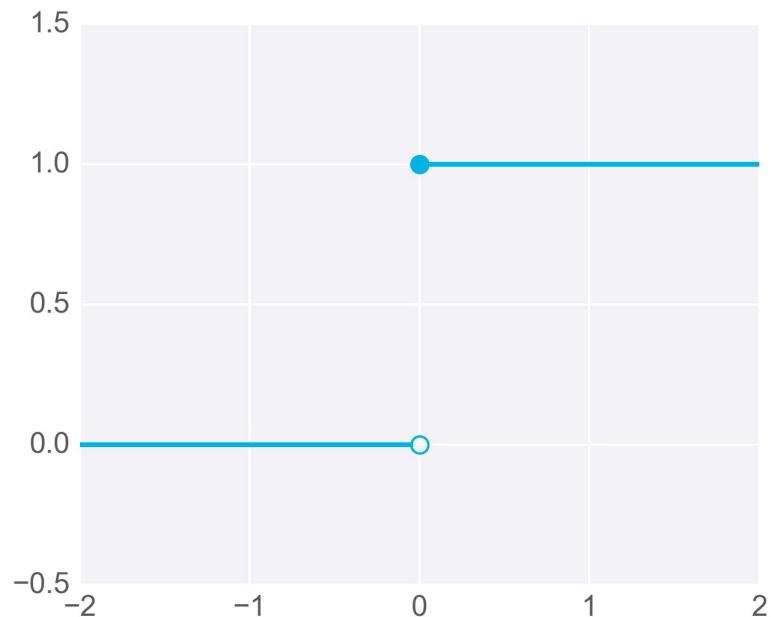
Объединяем линейные регрессии



Объединяем линейные регрессии



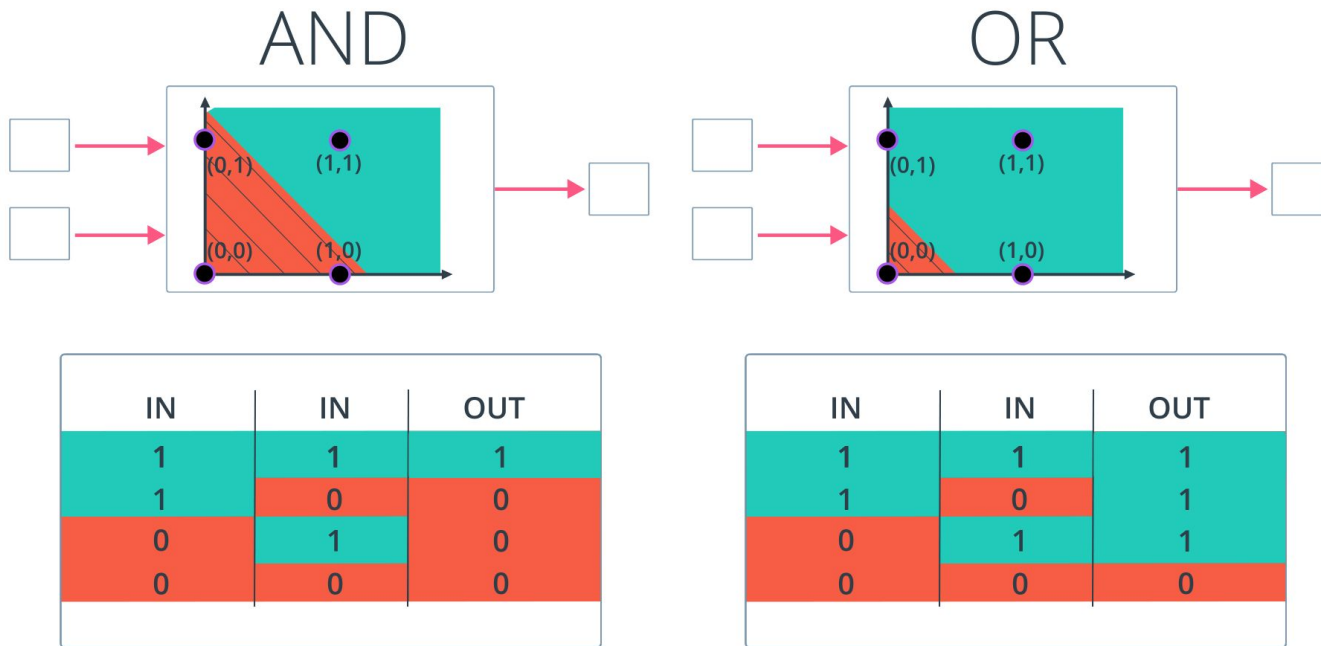
Функция активации



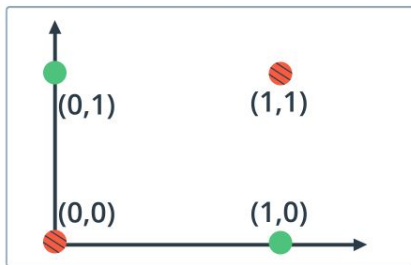
$$f(x_1, x_2, \dots, x_m) = \begin{cases} 0 & \text{if } b + \sum w_i \cdot x_i < 0 \\ 1 & \text{if } b + \sum w_i \cdot x_i \geq 0 \end{cases}$$

x - значения признаков
w - веса регрессии
b - свободный член

Логические операции. AND, OR

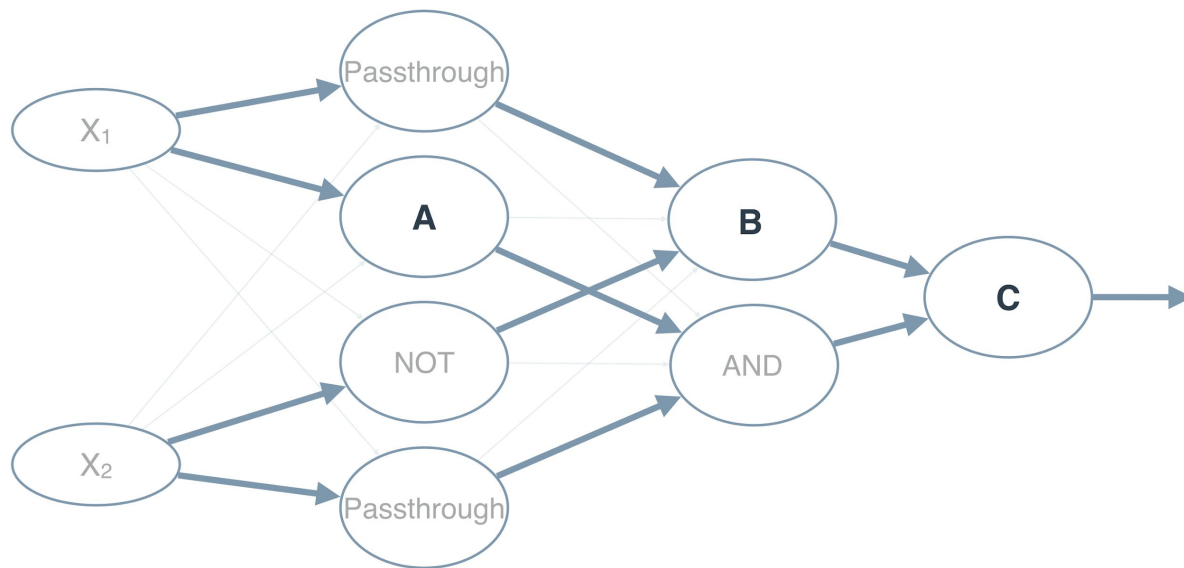


Логические операции. XOR



IN	IN	OUT
1	1	0
1	0	1
0	1	1
0	0	0

Логические операции. XOR



A		B			C		
IN	OUT	IN	IN	OUT	IN	IN	OUT
1	0	1	1	1	1	1	1
1	1	1	0	0	1	0	1
0	1	0	1	0	0	1	1
0	0	0	0	0	0	0	0

Принцип работы нейронной сети

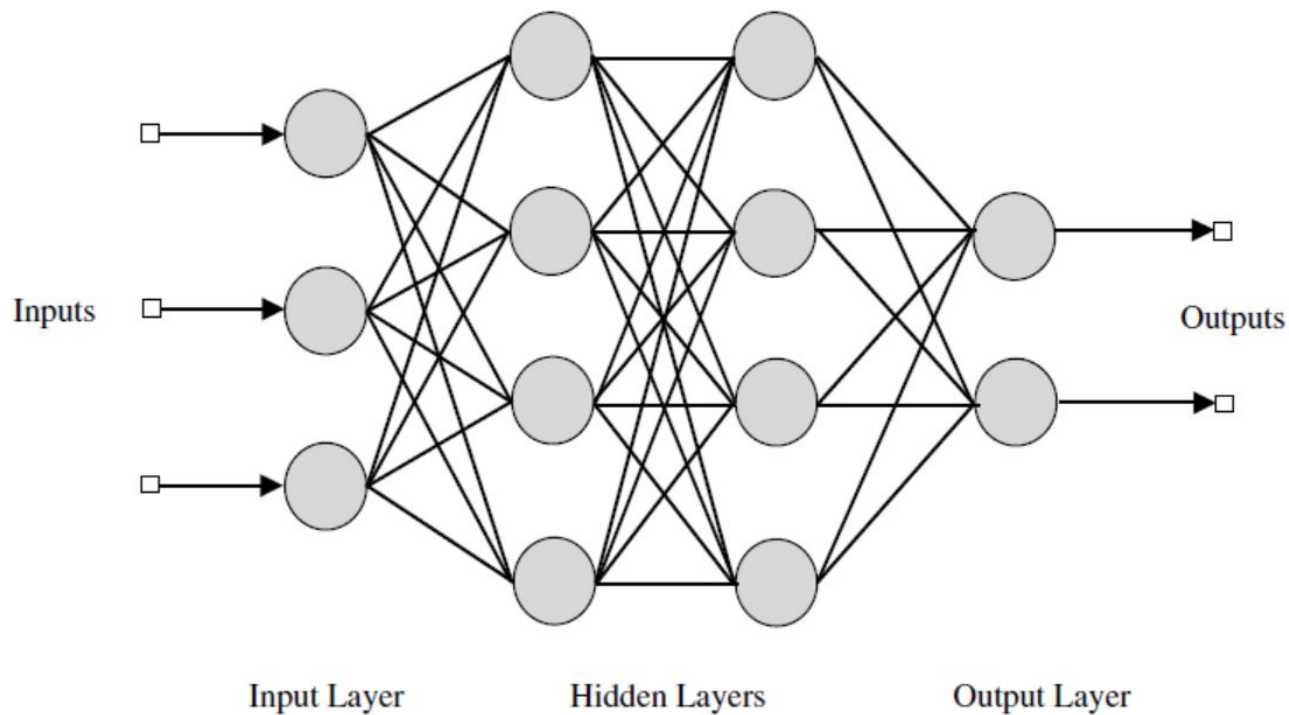
- перцептрон может иметь один или несколько входов
- значение на каждом входе взвешивается соответствующим весом
- выходом перцептрона является линейная комбинация входных значений и соответствующих весов
- к выходу перцептрона применяется функция активации
- изменяя веса перцептрона мы изменяем его функциональность

Полносвязная нейронная сеть

Структура полносвязной сети

- содержит один входной слой, один выходной
- и один или несколько внутренних слоев
- каждый нейрон предыдущего слоя связан с нейроном последующего слоя

Структура полносвязной сети

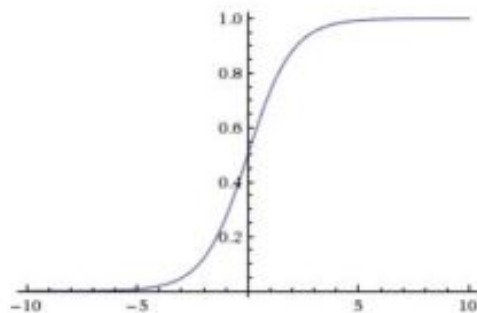


Функции активации

- в случае линейной функции активации нейронная сеть вырождается в линейное преобразование
- линейную функцию активации на практике используют только на выходном слое в задачах регрессии
- для внутренних слоев в качестве функции активации как правило используют `relu` или `tanh`

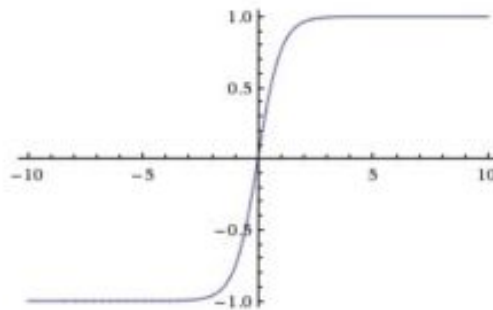
Функции активации

$$\sigma(x) = 1 / (1 + \exp(-x))$$



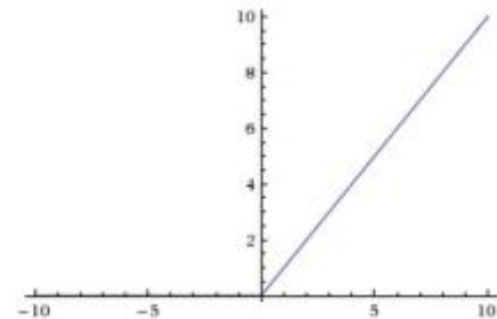
Sigmoid

$$\tanh(x) = 2\sigma(2x) - 1$$



tanh

$$f(x) = \max(0, x)$$



ReLU

Функции активации

- **sigmoid** как правило используется для выходного слоя
- для внутренних слоев обычно используют **tanh** или **relu**
- выбор этих функций в качестве активации для внутренних слоев связан с особенностью процесса обучения

Матричные операции

Input Layer

bias X1 X2

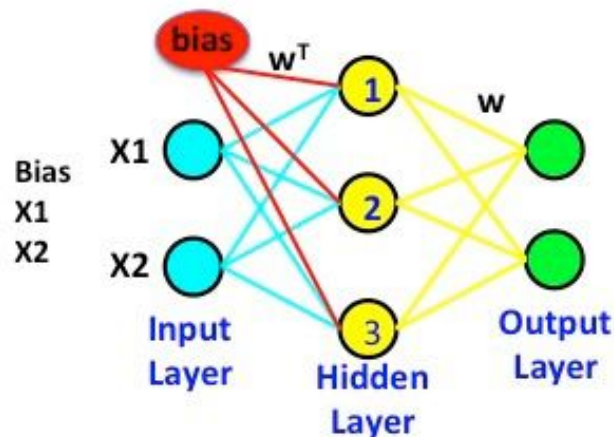
$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} .5 & .5 & .5 \\ .5 & .5 & .5 \\ .5 & .5 & .5 \end{bmatrix} =$$

Weights w^T (transposed)

Go to Hidden Nodes

1 2 3

3 x 3



Hidden Layer

$$= \begin{bmatrix} 1 & 1 & 1 \\ .5 & .5 & .5 \\ .5 & .5 & .5 \\ 1 & 1 & 1 \end{bmatrix}$$

4 x 3

linear

Weights

$$\begin{bmatrix} .2 & .1 \\ .4 & .1 \\ .4 & .1 \end{bmatrix}$$

3 x 2

Output Layer

$$= \begin{bmatrix} 1 & .3 \\ .5 & .15 \\ .5 & .15 \\ 1 & .3 \end{bmatrix}$$

4 x 2

Output

$$= \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$$

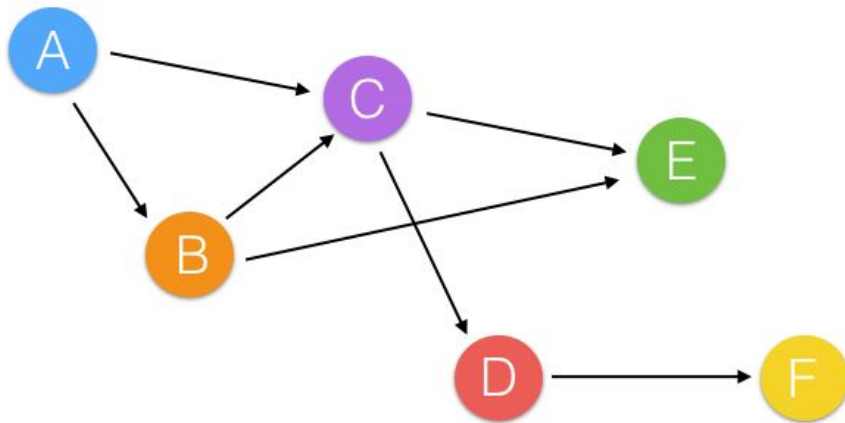
a_i

in_i

Граф вычислений

- процесс вычисления выходного значения нейронной сети можно представить в виде графа
- перемещаемся от входа нейронной сети к выходу, выполняя операции
- структура сети может быть сложной
- перед запуском вычислений необходимо определить последовательность выполнения операций

Топологическая сортировка



Обучение нейронной сети

Функция потерь

- необходимо сформулировать оптимизационную задачу, т.е. выбрать функцию потерь
- функция потерь определяет величину штрафа в случае неверной классификации
- функция потерь должна быть дифференцируема по параметрам модели
- решение задачи сводится к подбору параметров модели при которых функция потерь будет минимальна

Бинарная классификация - Logistic Loss

$$\tilde{y} = \frac{1}{1 + e^{-(hw^T + b)}}$$

$$E_i(\tilde{y}) = - [y_i \log(\tilde{y}_i) + (1 - y_i) \log(1 - \tilde{y}_i)]$$

i - номер объекта в обучающей выборке

\tilde{y}_i - предсказание вероятности модели для i -го объекта

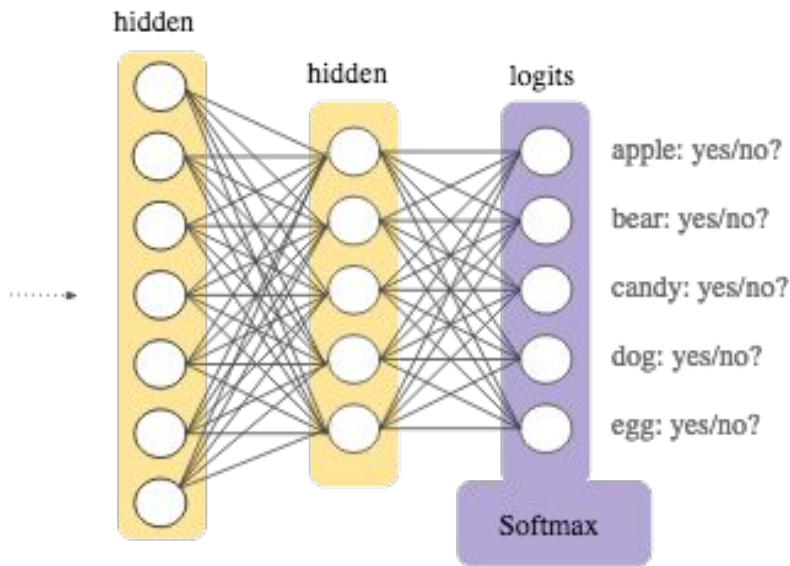
y_i - истинное значение i -го объекта из обучающей выборки

h - вектор значений скрытого слоя

w - вектор весов выходного слоя

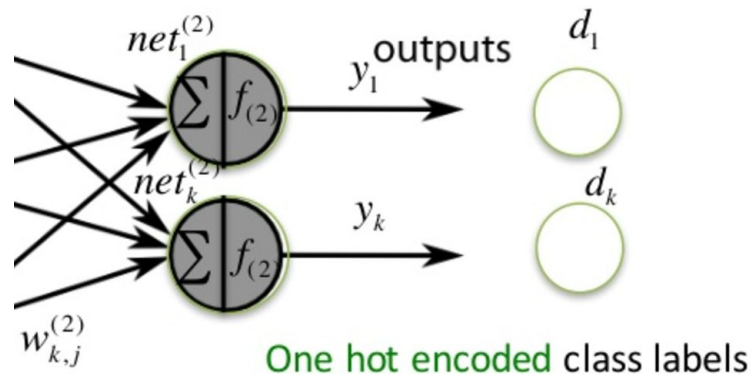
b - свободный член

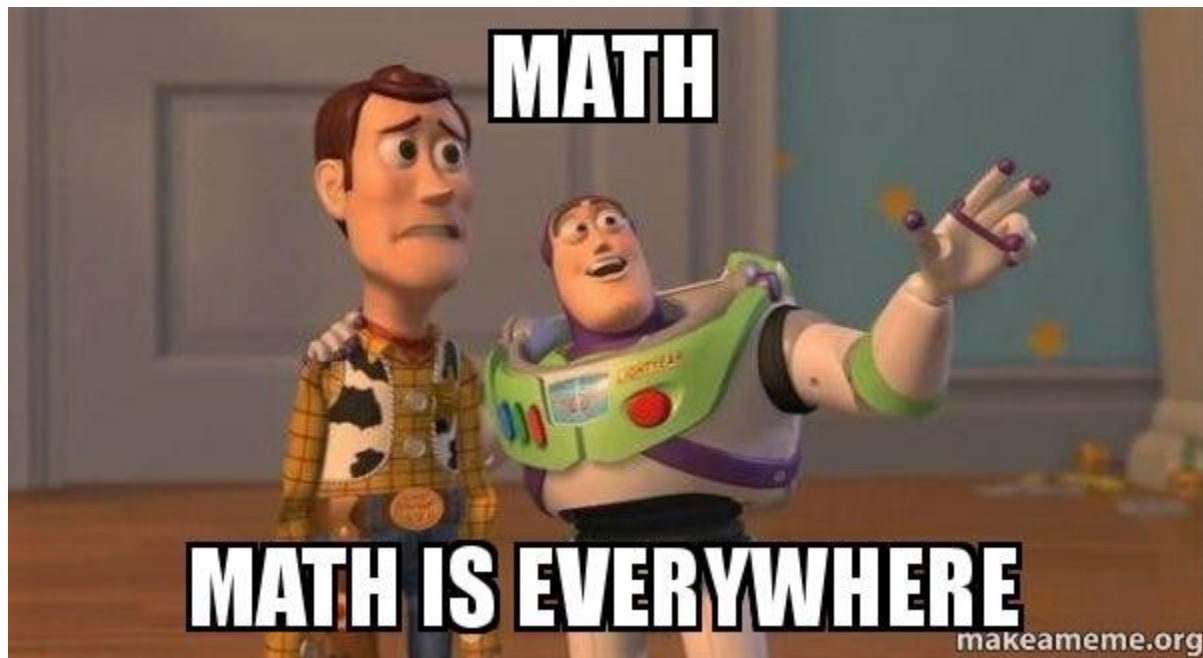
Несколько классов - Softmax (Cross-Entropy)



$$E(w) = - \sum_k (d_k \log y_k + (1 - d_k) \log(1 - y_k))$$

$$y_k = \frac{\exp(\sum_j w_{k,j}^{(2)} h_j^{(1)})}{\sum_{k'} \exp(\sum_j w_{k',j}^{(2)} h_j^{(1)})}$$





Минимизация функции потерь

$$w_{t+1} := w_t - \eta_t \frac{1}{N} \sum_{i=1}^N \frac{\partial E(w)}{\partial w}$$

w - параметр модели

η - параметр скорости обучения

N - число объектов в выборке/батче

E - функция потерь

Дифференцирование сложной функции

$$\frac{\partial E(\tilde{y}(w))}{\partial w} = \frac{\partial E(\tilde{y}(w))}{\partial \tilde{y}(w)} \frac{\partial \tilde{y}(w)}{\partial w}$$

Дифференцирование сложной функции

$$f(x, y, z) = (x + y)z$$

$$q = x + y$$

$$f = qz$$

Дифференцирование сложной функции

$$f(x, y, z) = (x + y)z$$

$$f = qz$$

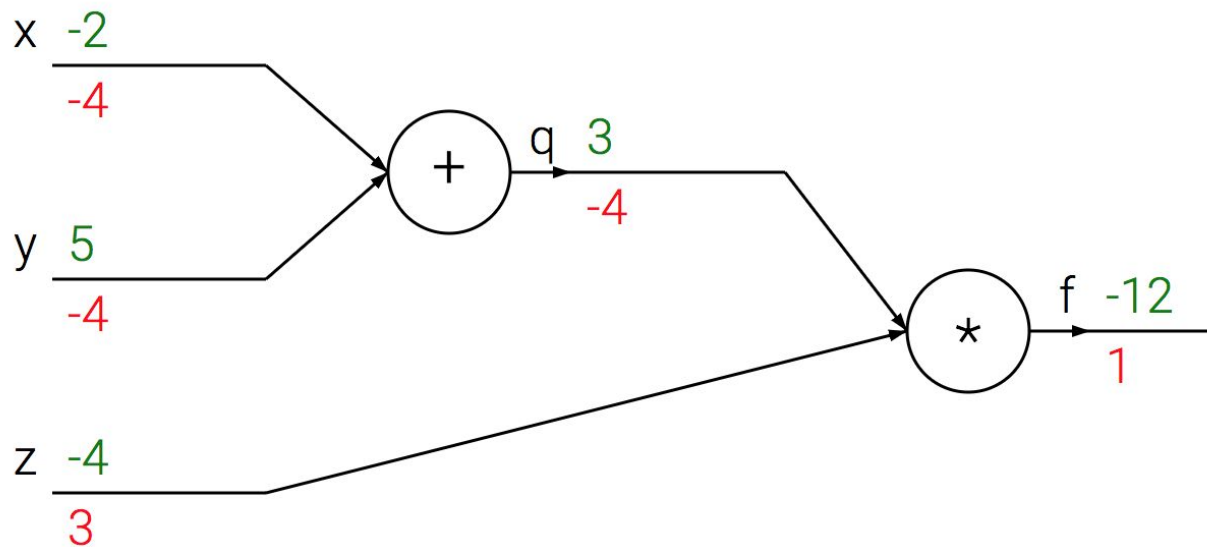
$$q = x + y$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

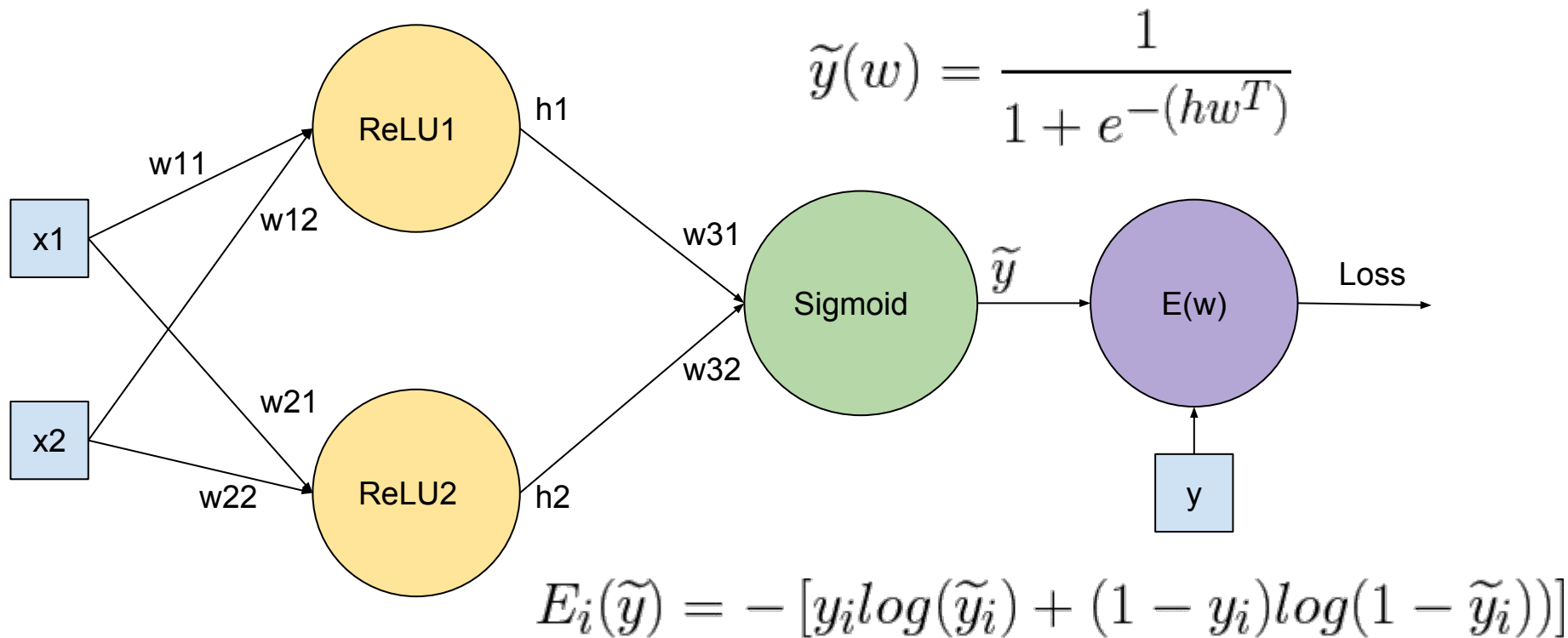
$$\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

Обратное распространение ошибки

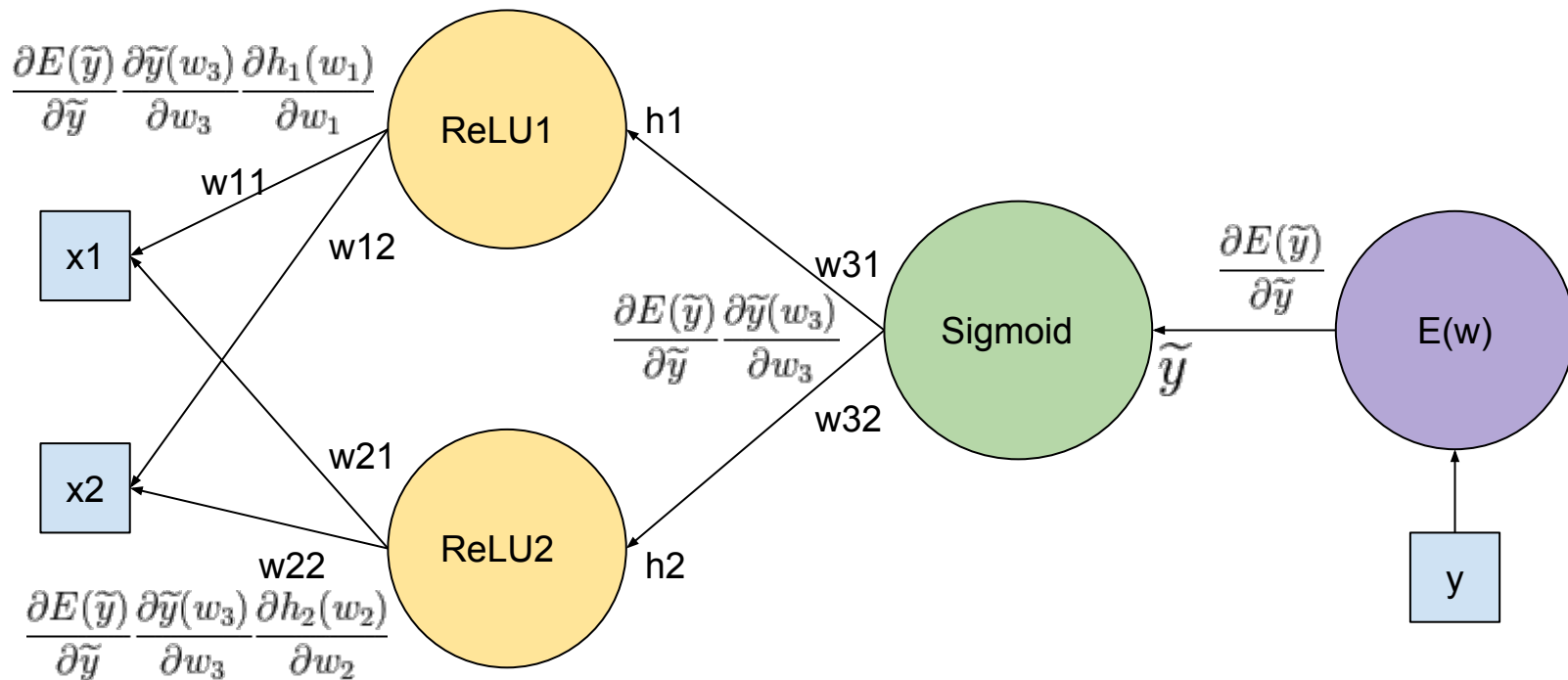
$$f(x, y, z) = (x + y)z$$



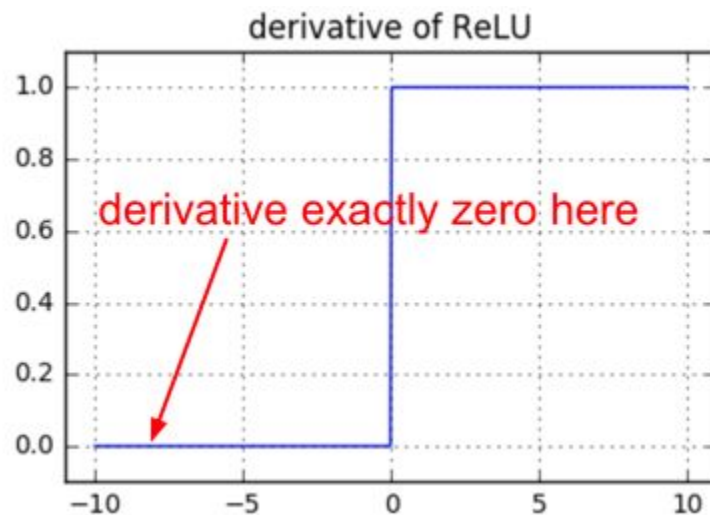
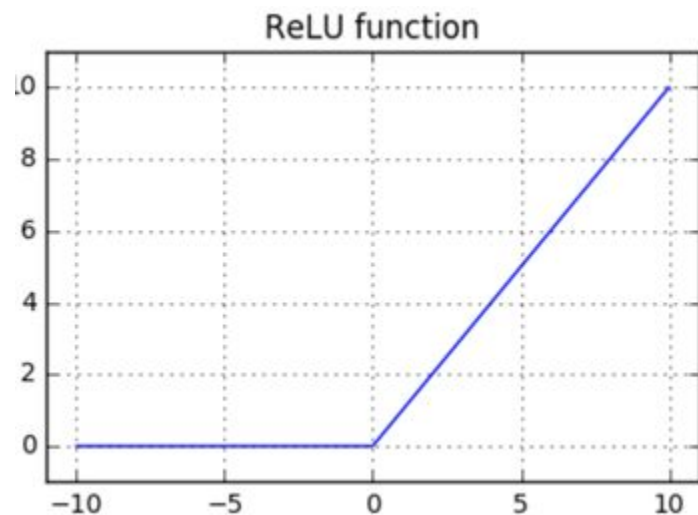
Обратное распространение ошибки



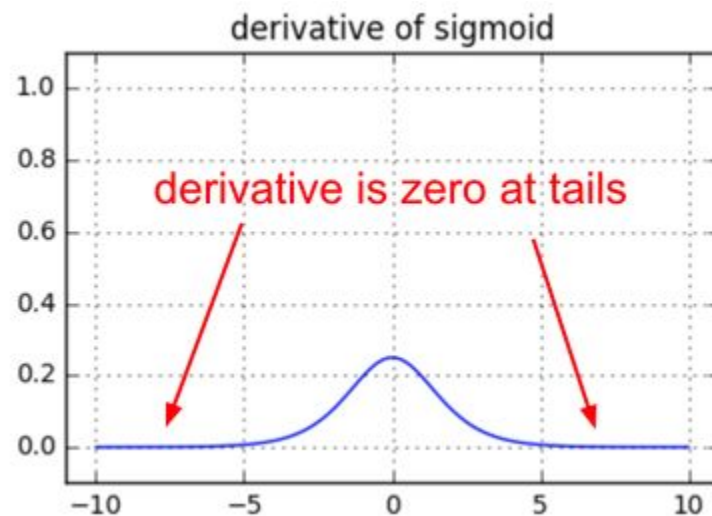
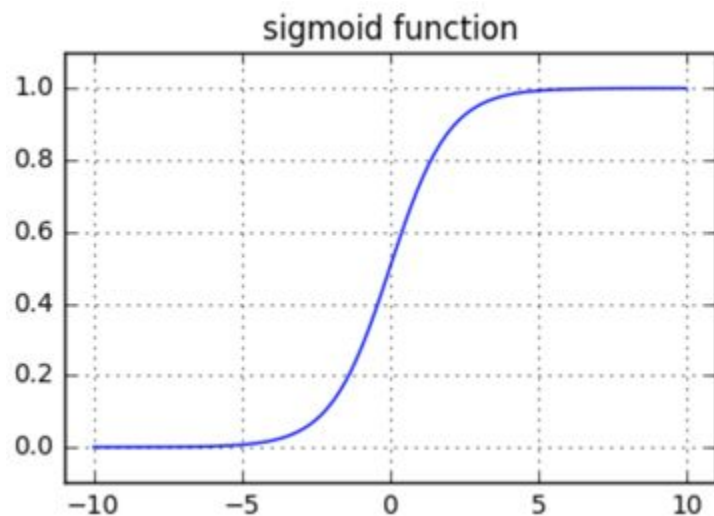
Обратное распространение ошибки




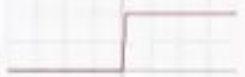



Исчезающие градиенты







Исчезающие градиенты



Функции активации

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$

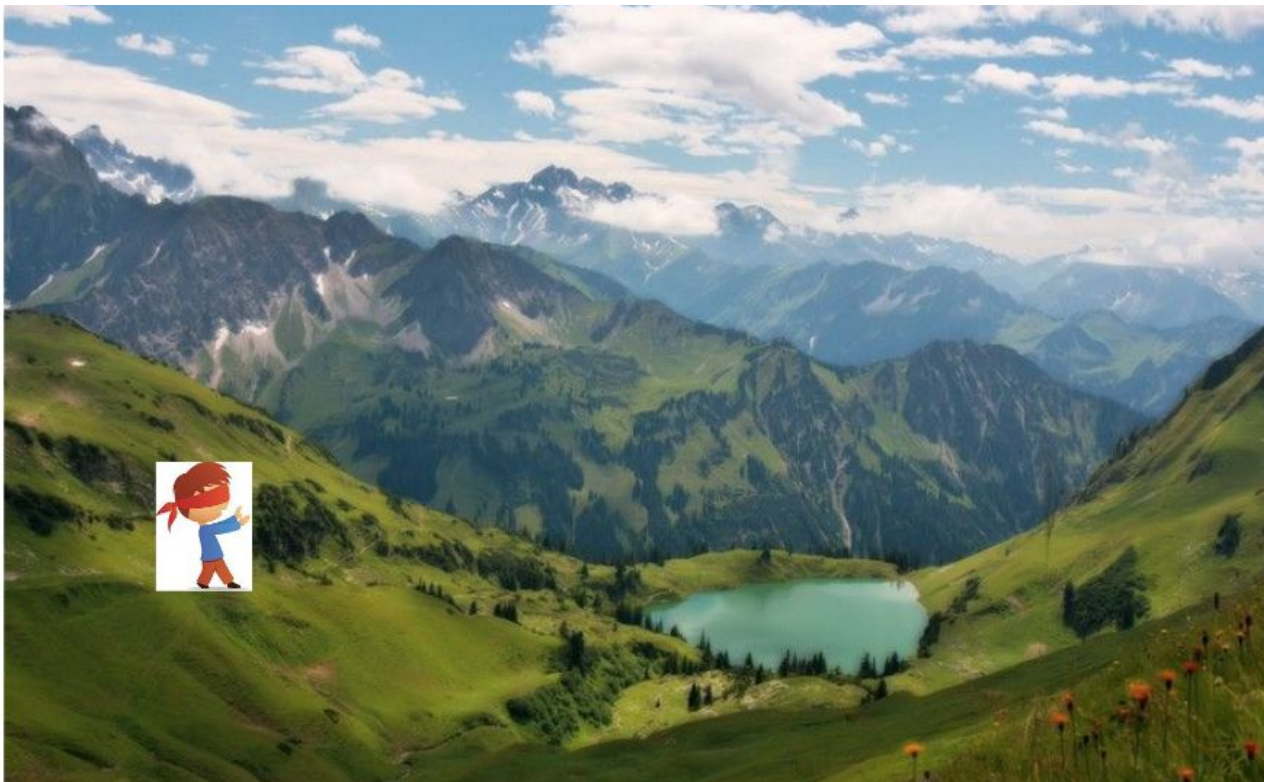
Функции активации

Name	Plot	Equation	Derivative
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric Rectified Linear Unit (PReLU)		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Инициализация весов

- не стоит инициализировать одинаковыми значениями
- простой способ: случайные числа из равномерного распределения в диапазоне 0..1
- можно инициализировать случайными ортогональными значениями
- некоторые алгоритмы инициализируют веса пропорционально числу входов в узел

Градиентный спуск



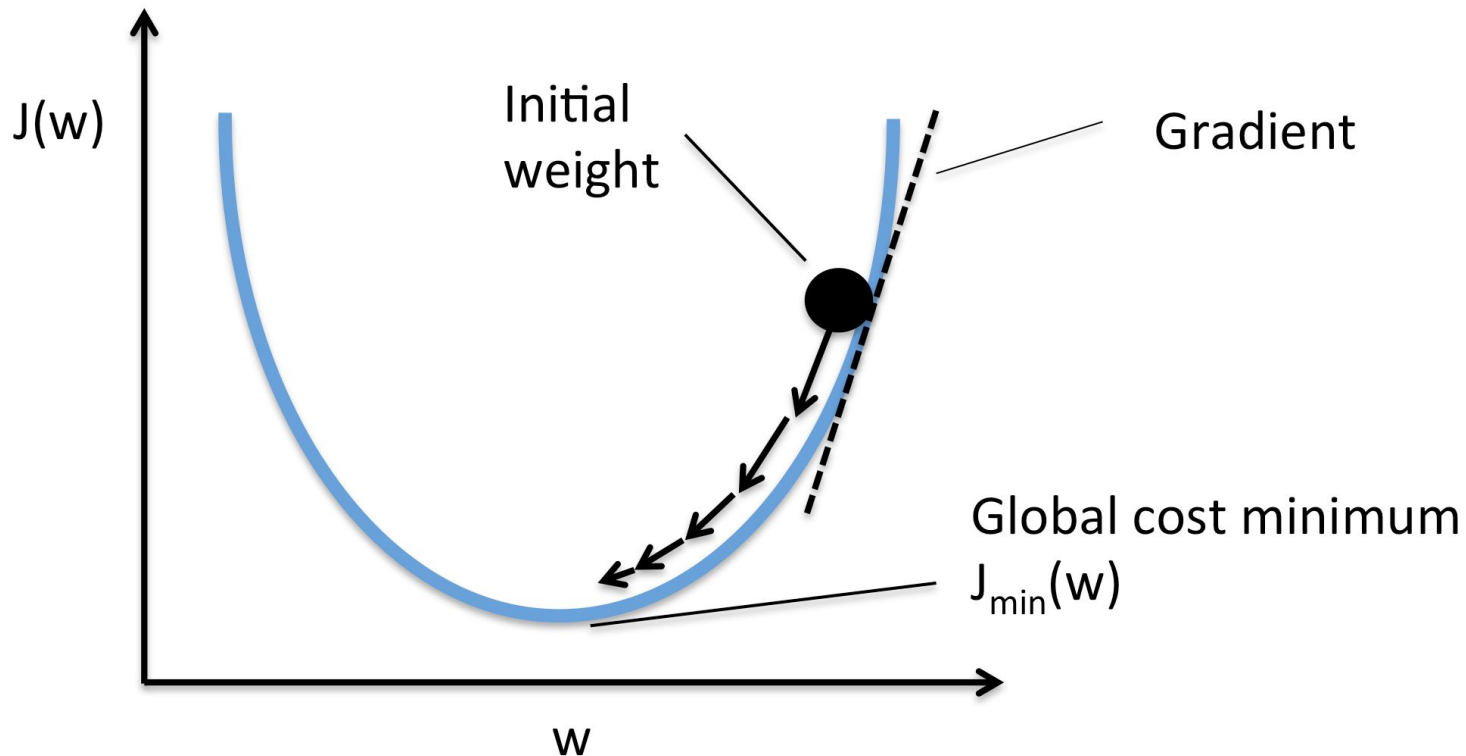
Градиентный спуск

- оцениваем направление градиента функции потерь в текущей точке (для текущих значений параметров модели)
- делаем шаг в направлении противоположном направлению вектора градиента
- оценку градиента можно уточнить взяв среднее оценок по нескольким примерам из обучающей выборки (batch)

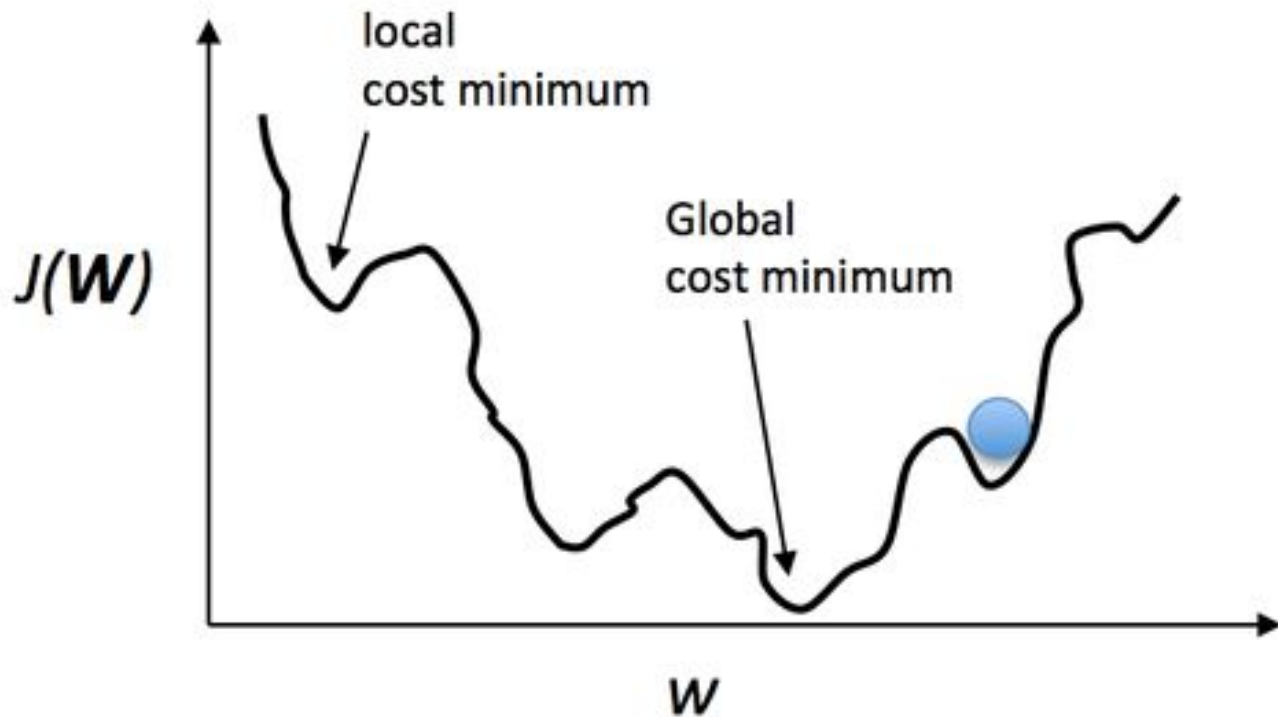
Градиентный спуск

- необходимо чтобы функция была выпуклой (можно посчитать производную в любой точке)
- при большом числе параметров функция потерь как правило имеет локальные минимумы
- седловые точки - такие точки градиент в которых равен нулю, при этом точка не является ни минимумом, ни максимумом

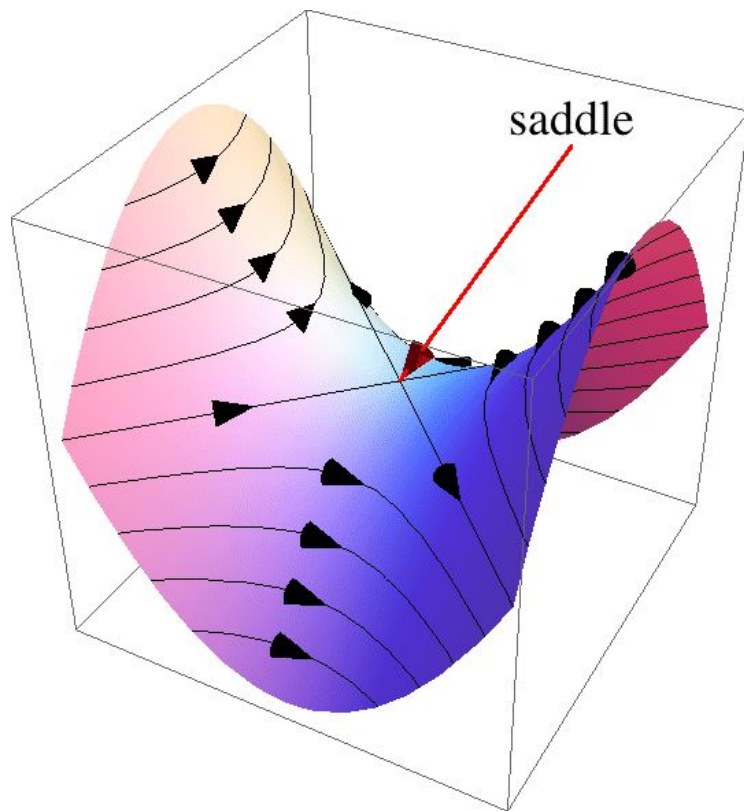
Градиентный спуск



Градиентный спуск



Седловая точка

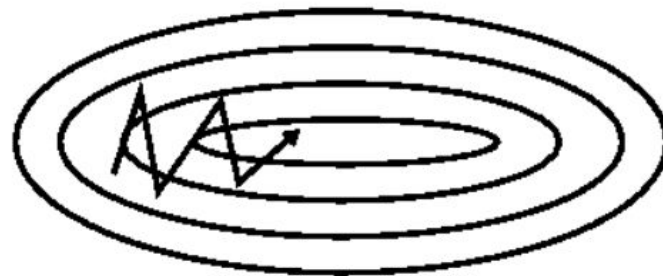
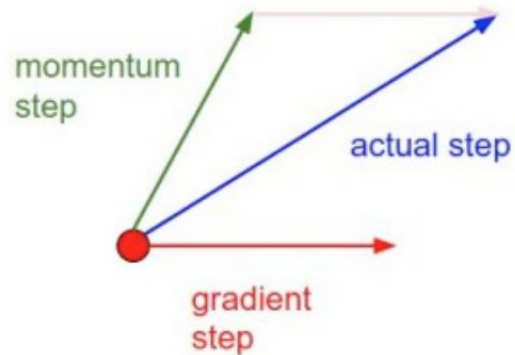


Оптимизация

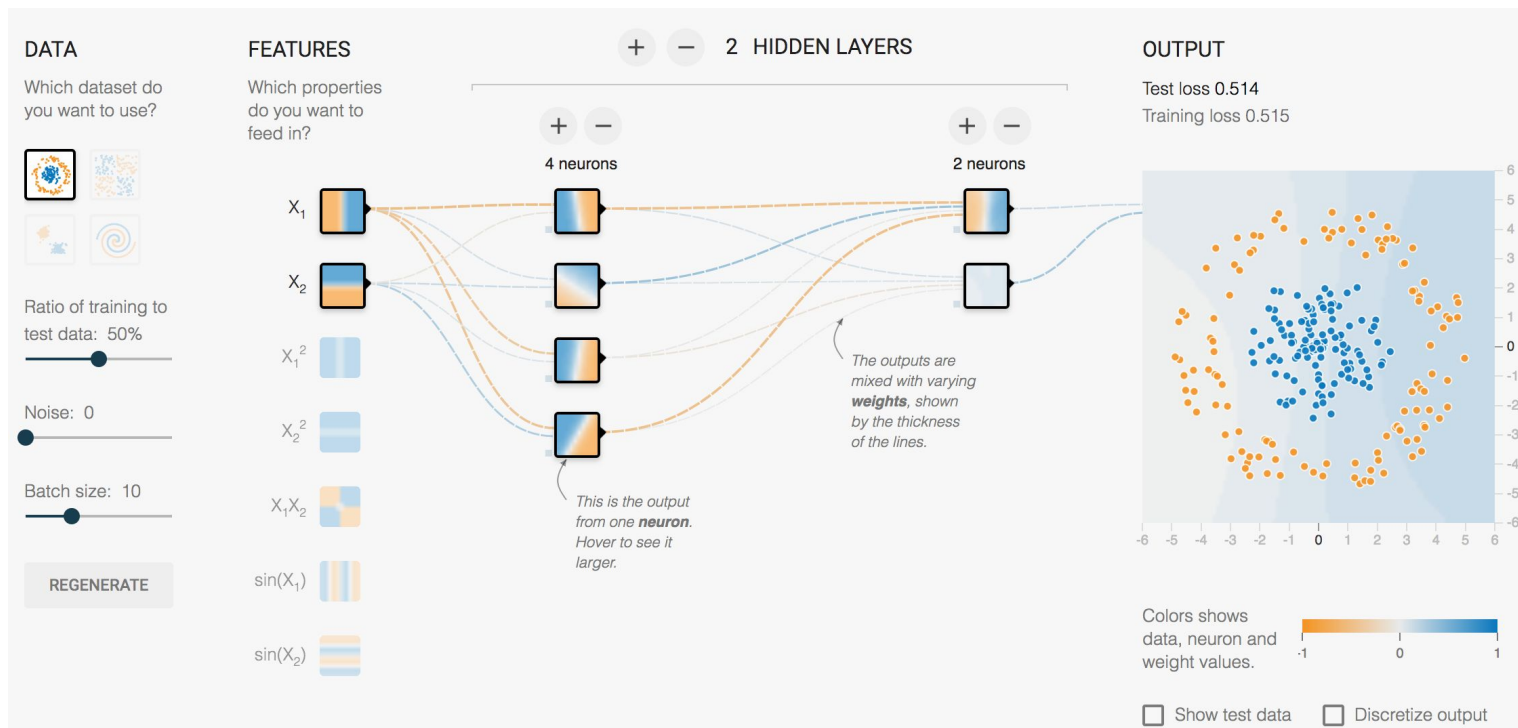
- существует несколько оптимизационных алгоритмов основанных на градиентном спуске
- стандартного градиентного спуска может быть недостаточно для стабильного и быстрого поиска минимума
- основная идея алгоритмов - коррекция длины и направления вектора градиента с учетом предыдущих итераций

Оптимизация - Момент

Momentum update



Tensorflow Playground



<http://playground.tensorflow.org/>

Резюме

- изучили принцип работы нейронной сети
- познакомились с процессом обучения нейронной сети
- реализовали решение задачи классификации MNIST с помощью фреймворка Keras

Полезные материалы

- [Neural Networks and Deep Learning](#)
- [Deep Learning](#)
- [CS231n Winter 2016](#)
- [Yes you should understand backprop](#)
- [An overview of gradient descent optimization algorithms](#)
- [The Neural Network Zoo](#)
- [Tensorflow Playground](#)
- [Loss Functions](#)