

Введение в нейронные сети

ЦИФРОВАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ

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

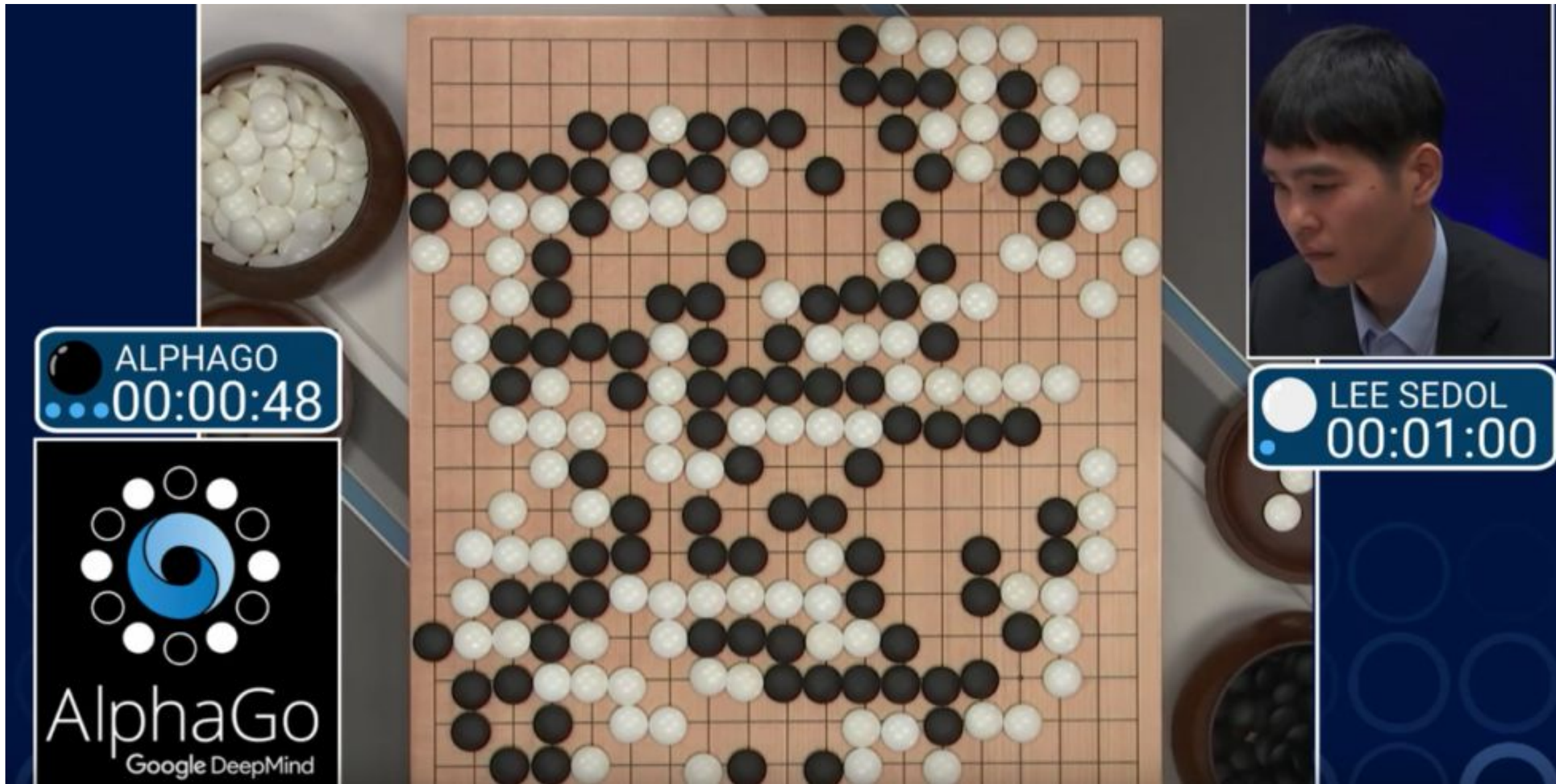
Обучение

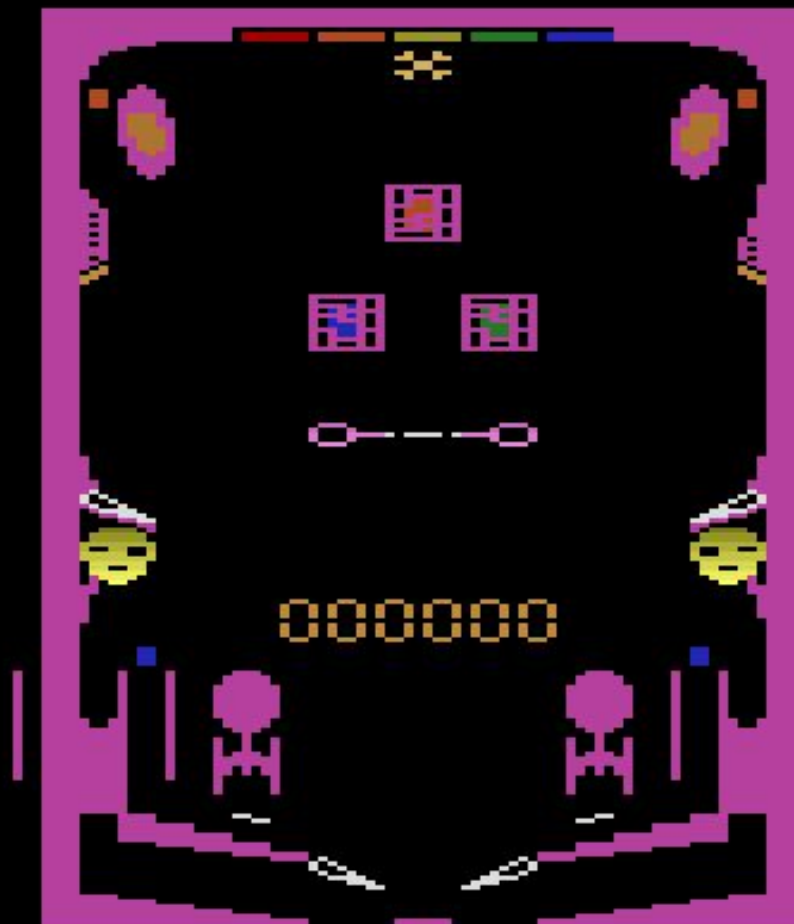
- Функции потерь — логит, софтмакс
- Градиентный спуск и методы оптимизации
- Обратное распространение градиента и граф вычислений, автоматическое дифференцирование
- Регуляризация
- Инициализация весов

ПРИМЕРЫ ПРИМЕНЕНИЯ НЕЙРОННЫХ СЕТЕЙ

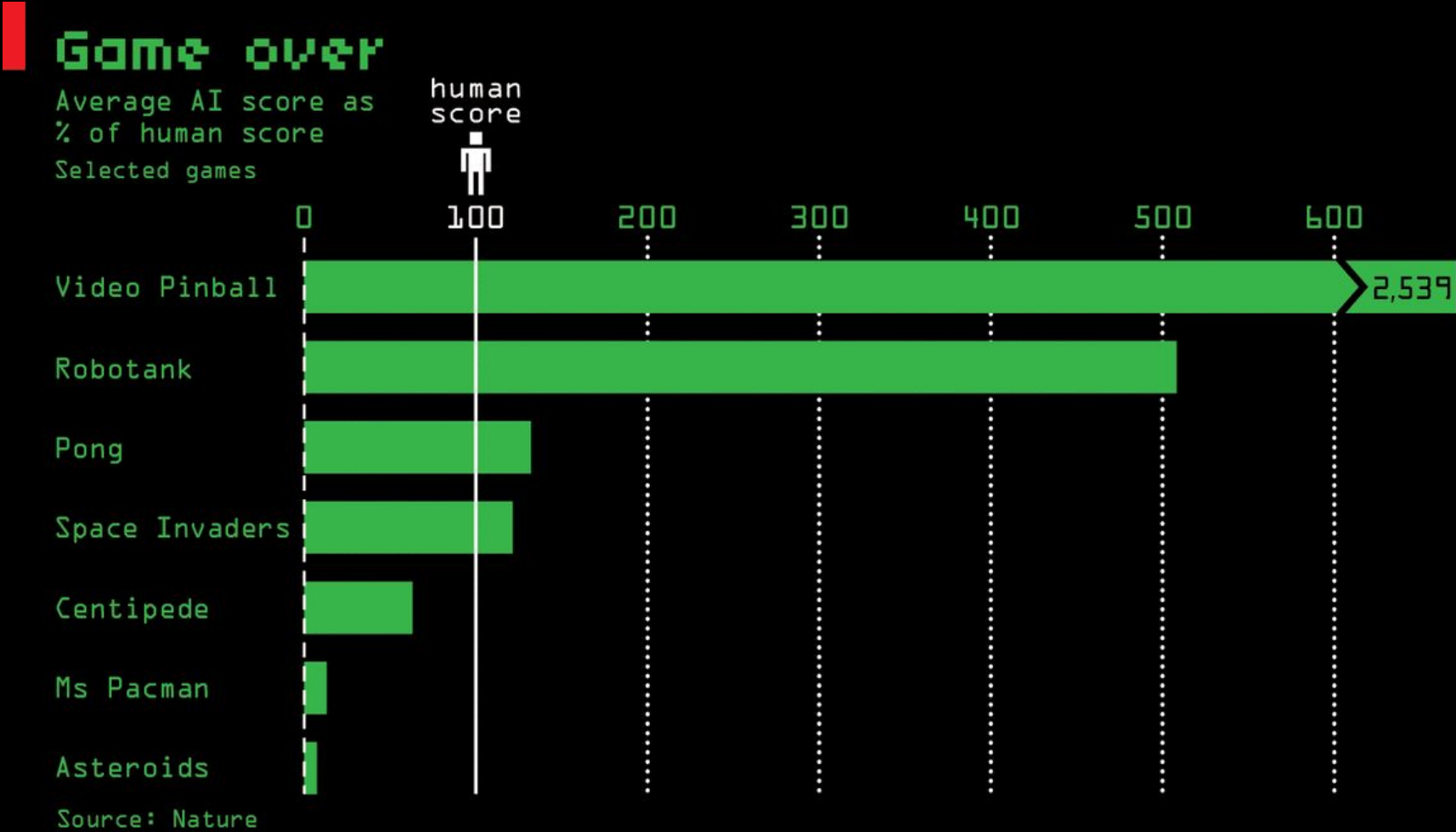
ПРИМЕРЫ ПРИМЕНЕНИЯ НЕЙРОННЫХ СЕТЕЙ

ALPHAGO



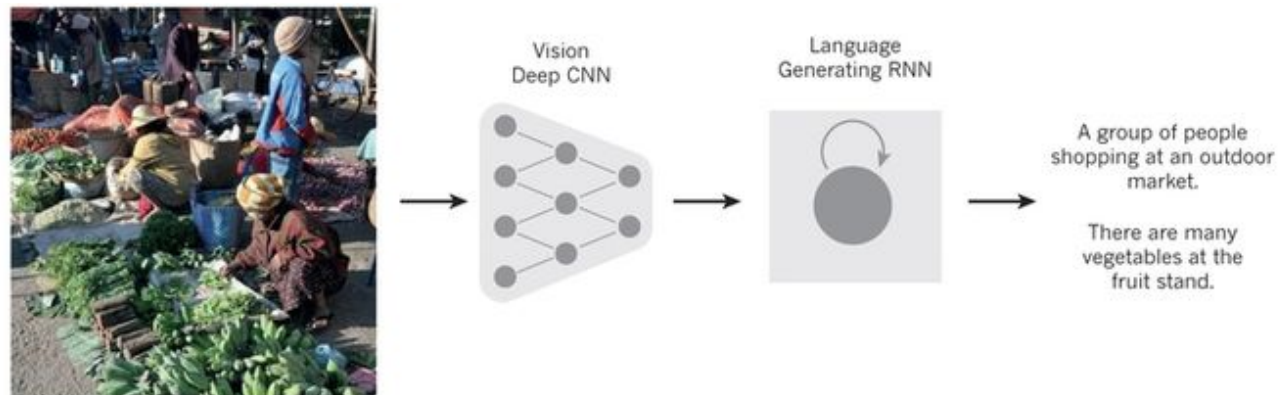


PINBALL



ПРИМЕРЫ ПРИМЕНЕНИЯ НЕЙРОННЫХ СЕТЕЙ

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.

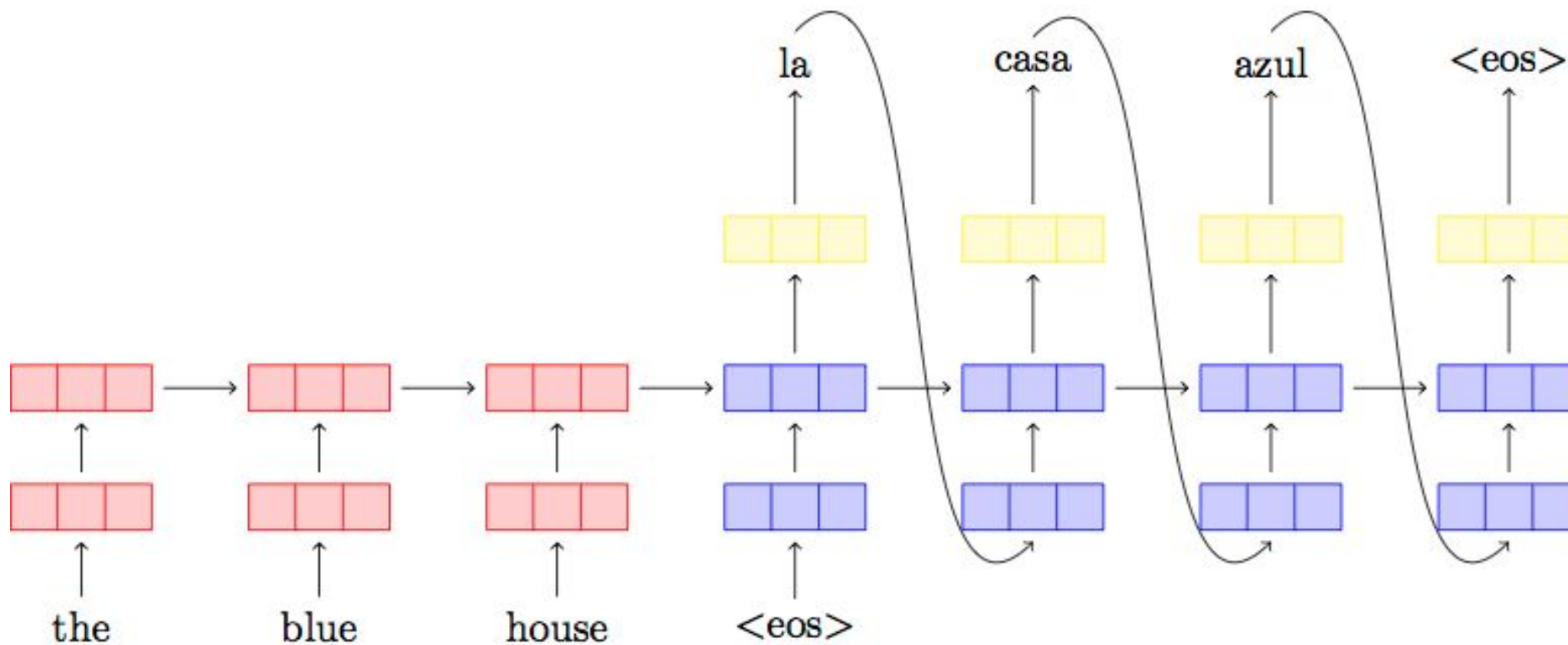


A group of **people** sitting on a boat in the water.



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

Машинный перевод

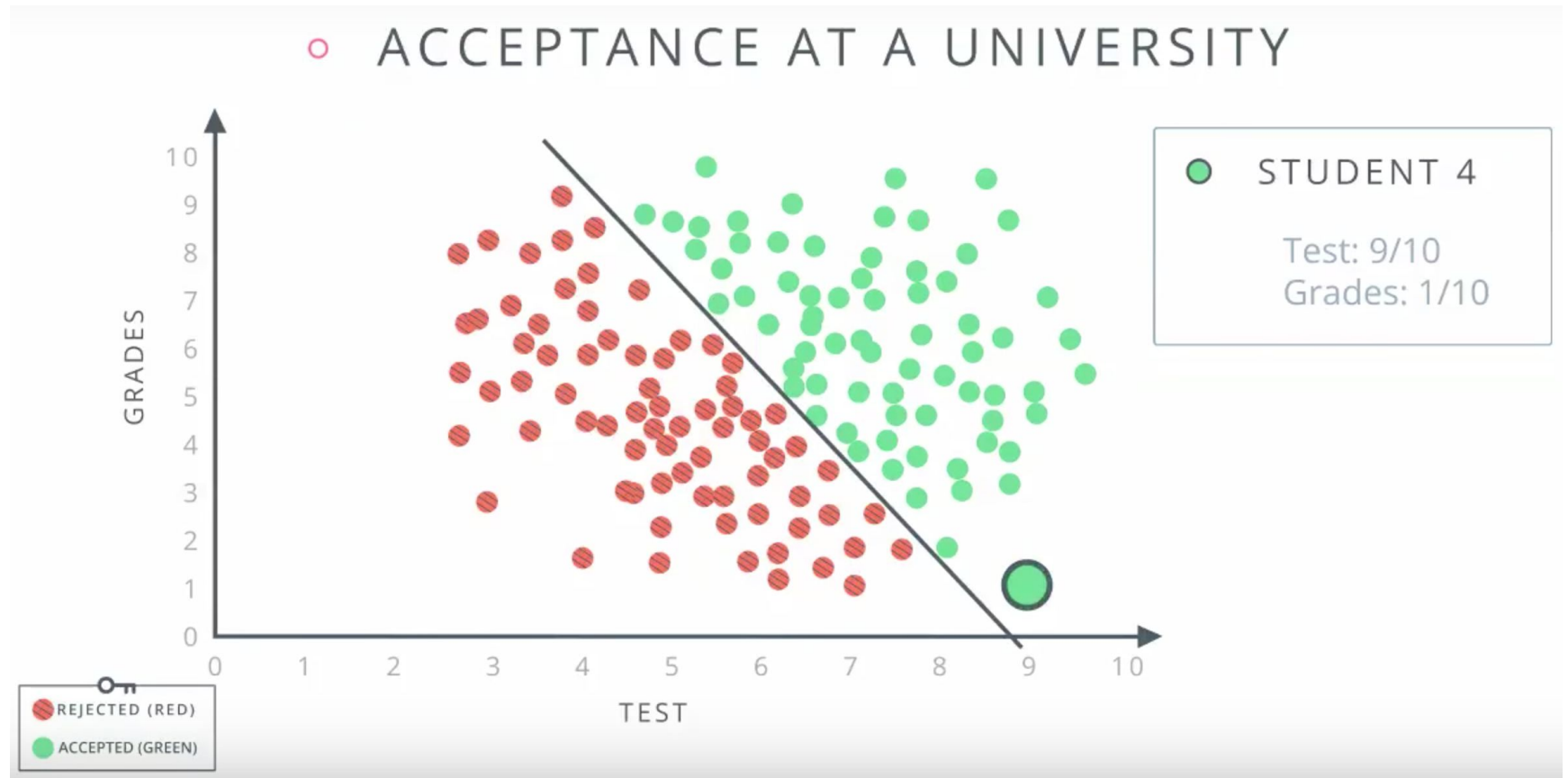


ПРИНЦИП РАБОТЫ НЕЙРОННОЙ СЕТИ

ПРИНЦИП РАБОТЫ НЕЙРОННОЙ СЕТИ

Задача.

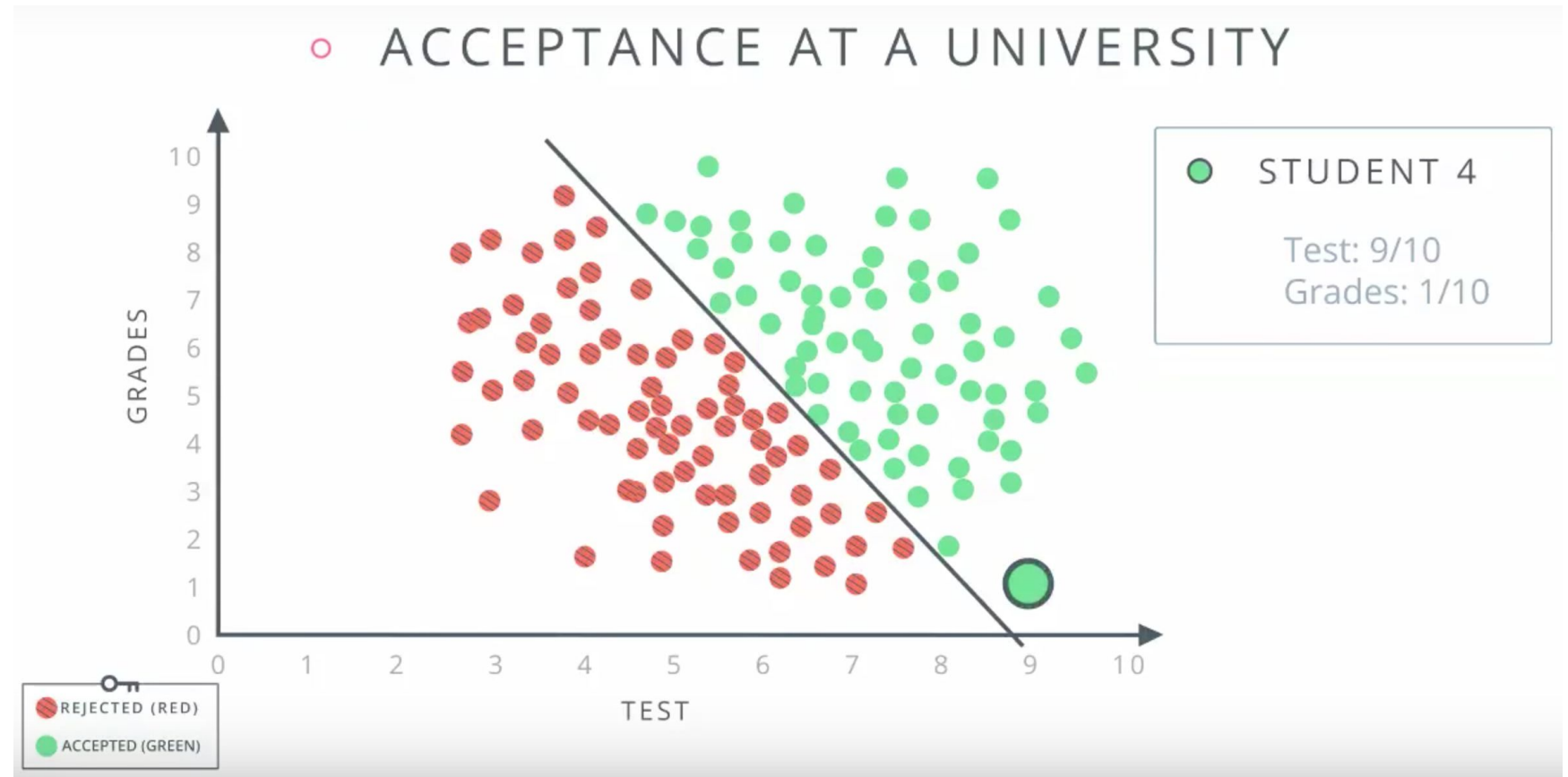
Предсказать поступление в университет по результатам теста и оценкам



ПРИНЦИП РАБОТЫ НЕЙРОННОЙ СЕТИ

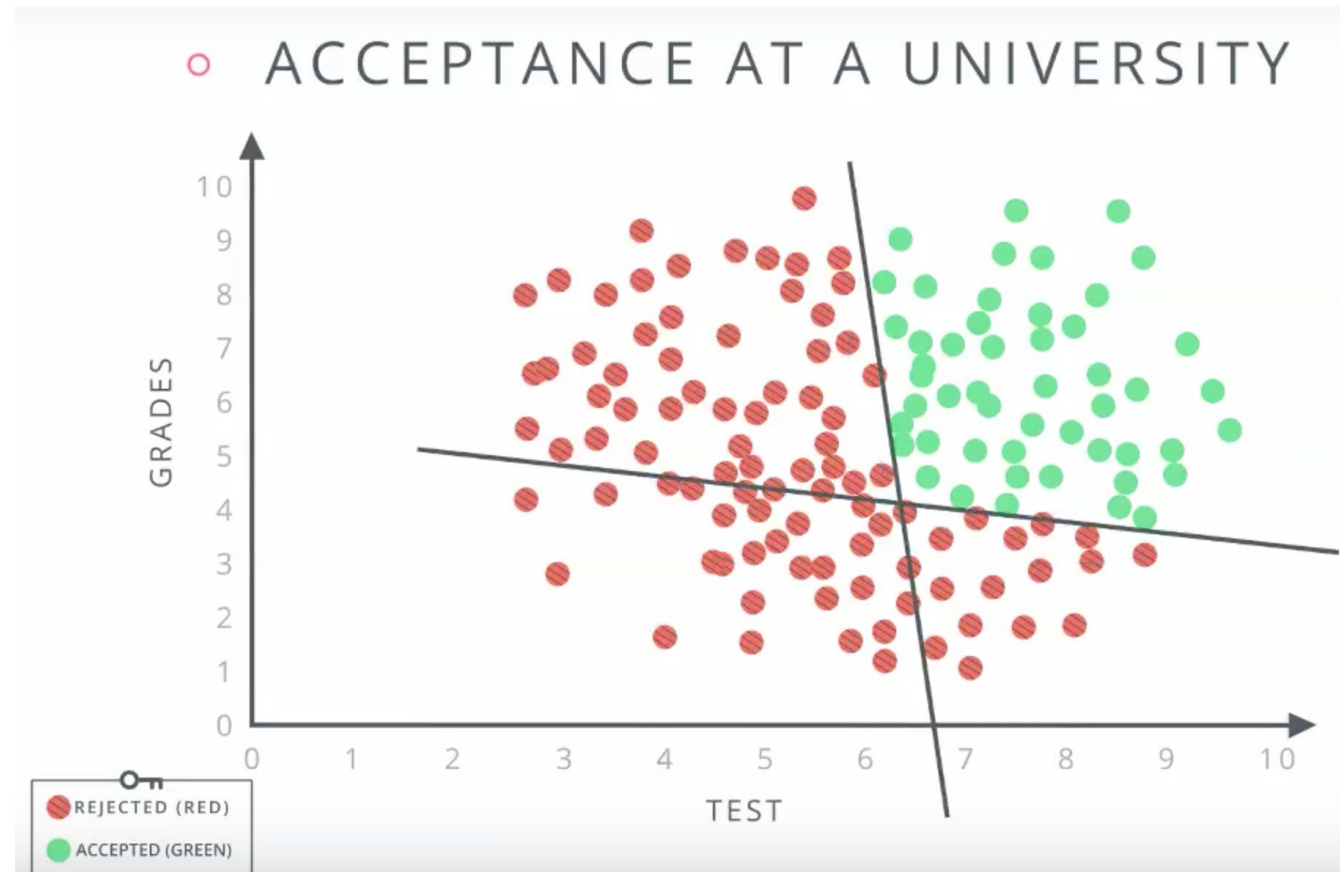
Задача.

Предсказать поступление в университет по результатам теста и оценкам



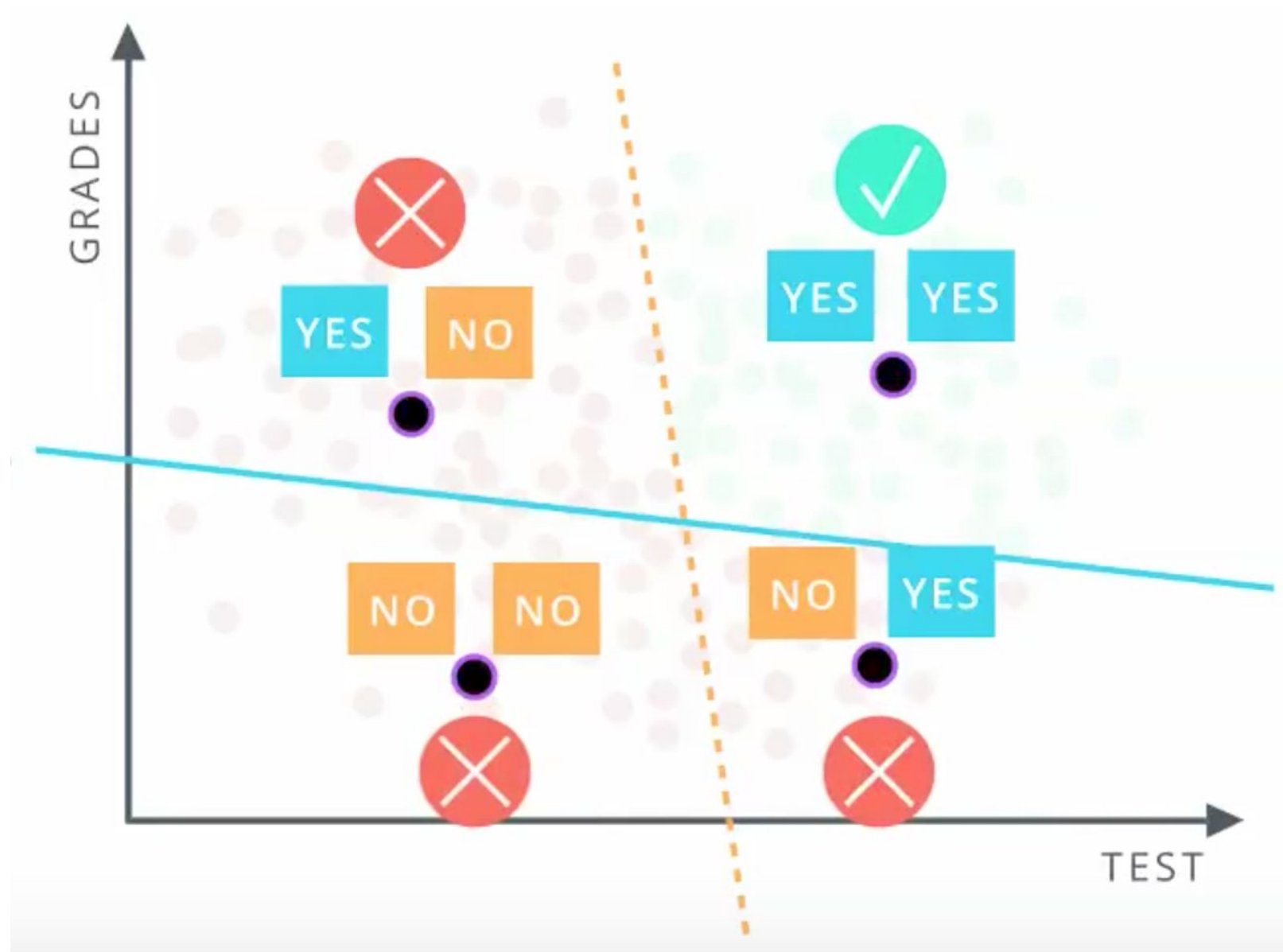
ПРИНЦИП РАБОТЫ НЕЙРОННОЙ СЕТИ

Линейная регрессия



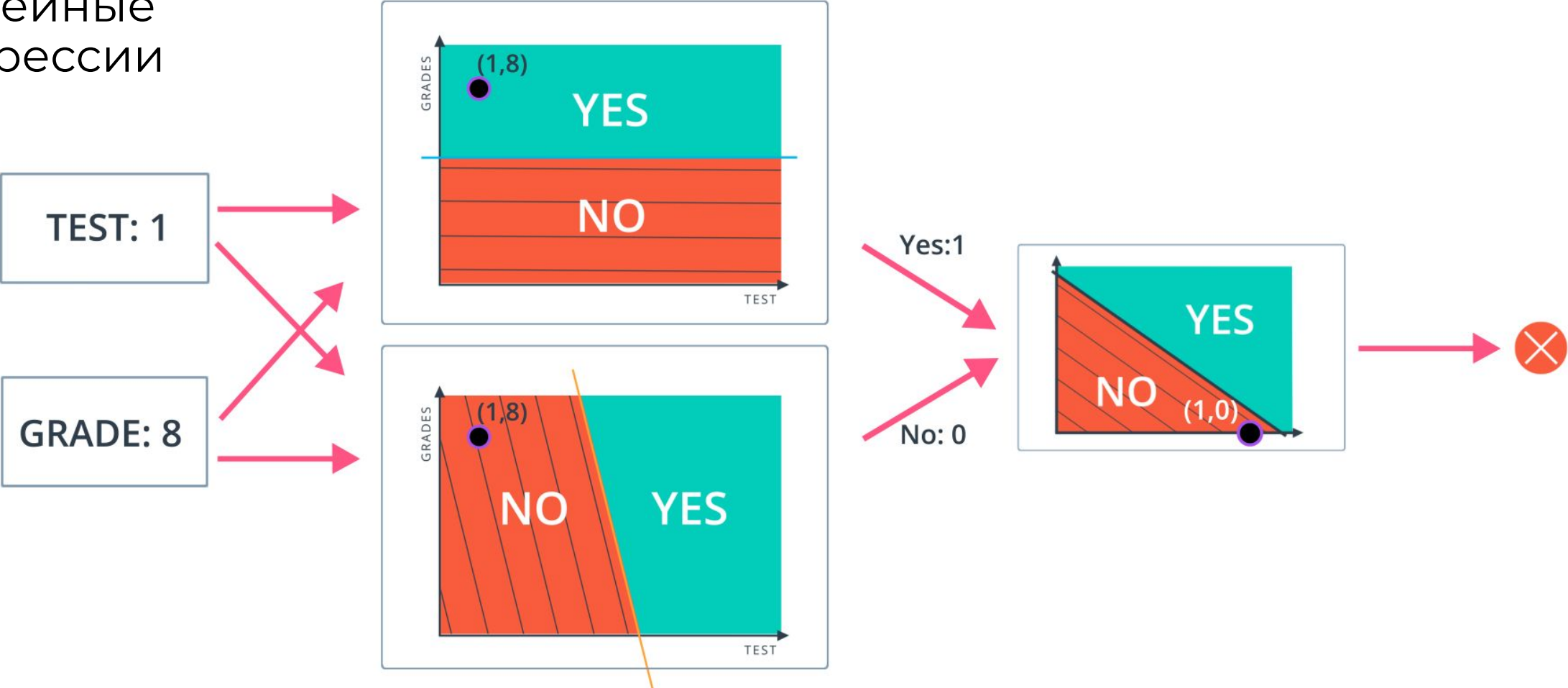
ПРИНЦИП РАБОТЫ НЕЙРОННОЙ СЕТИ

Объединяем
линейные
регрессии



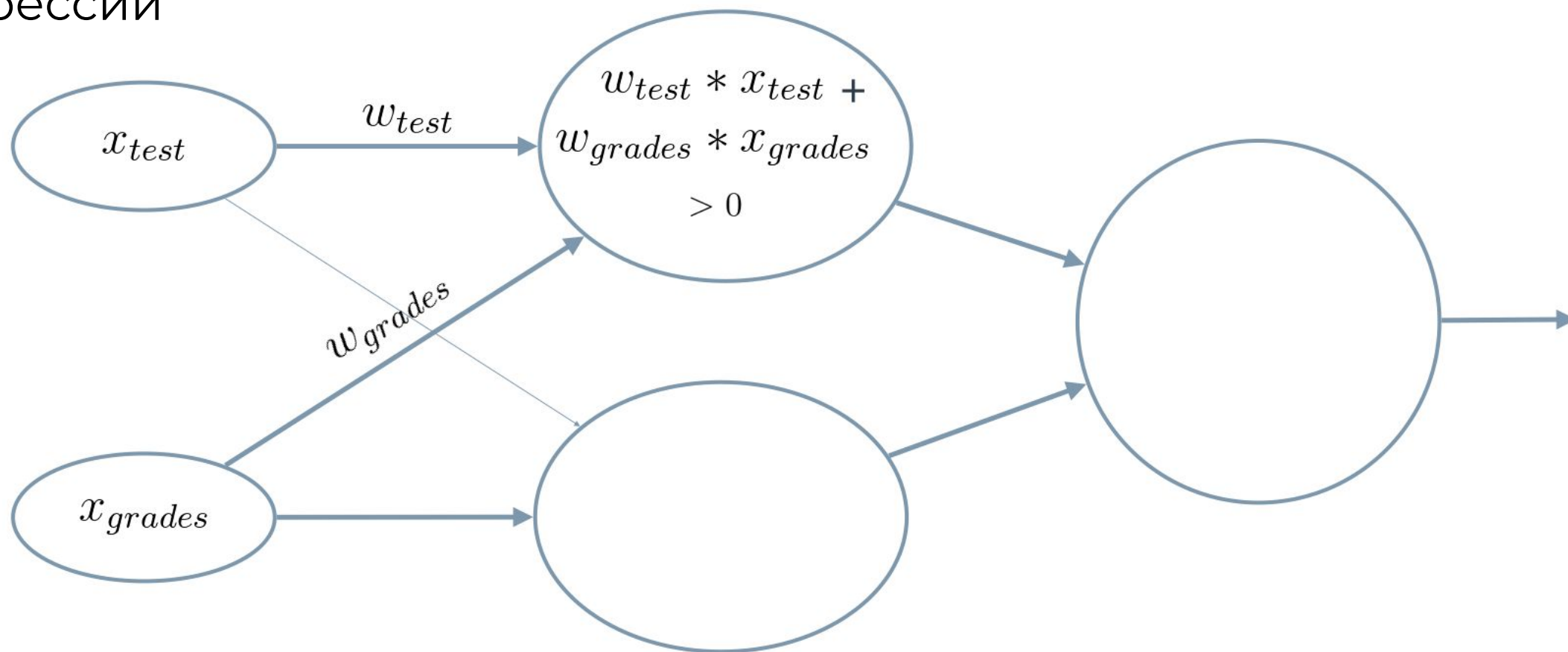
ПРИНЦИП РАБОТЫ НЕЙРОННОЙ СЕТИ

Объединяем
линейные
регрессии

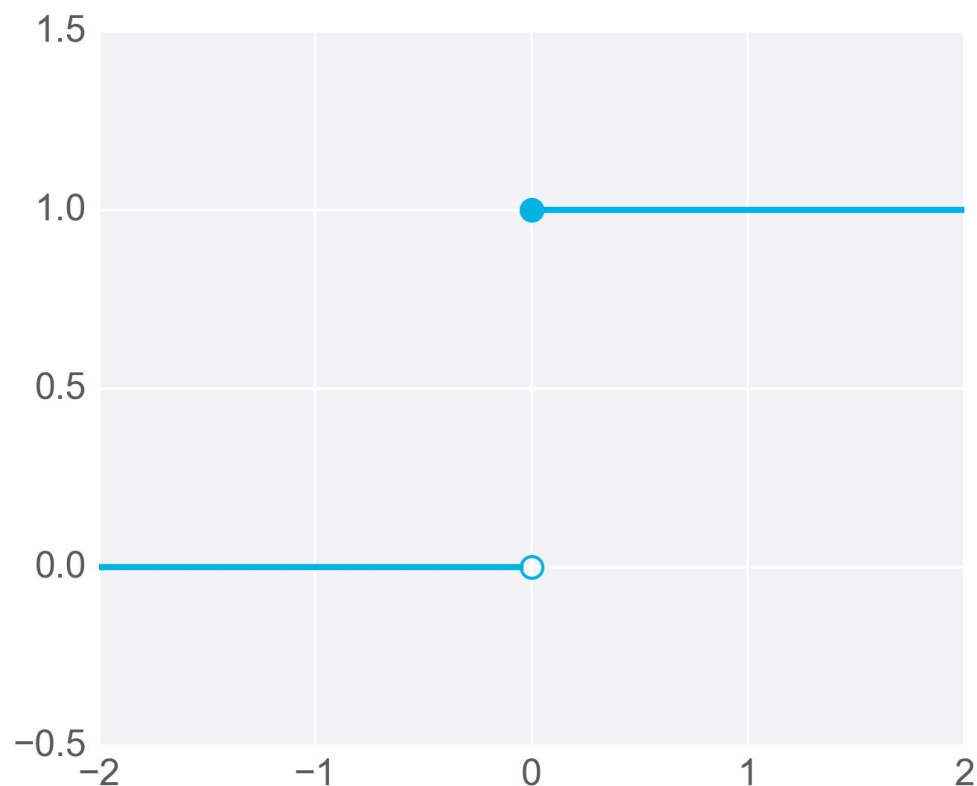


ПРИНЦИП РАБОТЫ НЕЙРОННОЙ СЕТИ

Объединяем
линейные
регрессии



Функция активации



$$f(x_1, x_2, \dots, x_m) =$$

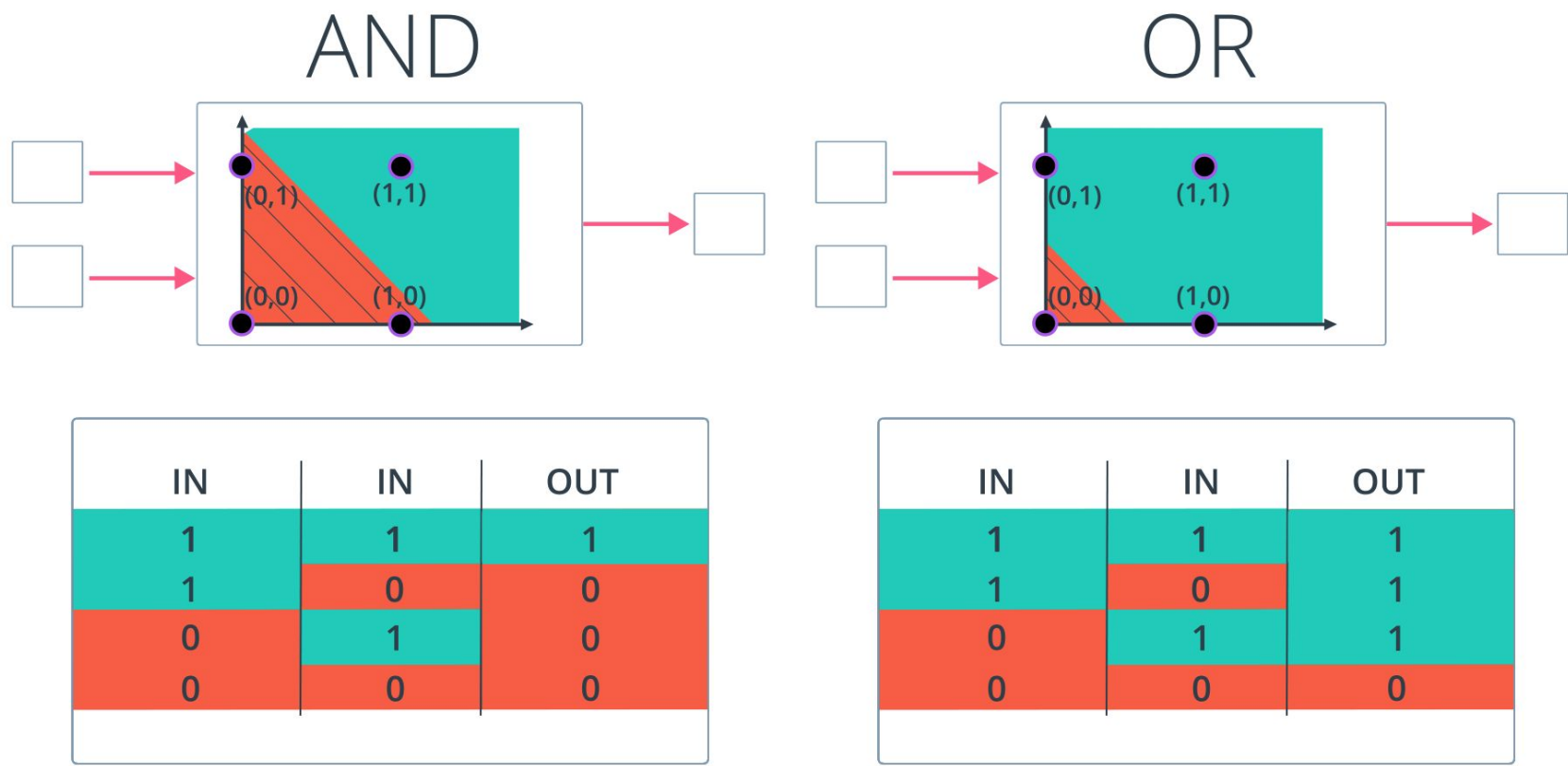
$$\begin{cases} 0 & \text{if } b + \sum w_i \cdot x_i < 0 \\ 1 & \text{if } b + \sum w_i \cdot x_i \geq 0 \end{cases}$$

x значения признаков

w веса регрессии

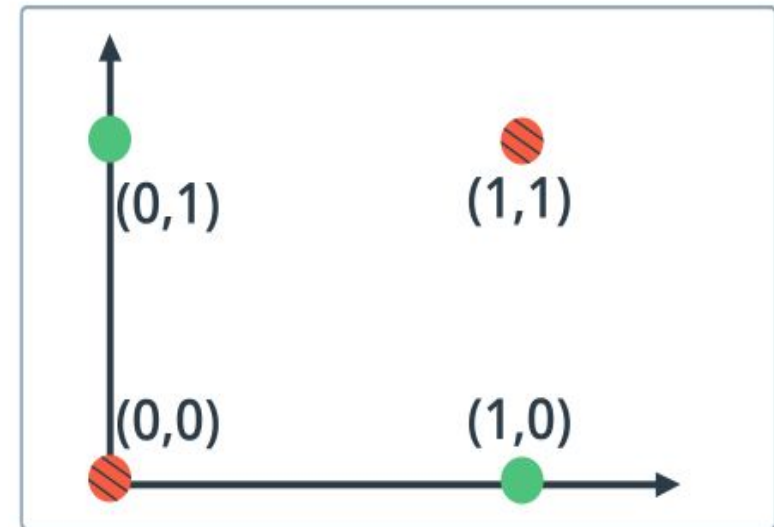
b свободный член

Логические операции. AND, OR

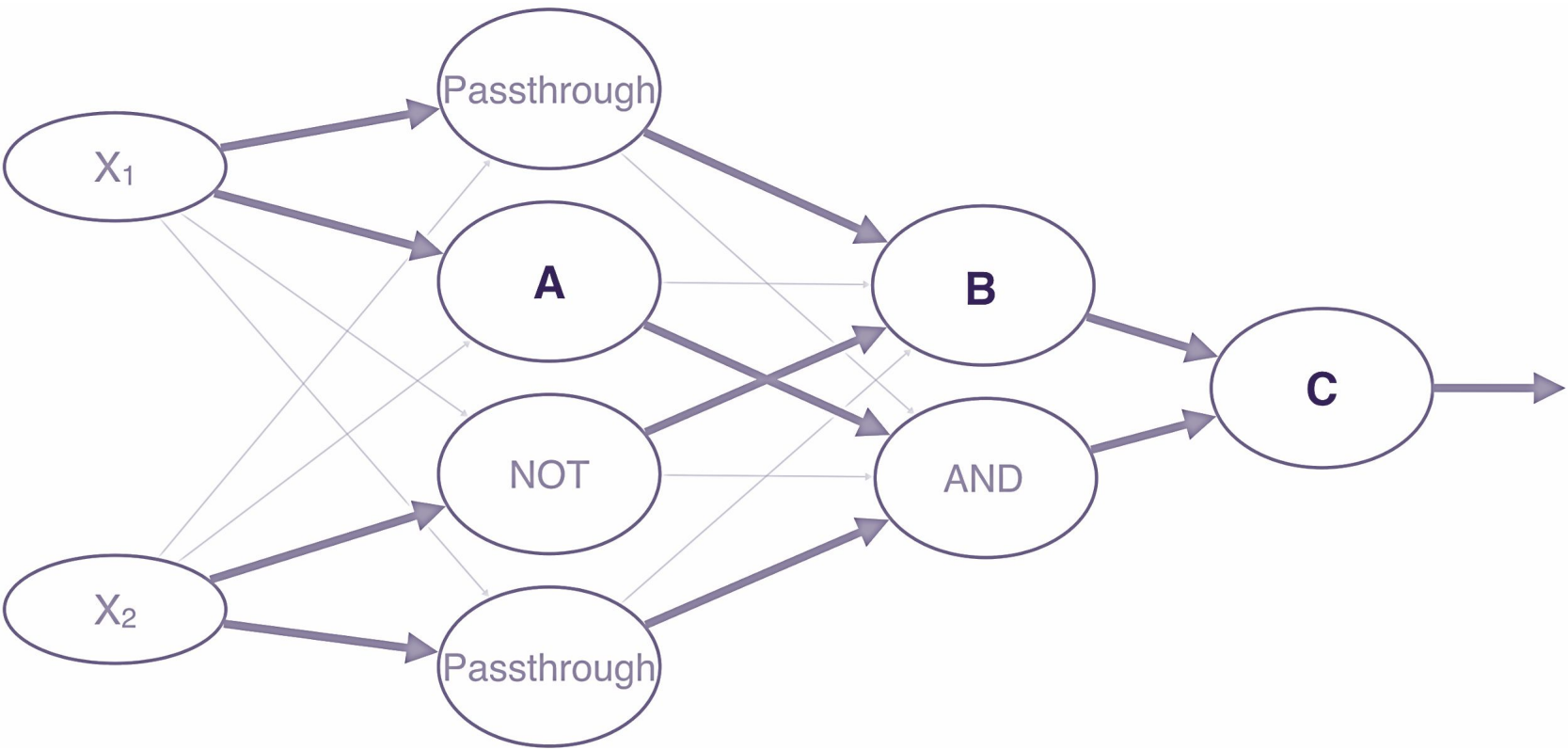


Логические операции. XOR

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



Логические операции. XOR



A

IN	OUT
1	0
0	1

B

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

C

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

Принцип работы нейронной сети

1. Перцептрон может иметь один или несколько входов
2. Значение на каждом входе взвешивается соответствующим весом
3. Выходом перцептрона является линейная комбинация входных значений и соответствующих весов
4. К выходу перцептрона применяется функция активации
5. Изменяя веса перцептрона мы изменяем его функциональность

ПОЛНОСВЯЗНАЯ НЕЙРОННАЯ СЕТЬ

Структура полносвязной сети

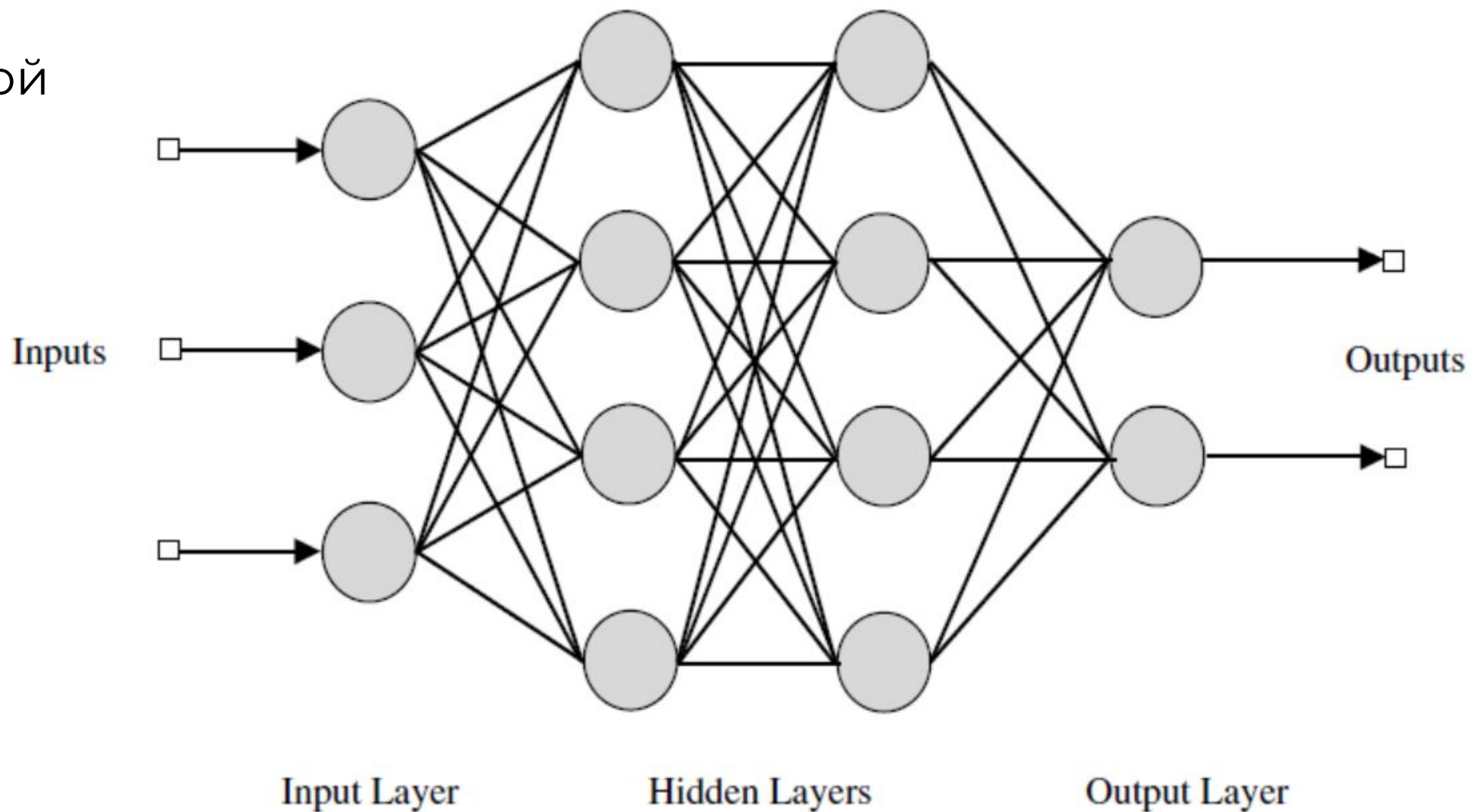
Содержит
один входной слой,
один выходной

И один
или несколько
внутренних
слоев

Каждый нейрон
предыдущего слоя
связан с нейроном
последующего слоя

ПОЛНОСВЯЗНАЯ НЕЙРОННАЯ СЕТЬ

Структура
полносвязной
сети



Функции активации

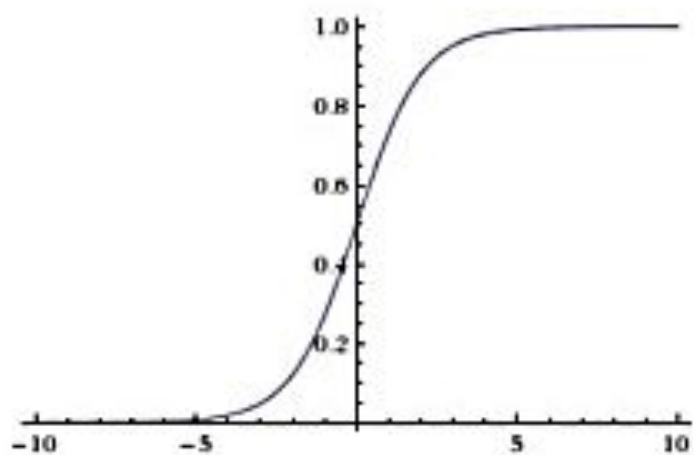
В случае
линейной
функции
активации
**нейронная сеть
вырождается
в линейное
преобразование**

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**

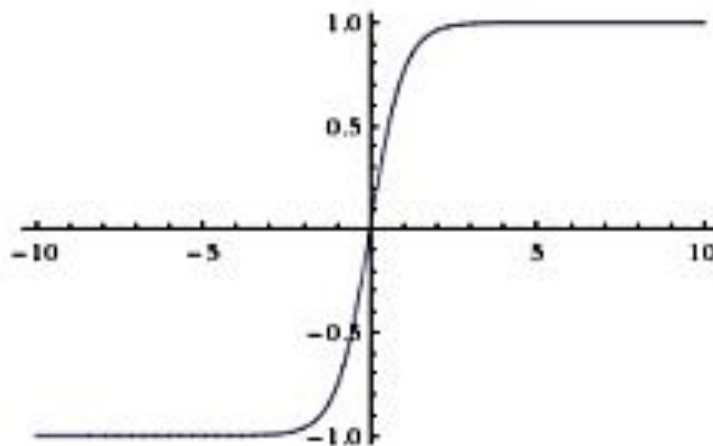
Функции активации

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



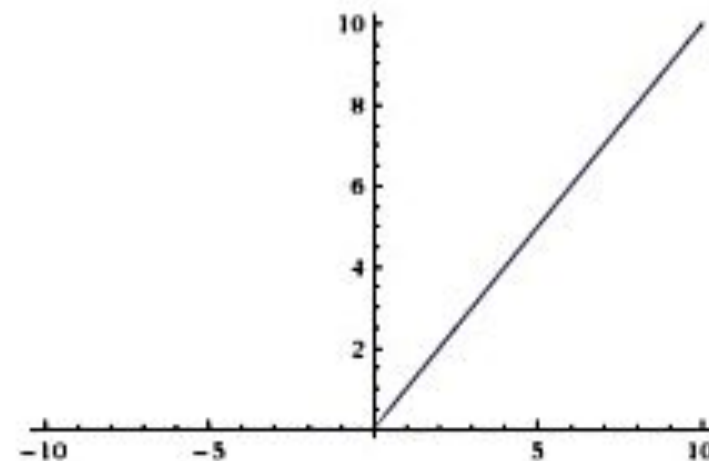
Sigmoid

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



tanh

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



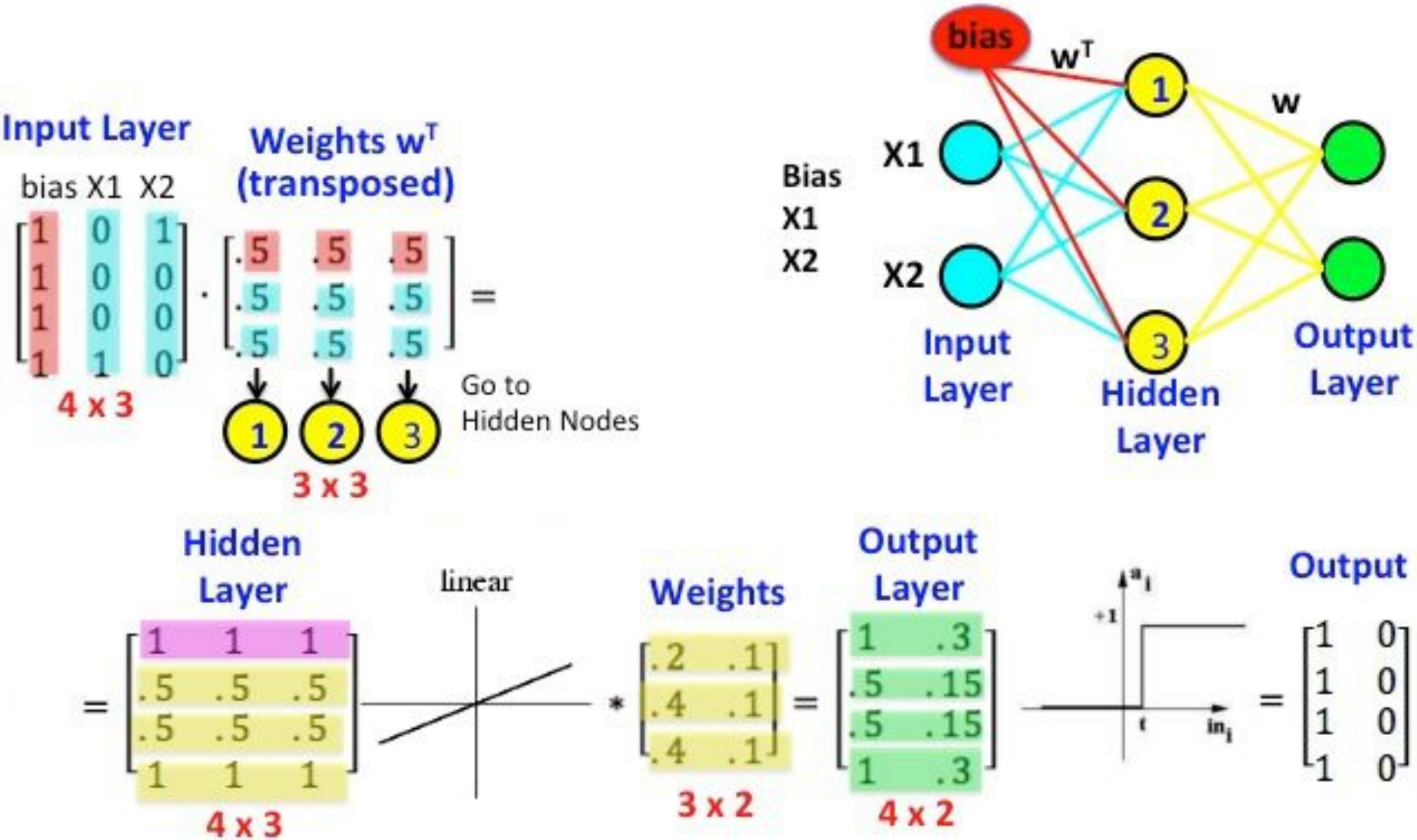
ReLU

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

ПОЛНОСВЯЗНАЯ НЕЙРОННАЯ СЕТЬ

Матричные операции



Граф вычислений

1

Процесс вычисления выходного значения нейронной сети можно представить в виде графа

2

Перемещаемся от входа нейронной сети к выходу, выполняя операции

3

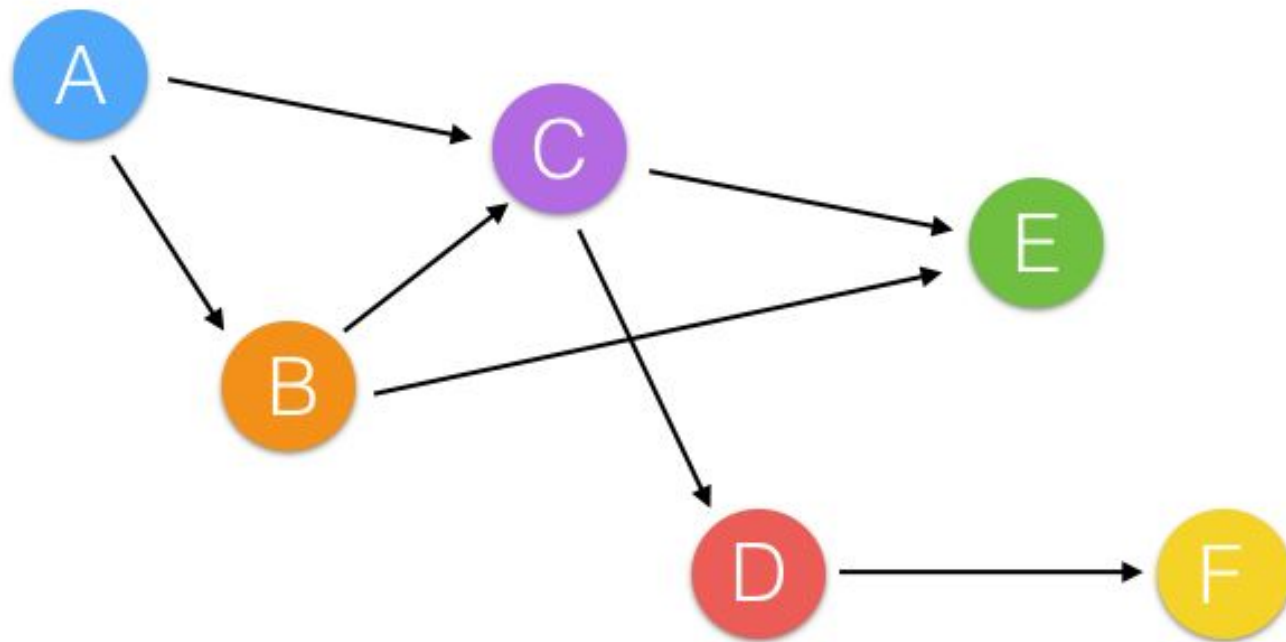
Структура сети может быть сложной

4

Перед запуском вычислений необходимо определить последовательность выполнения операций

ПОЛНОСВЯЗНАЯ НЕЙРОННАЯ СЕТЬ

Топологическая
сортировка





ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ

Функция потерь

1

Необходимо сформулировать оптимизационную задачу, т.е. выбрать функцию потерь

2

Функция потерь определяет величину штрафа в случае неверной классификации

3

Функция потерь должна быть дифференцируема по параметрам модели

4

Решение задачи сводится к подбору параметров модели при которых функция потерь будет минимальна

Бинарная классификация — Logistic Loss

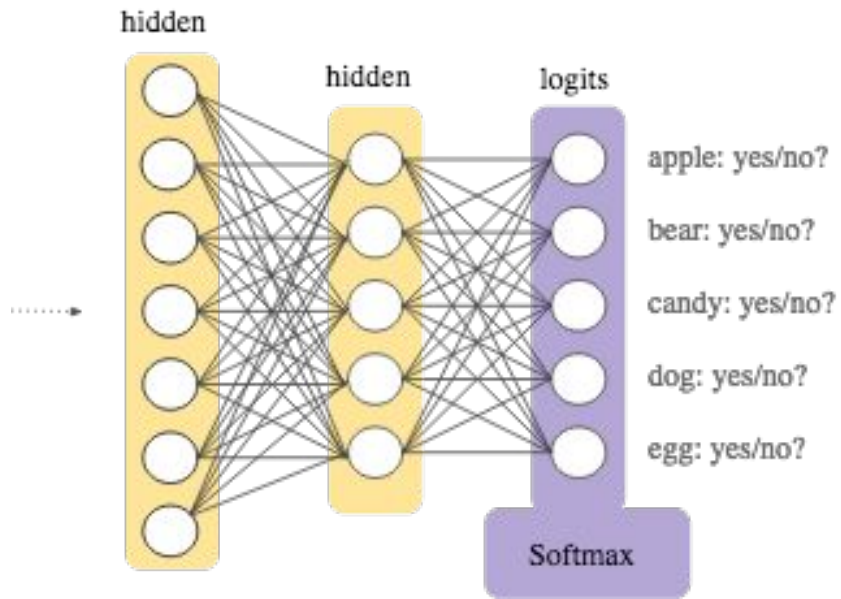
$$\tilde{y} = \frac{1}{1 + e^{-(hw^T + b)}}$$

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

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

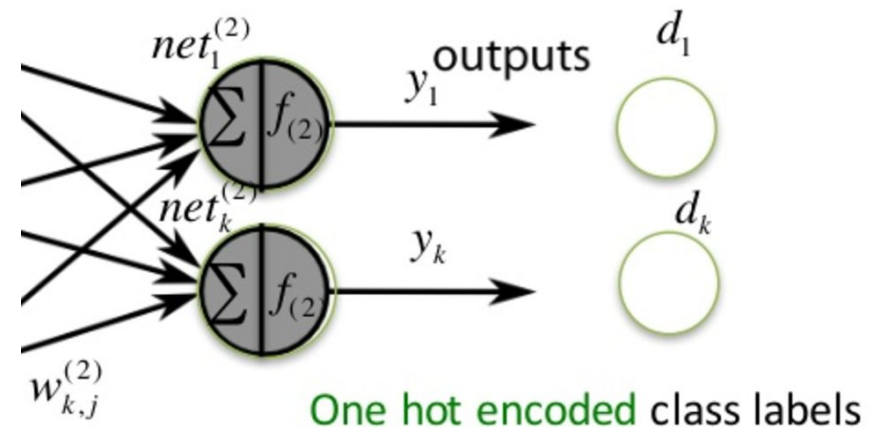
h вектор значений
w скрытого слоя
вектор весов выходного
b слоя
свободный член

Несколько классов — Softmax *Cross-Entropy*



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

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



A still from the movie Toy Story showing Woody and Buzz Lightyear. Woody is on the left, looking slightly concerned. Buzz is on the right, wearing his green and white space suit, with his right arm raised and fingers spread in a 'five' gesture. The background is a simple room with a door and a window.

MATH

MATH IS EVERYWHERE

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
- eta** 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

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

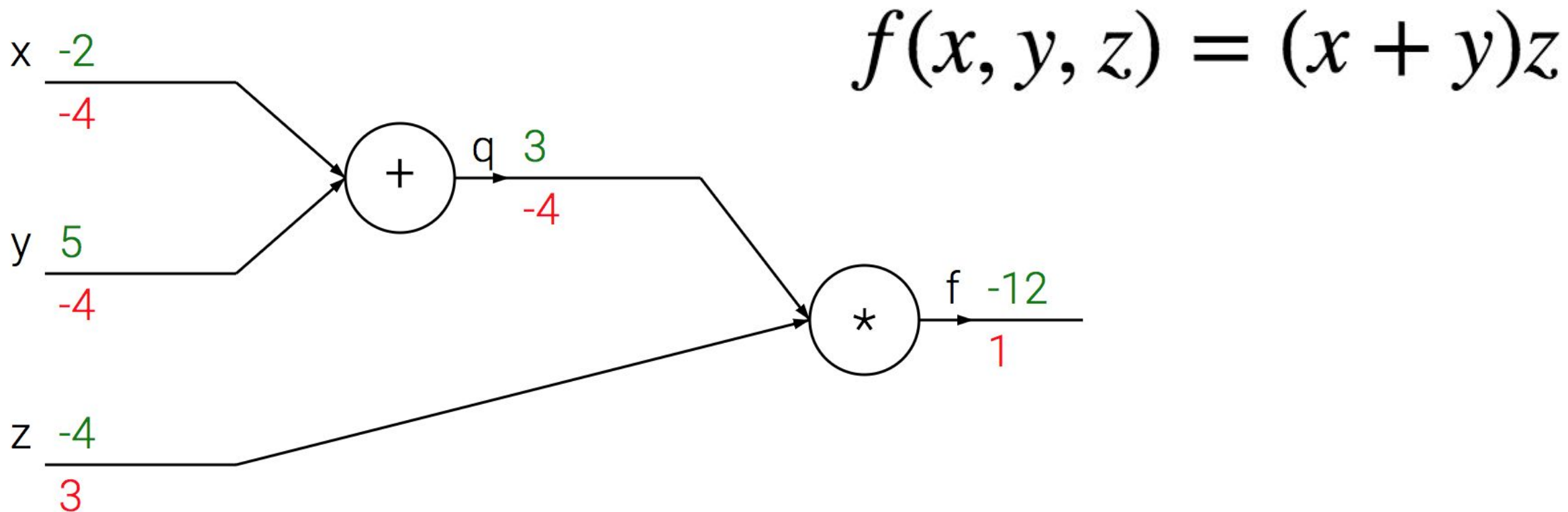
$$f = qz$$

$$q = x + y$$

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

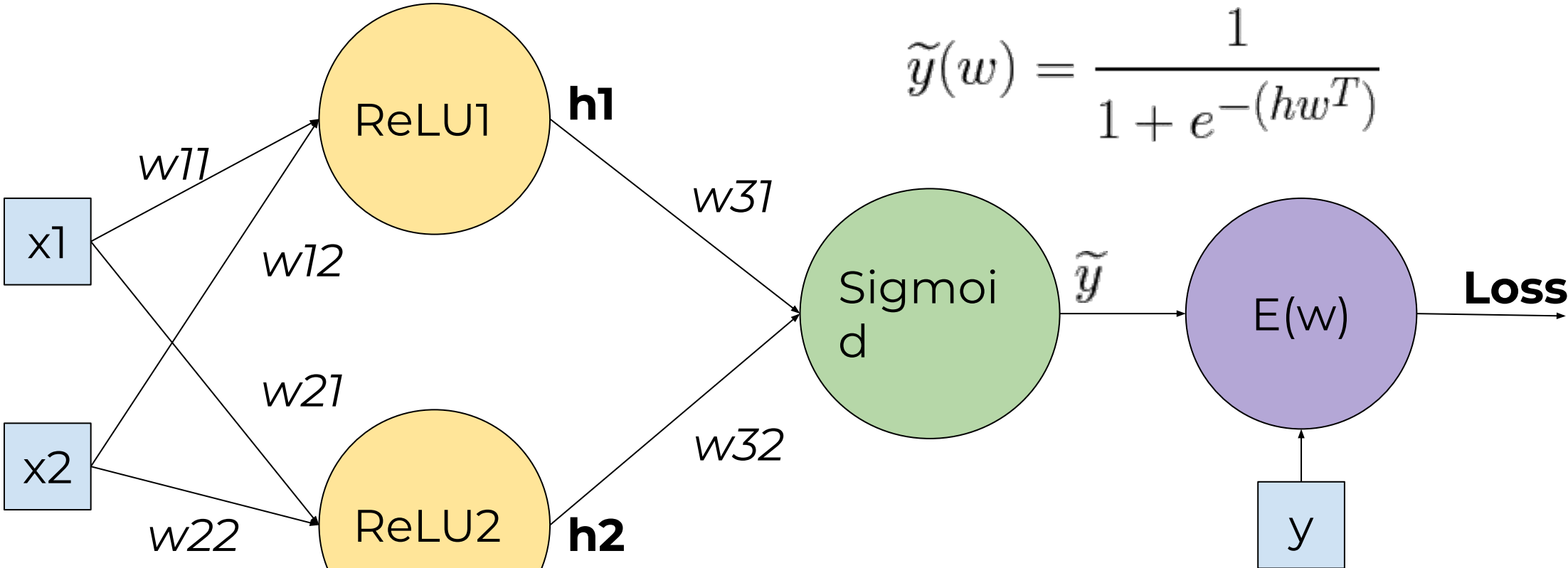
$$\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

Обратное распространение ошибки



ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ

Обратное
распространение
ошибки

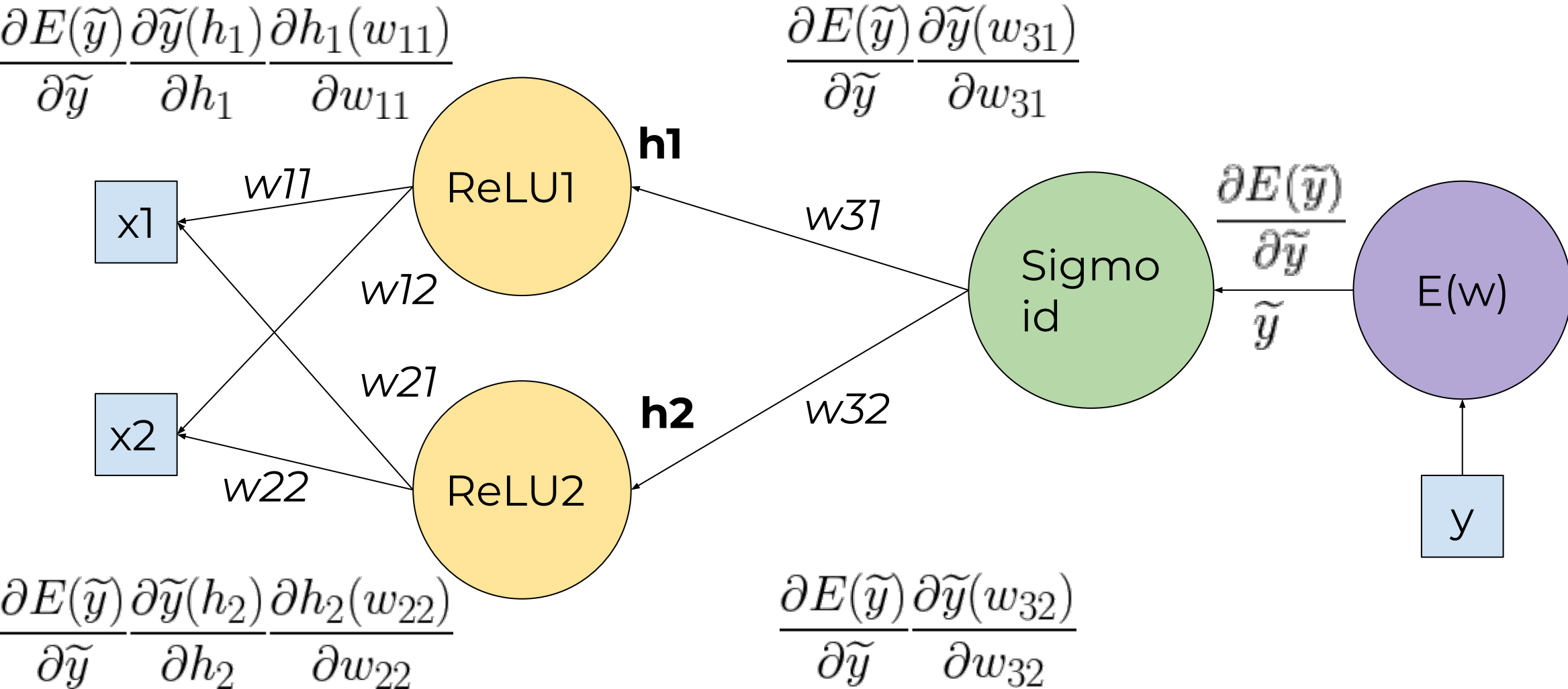


$$\tilde{y}(w) = \frac{1}{1 + e^{-(hw^T)}}$$

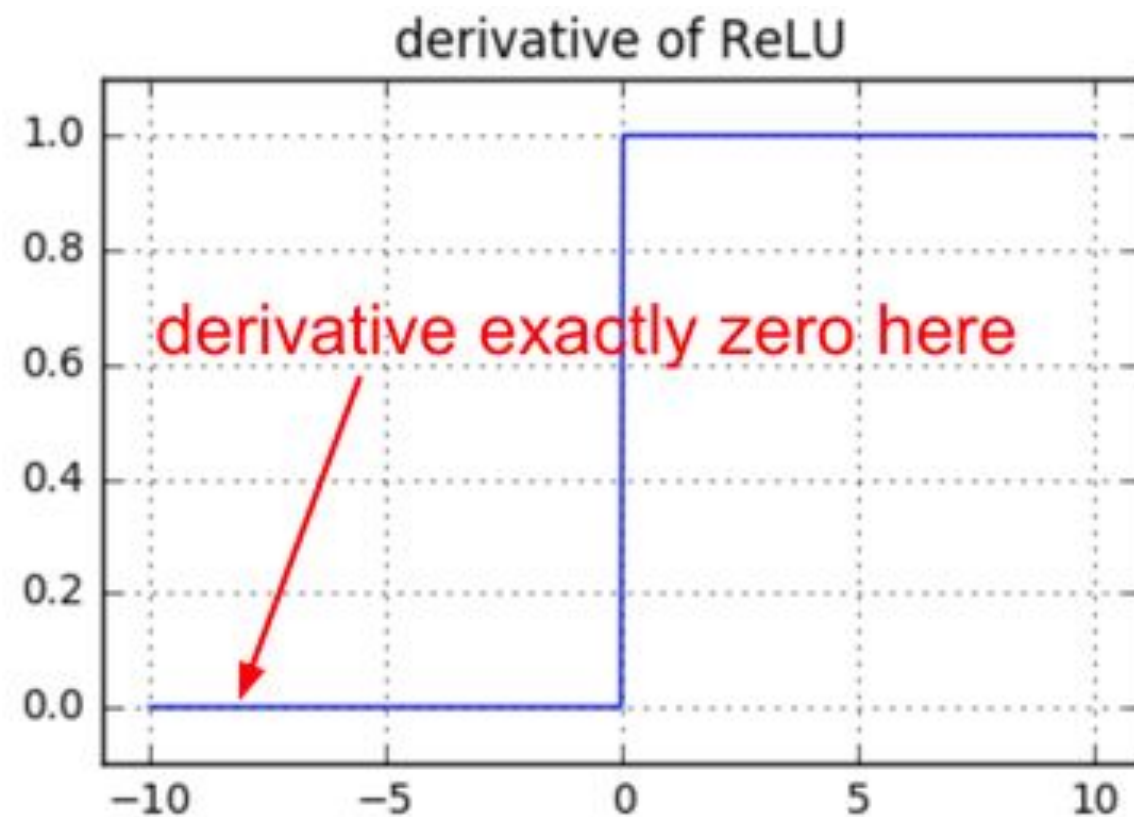
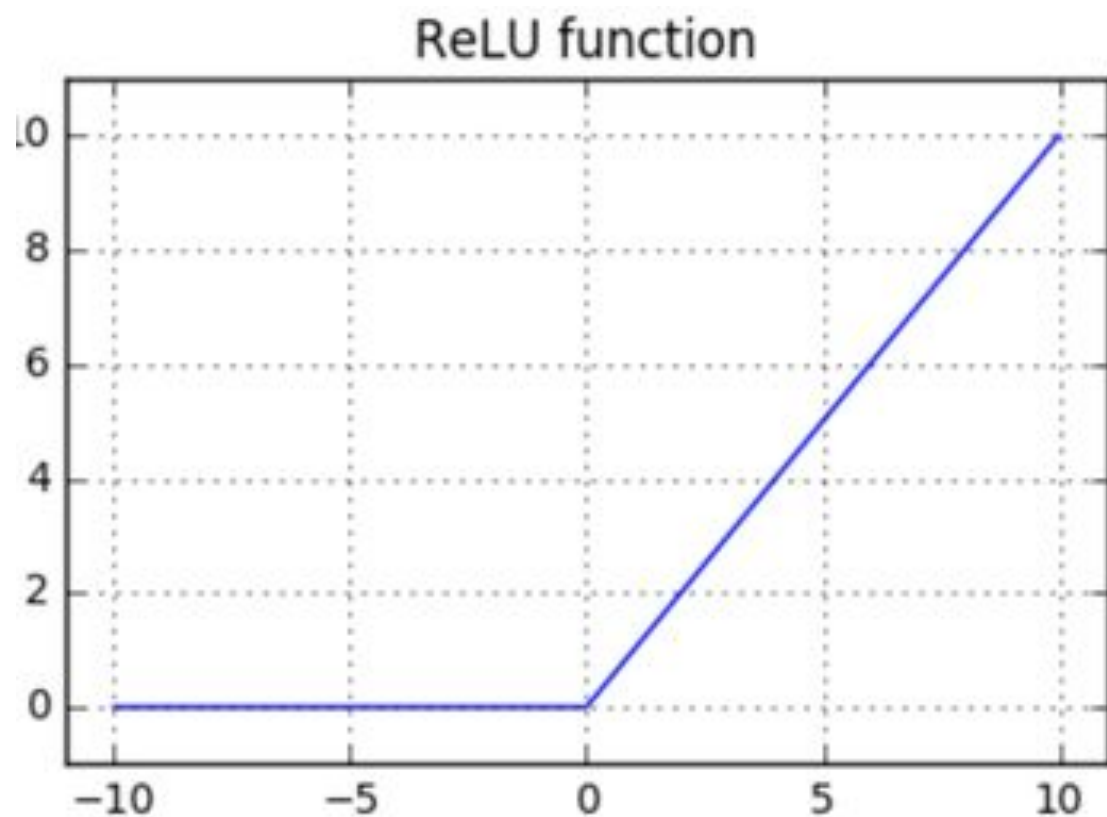
$$E_i(\tilde{y}) = -[y_i \log(\tilde{y}_i) + (1 - y_i) \log(1 - \tilde{y}_i)]$$

ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ

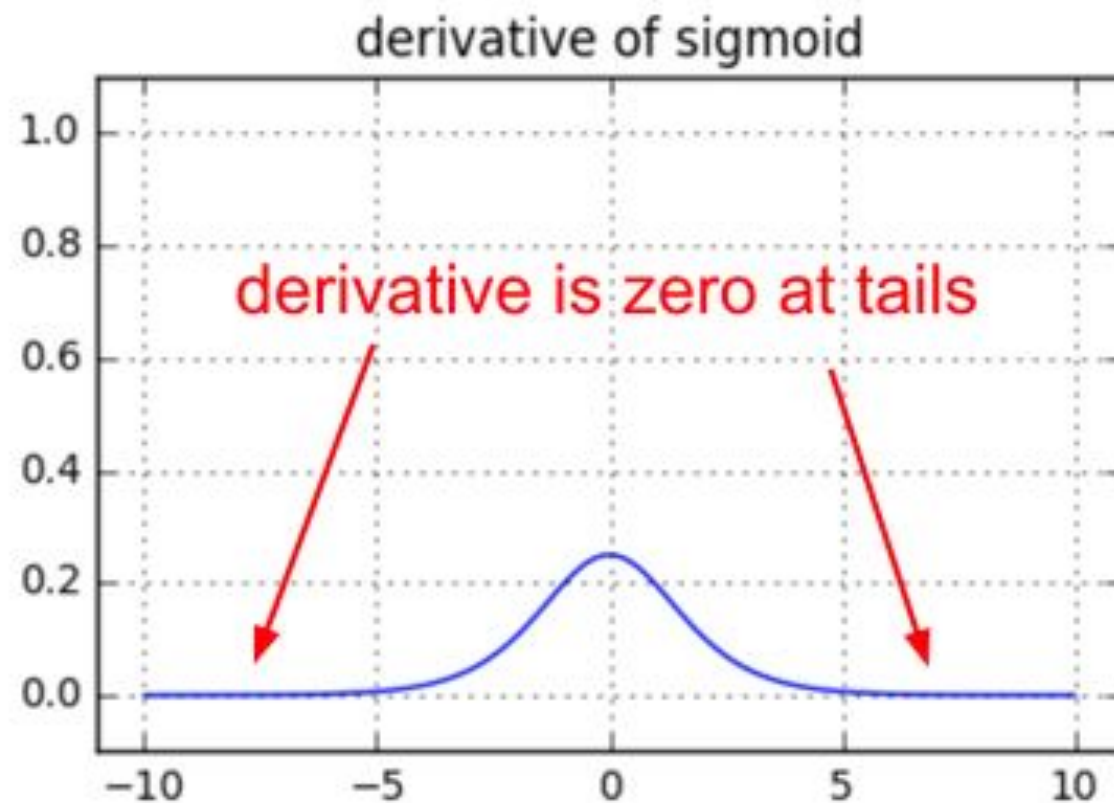
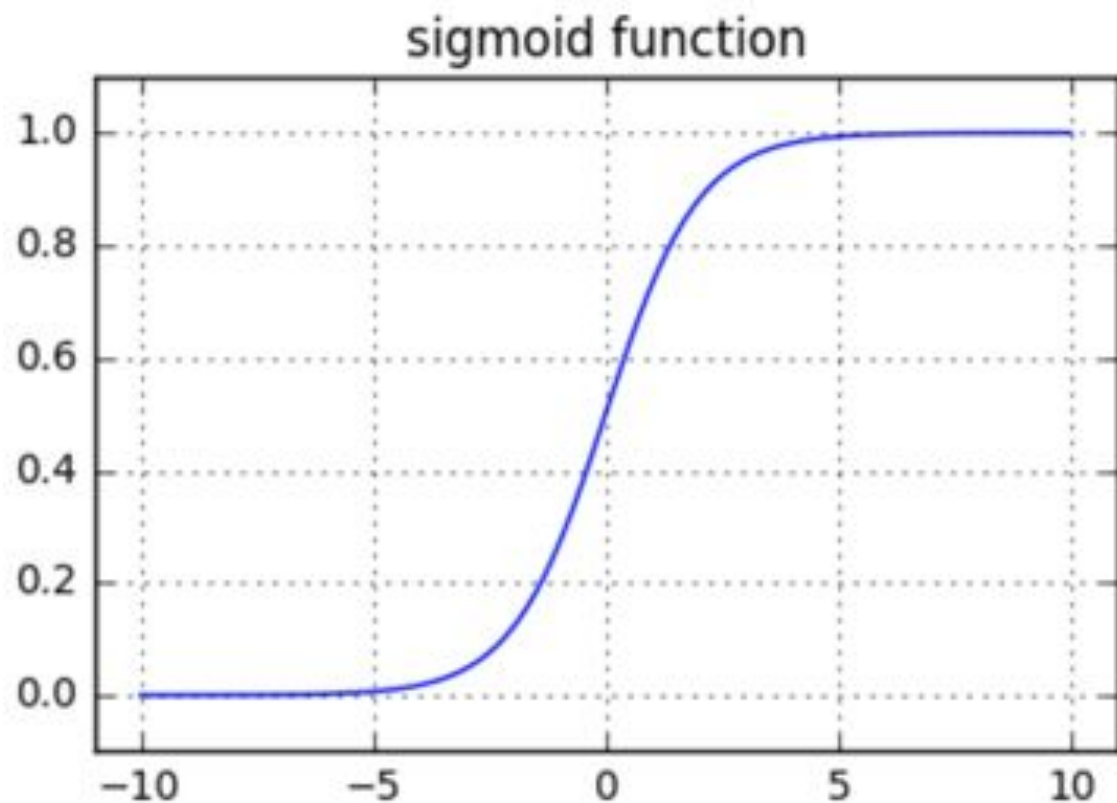
Обратное
распространение
ошибки













Исчезающие градиенты







Исчезающие градиенты



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		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$

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		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$

Name	Plot	Equation	Derivative
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric 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

[Keras Available Initializers](#)

ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ

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

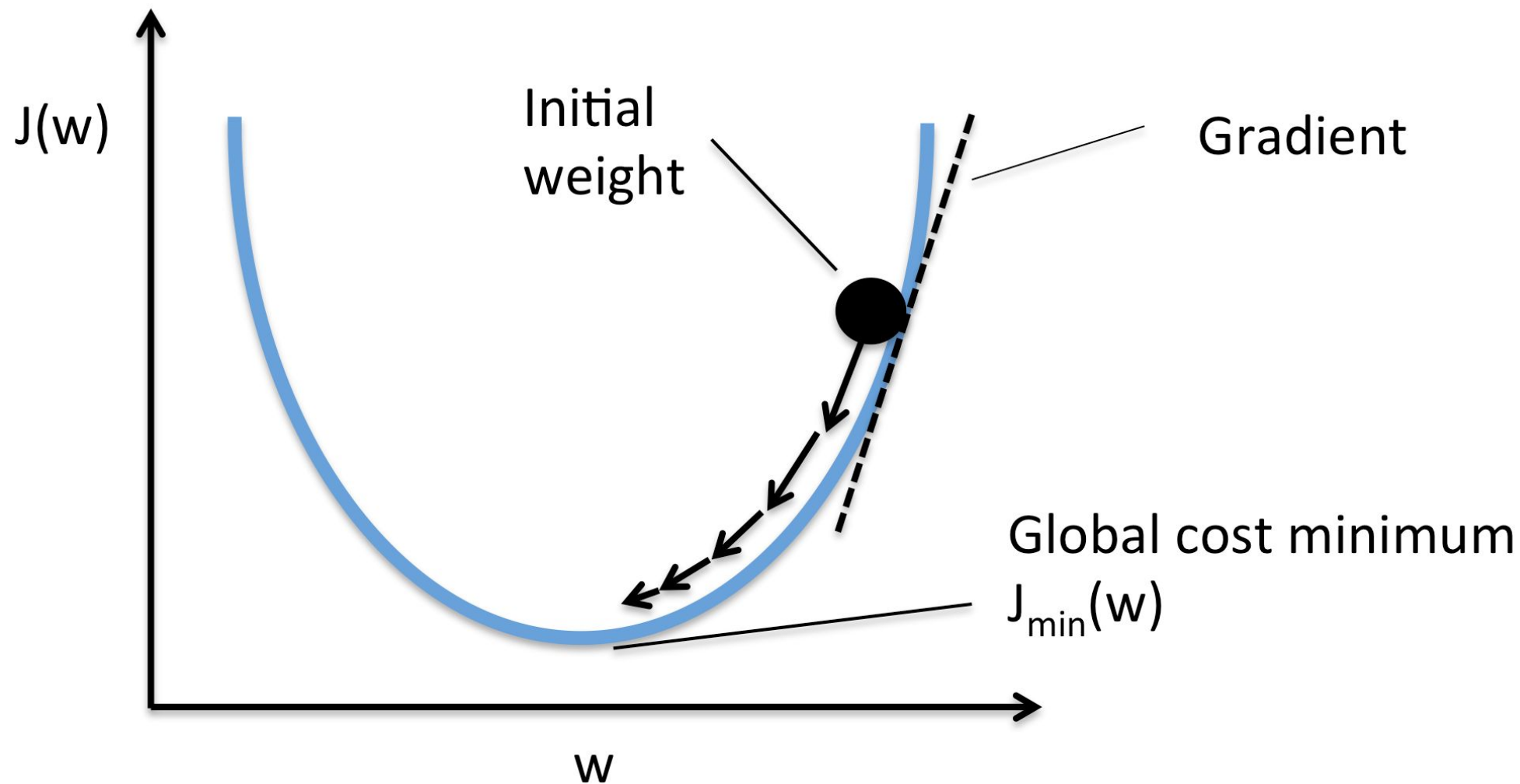
Оценку градиента
можно уточнить взяв
среднее оценок
по нескольким
примерам
из обучающей
выборки
batch

Градиентный спуск

Необходимо
чтобы функция
была выпуклой
*можно посчитать
производную
в любой точке*

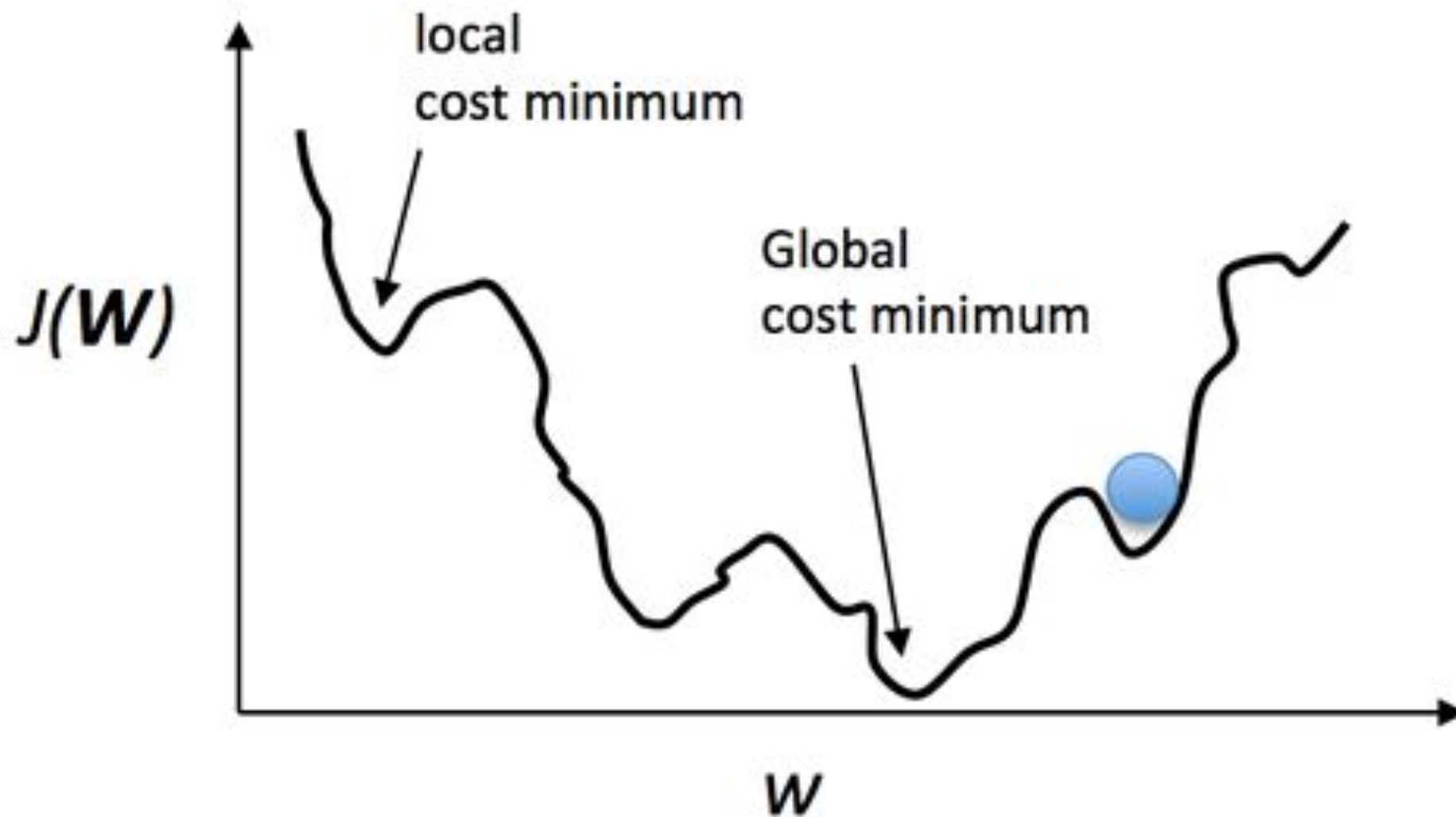
При большом
числе параметров
функция потерь
как правило имеет
локальные
минимумы

Седловые точки —
такие точки градиент
в которых равен нулю,
при этом точка
не является
ни минимумом,
ни максимумом

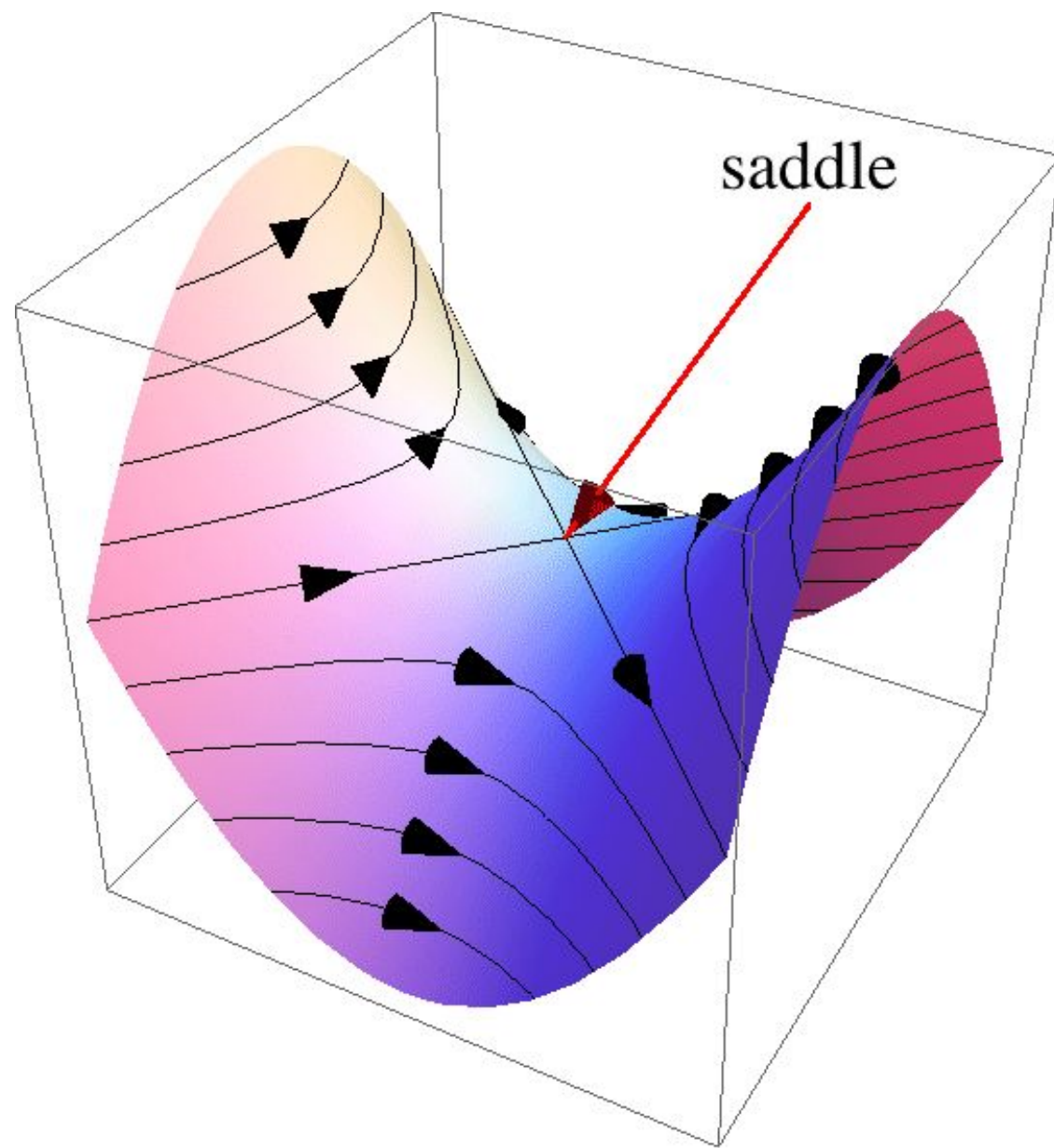


ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ

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

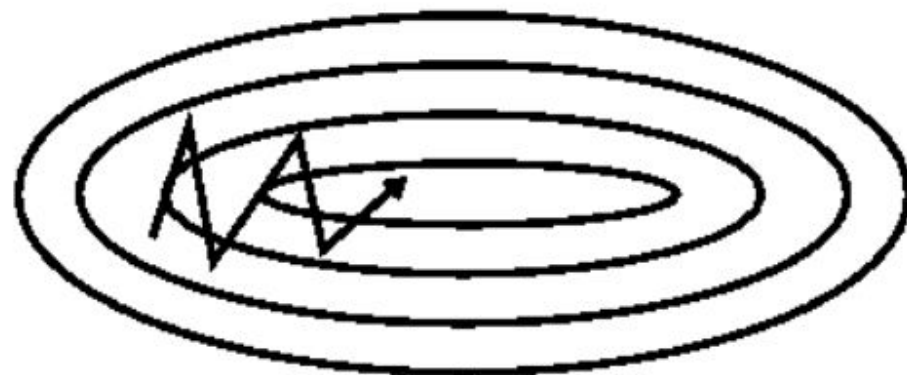
Оптимизация Момент

Momentum update

momentum
step

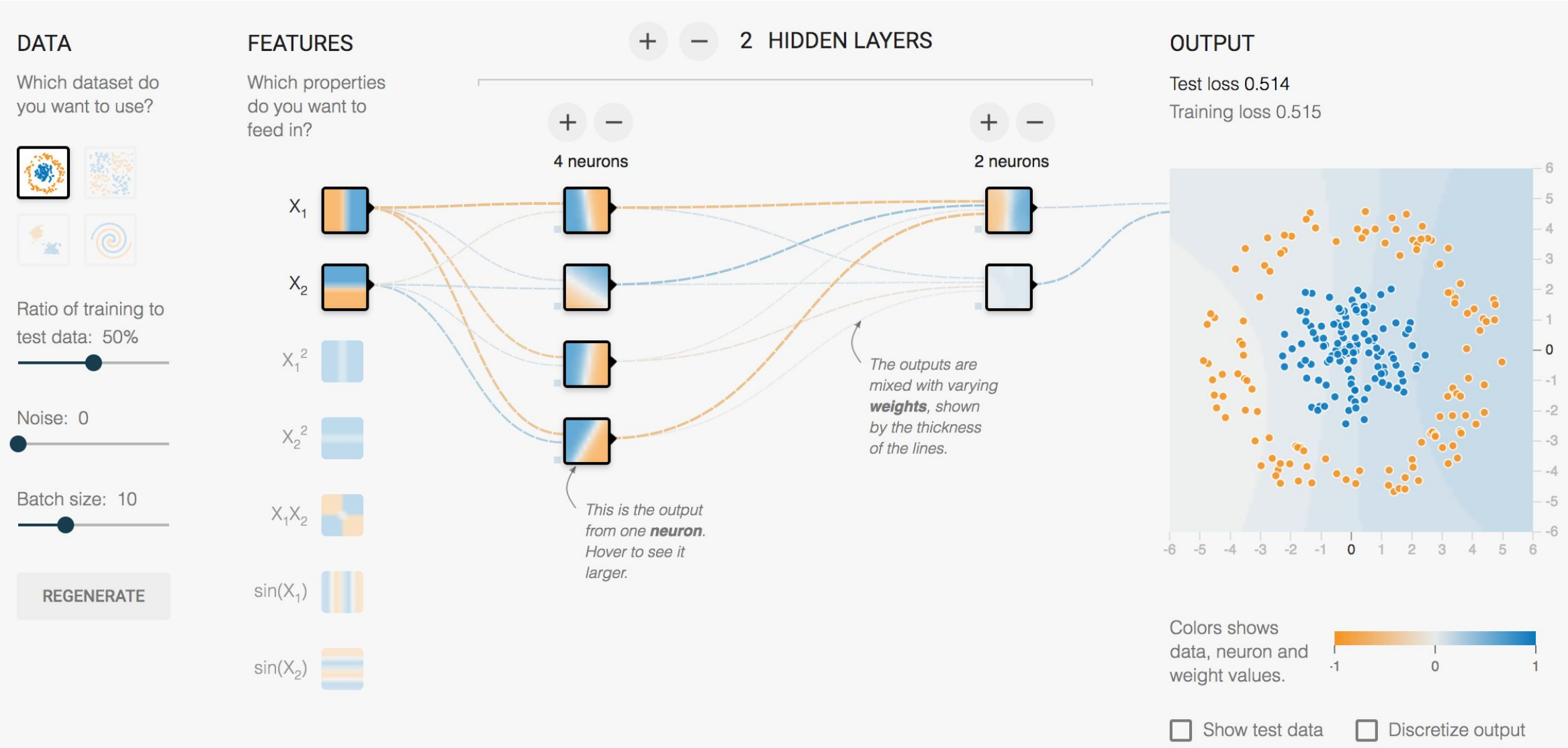
actual step

gradient
step



ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ

Tensorflow
Playground
playground.tensorflow.org/





РЕЗЮМЕ

РЕЗЮМЕ

1

Изучили
принципы работы
нейронной сети

2

Познакомились
с процессом
обучения
нейронной сети

3

Реализовали
решение задачи
классификации
MNIST с помощью
фреймворка Keras

Полезные материалы

1. [Neural Networks and Deep Learning](#)
2. [Deep Learning](#)
3. [CS231n Winter 2016](#)
4. [Yes you should understand backprop](#)
5. [An overview of gradient descent optimization algorithms](#)
6. [The Neural Network Zoo](#)
7. [Tensorflow Playground](#)
8. [Loss Functions](#)

СПАСИБО ЗА ВНИМАНИЕ