Московский государственный
технический университет
имени Н.Э. Баумана

Кафедра ИУ5
«Системы обработки информации
и управления»

Мастер-класс

Веб-приложение классификации изображений с помощью нейронной сети

Терехов Валерий Игоревич
Канев Антон Игоревич
кафедра ИУ5

lu5.bmstu.ru

Кафедра ИУ5

iu5.bmstu.ru

VK
Telegram
YouTube

Одна из крупнейших кафедр университета

150 бакалавров и 80 магистров каждый год

Системная и бизнес аналитика

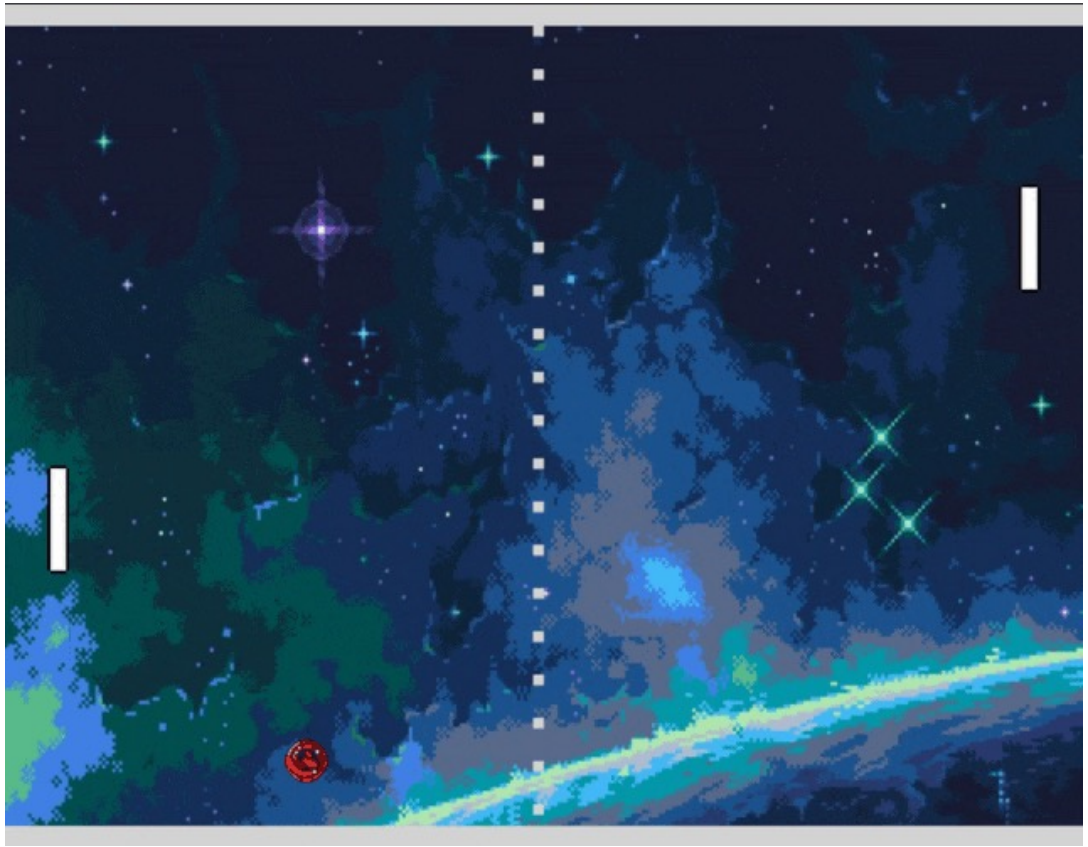
Аналитики данных и Data Science

Web разработка
DevOps/SRE

Конференция ИИАСУ

http://iiasu23.bmstu.ru/

Компьютерные игры



https://github.com/iu5git/ai-bot-games-in-js

- Пример интеллектуального агента ping-pong с обучением нейронной сети

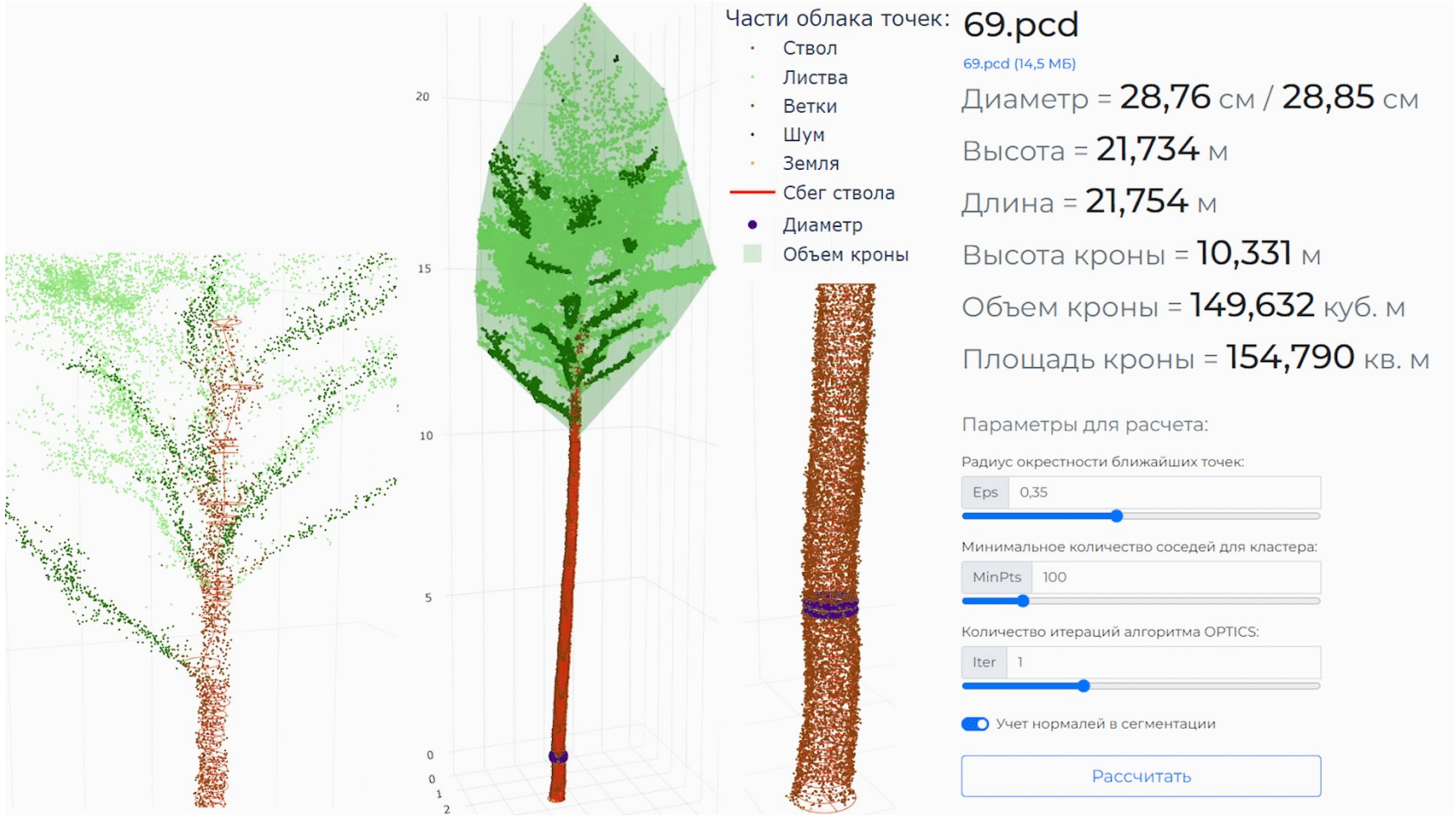
Обучение с учителем

Обучающие данные предварительно разметили

Пример - распознавание и локализация с помощью моделей yolo:

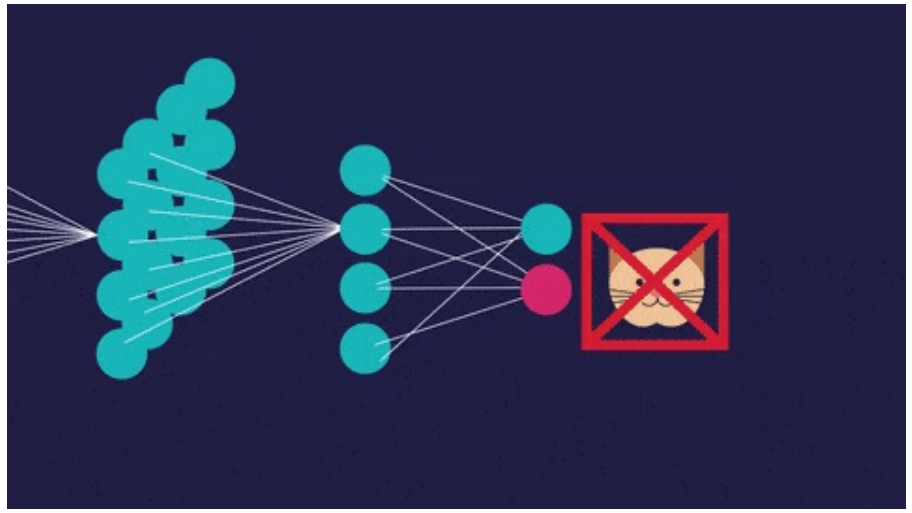
- Оружия
- Масок и тд

Обучение без учителя - Lidar

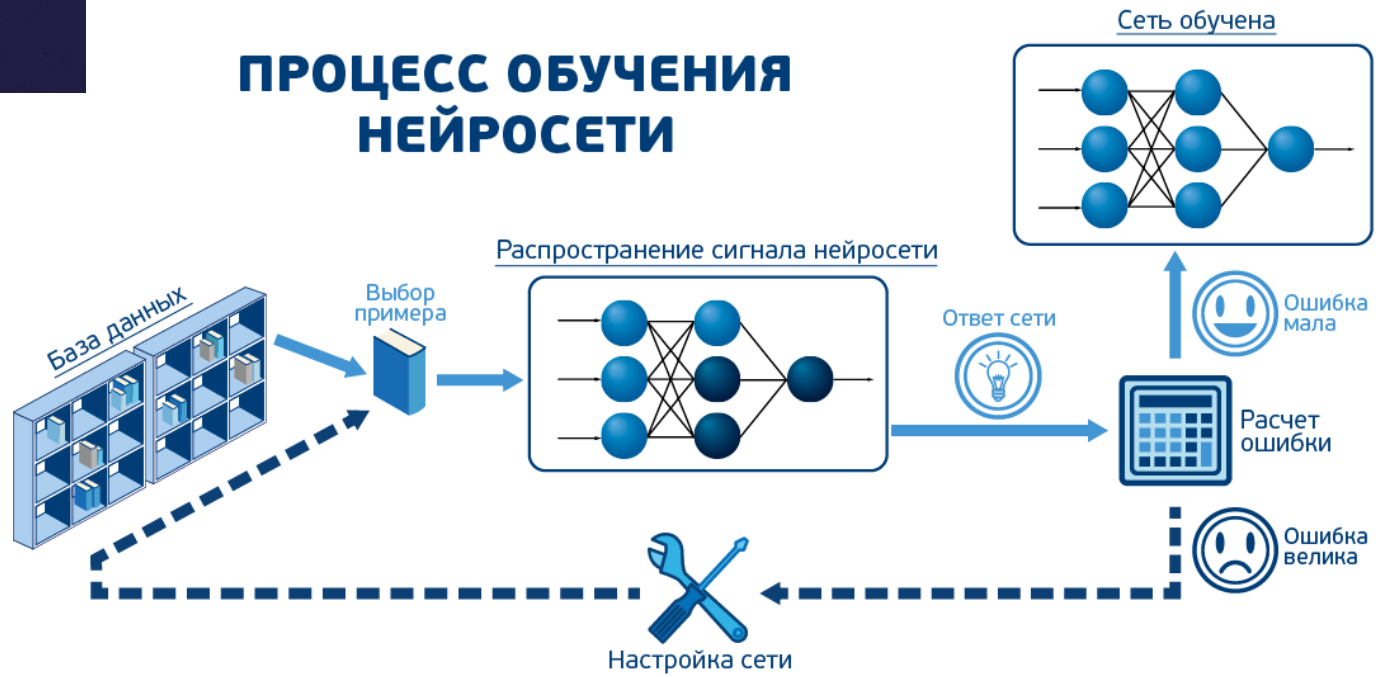


- Обучающие данные не размечены
- Объединяем в группы близкие точки

Обучение нейросети



- Произвести инференс на обучающем наборе
- Вычислить ошибку между спрогнозированными значениями и истинными значениями на обучающем наборе
- Определите вклад каждого нейрона в ошибку
- Изменить веса нейронной сети для минимизации ошибки



Тестовая и обучающая выборки

- Входные данные и метки
- Тестовая и обучающая части набора данных

```
with open('cifar-100-python/train', 'rb') as f:
        data_train = pickle.load(f, encoding='latin1’)
with open('cifar-100-python/test', 'rb') as f:
        data_test = pickle.load(f, encoding='latin1')
```

Метрики для классификации

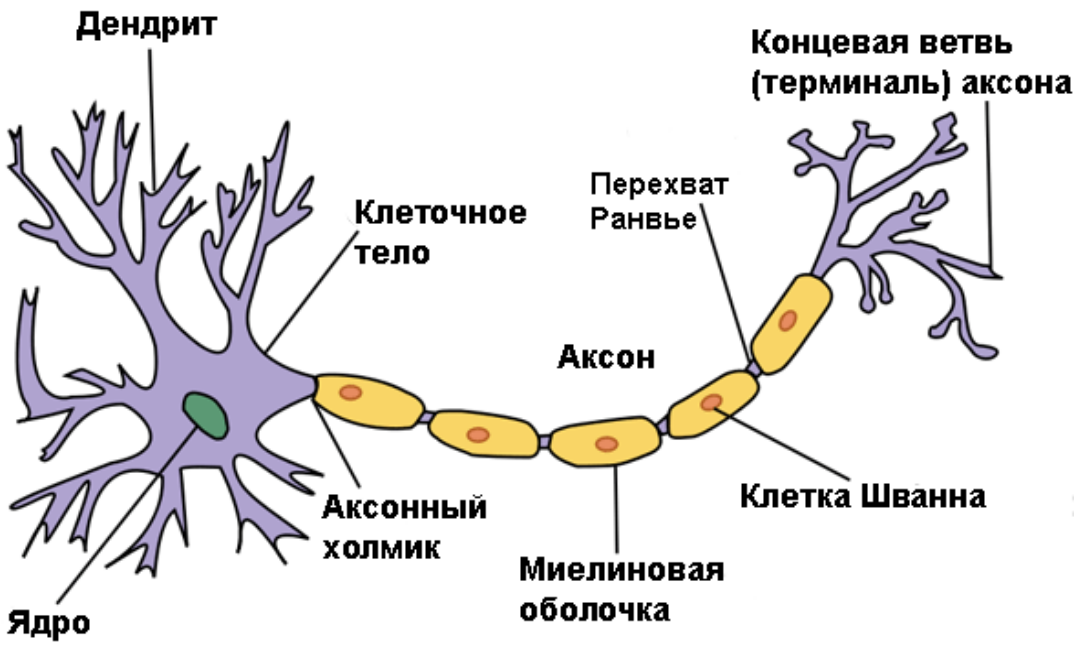
Прогноз: + − − + − +

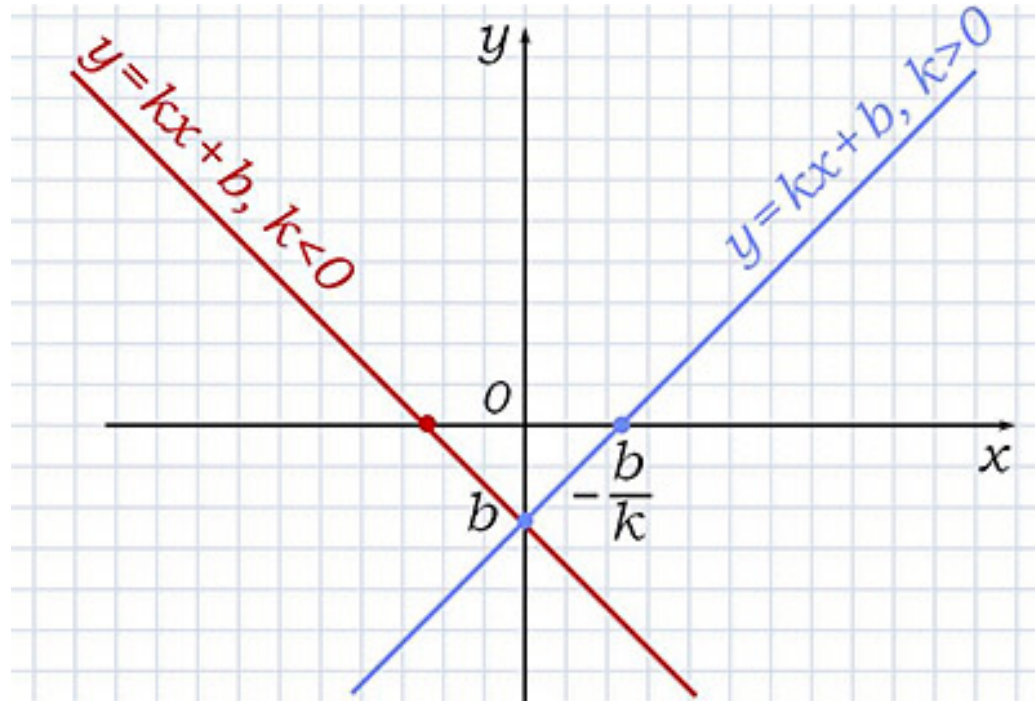
Картинка:

Истинно Положительное (TP)	**Истинно** Отрицательное (TN)	Ложно Отрицательное (FP)	Ложно Положительное (FN)

Precision = TP / (TP + FP) — не открыть телефон другому
Recall = TP / (TP + FN) — найти болезнь
Accuracy = (TP + TN) / (количество ответов) — общий % правильных

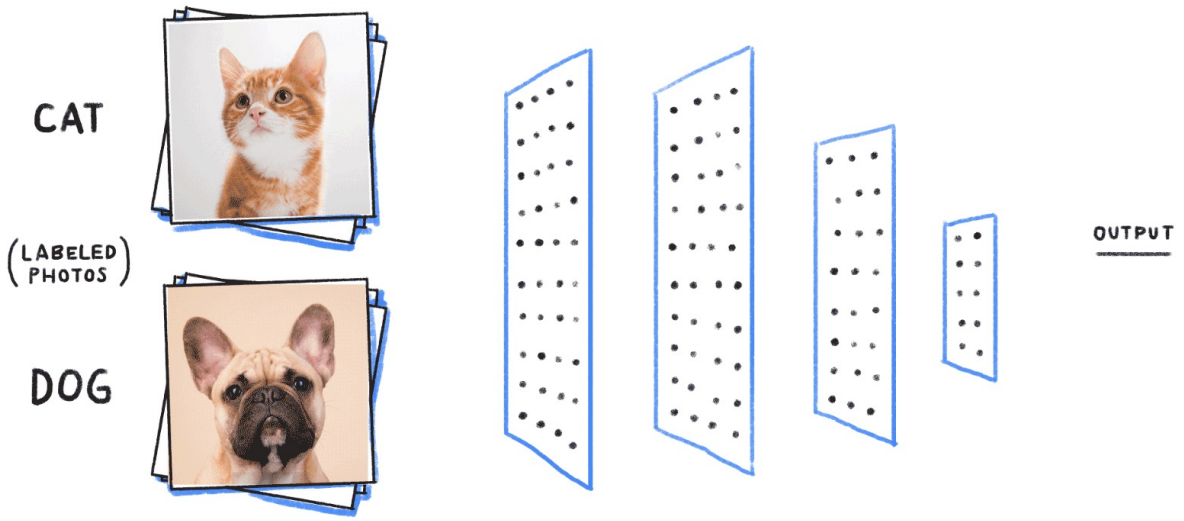
Биология и математика

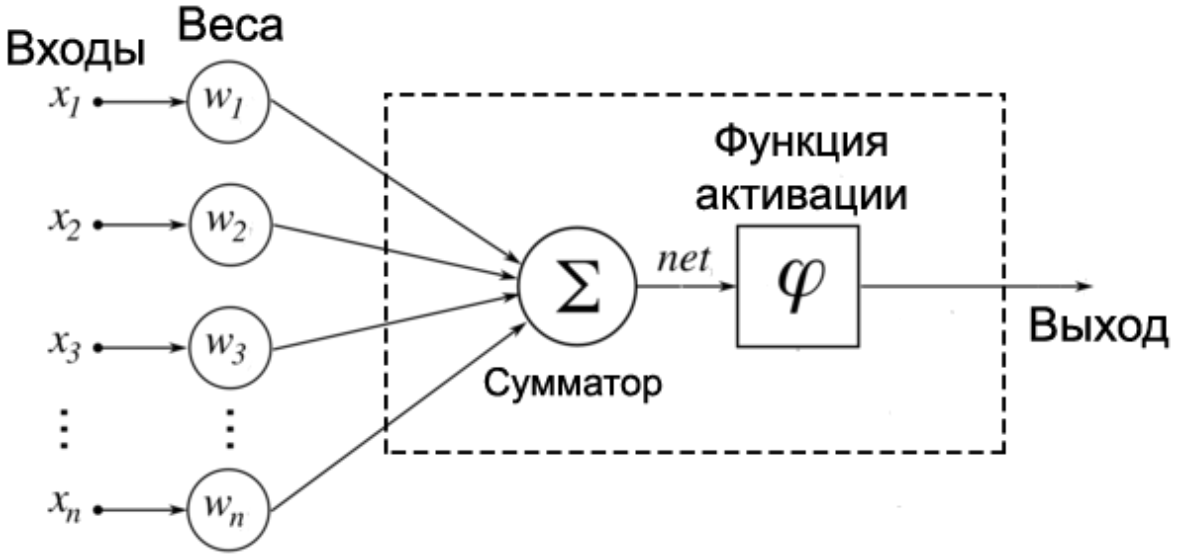




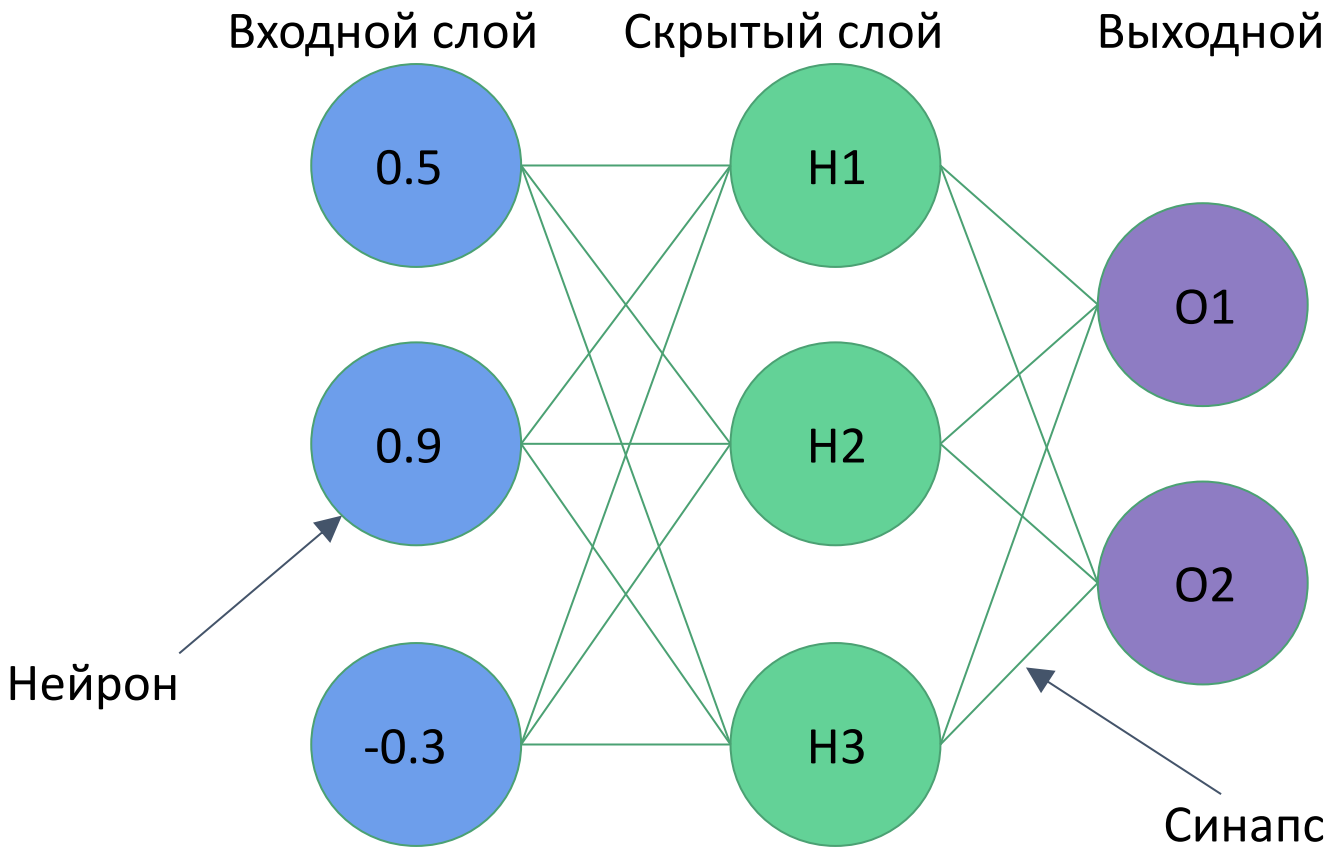
Нейронная сеть

- Сеть состоит из слоев нейронов
- Каждый нейрон – сумматор
- Обучение – вычисление весов w нейрона
- После каждого нейрона – активационная функция, без нее все превращается в линейное уравнение





Вычисление прогноза (инференс)



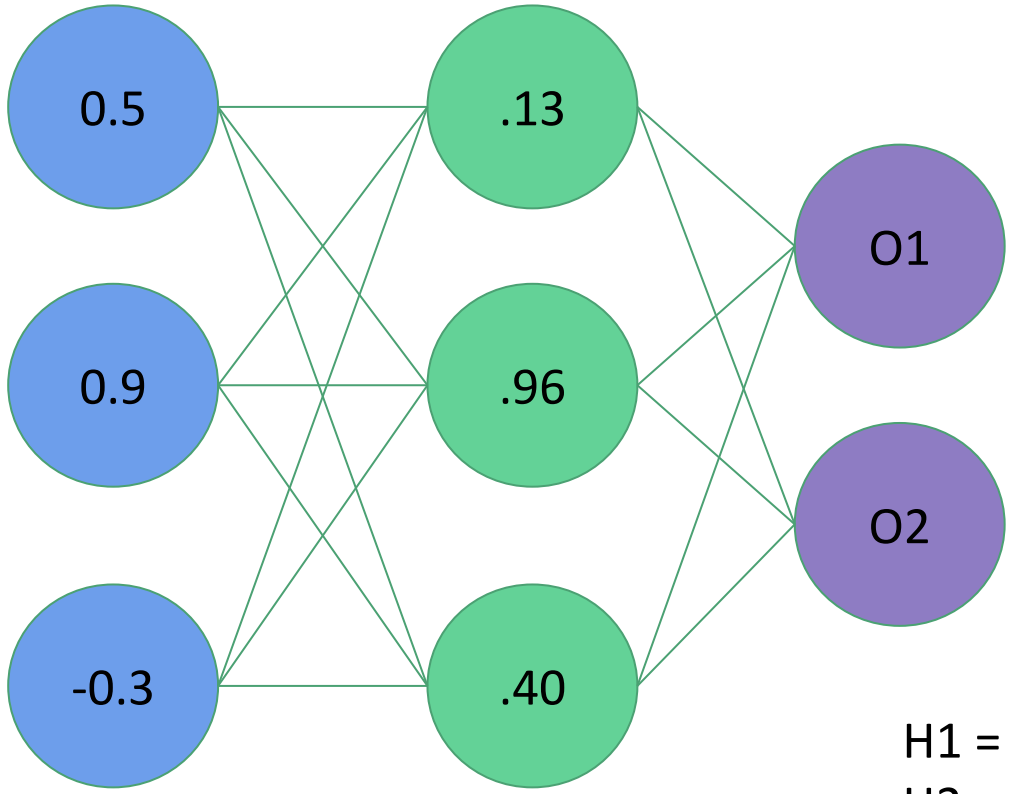
Веса – коэффициенты на связях между нейронами

Синие нейроны – наши входные данные

Веса H1 = (1.0, -2.0, 2.0)
Веса H2 = (2.0, 1.0, -4.0)
Веса H3 = (1.0, -1.0, 0.0)

Веса O1 = (-3.0, 1.0, -3.0)
Веса O2 = (0.0, 1.0, 2.0)

Вычисление прогноза (инференс)



Веса H1 = (1.0, -2.0, 2.0)
Веса H2 = (2.0, 1.0, -4.0)
Веса H3 = (1.0, -1.0, 0.0)

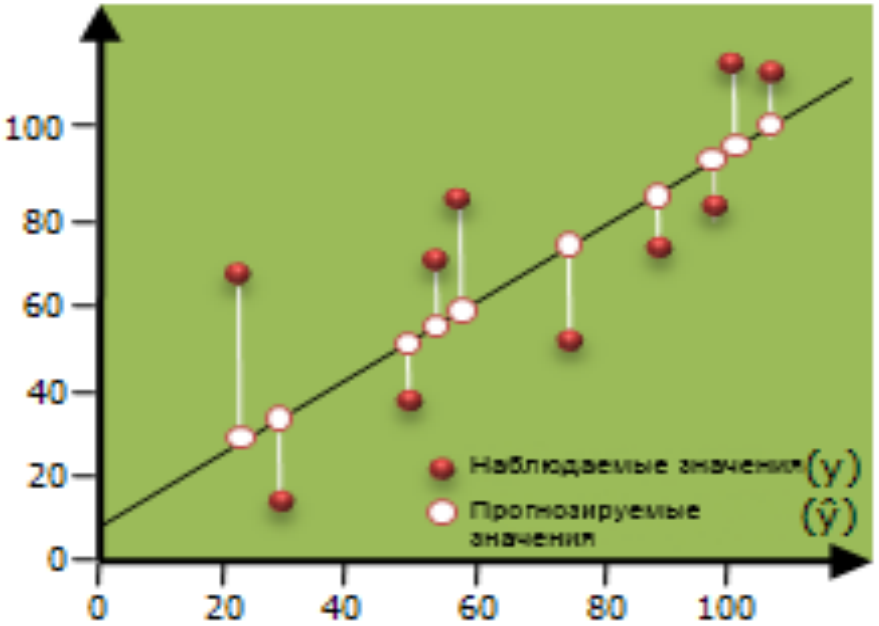
Веса O1 = (-3.0, 1.0, -3.0)
Веса O2 = (0.0, 1.0, 2.0)

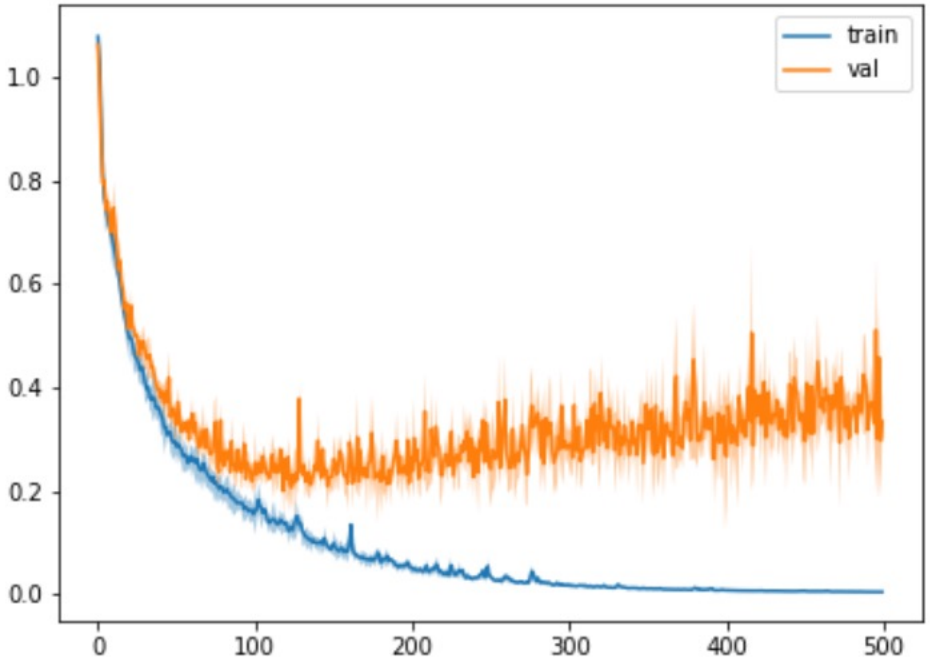
Используем входы и веса нейронов, чтобы посчитать следующие нейроны:

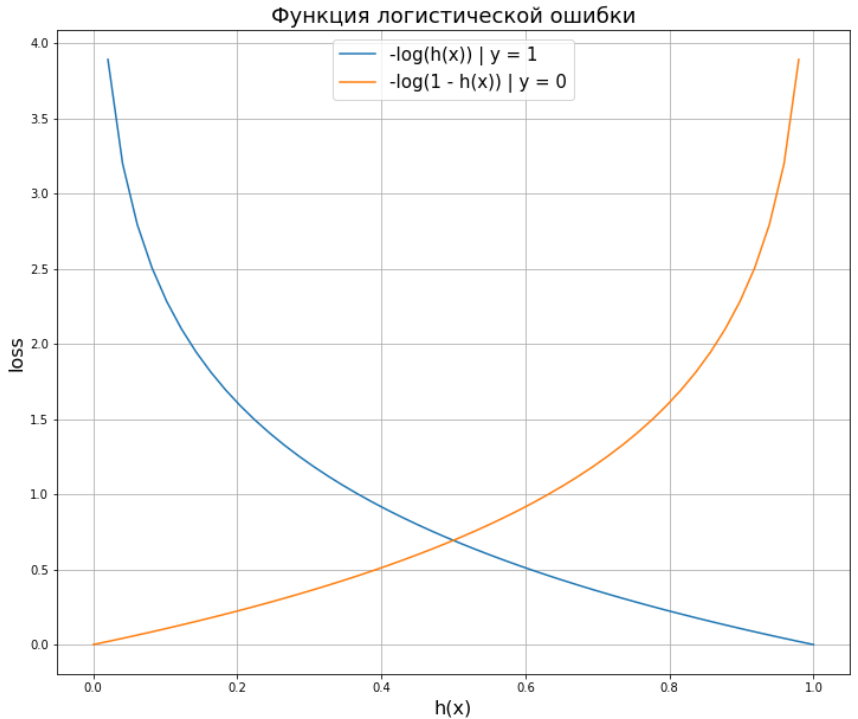
H1 = S(0.5 * 1.0 + 0.9 * -2.0 + -0.3 * 2.0) = S(-1.9) = .13
H2 = S(0.5 * 2.0 + 0.9 * 1.0 + -0.3 * -4.0) = S(3.1) = .96
H3 = S(0.5 * 1.0 + 0.9 * -1.0 + -0.3 * 0.0) = S(-0.4) = .40

Функции потерь

- Нам нужен минимум функции потерь (**ошибки**)
- Метод наименьших квадратов (сверху)
- Кроссэнтропия (справа)
- График функции потерь от количества шагов (снизу)





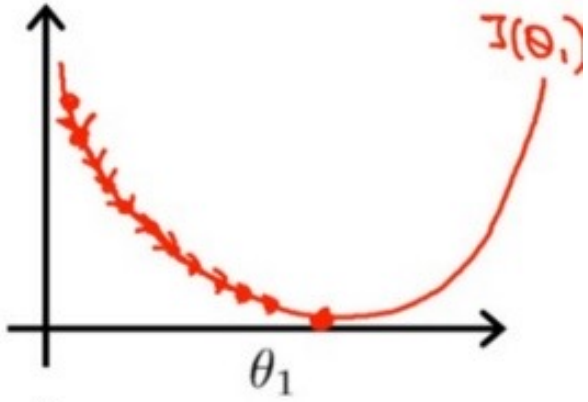


Градиентный спуск

- Параметры (веса) сети меняются при обучении
- Гиперпараметры – нет. Управляем обучением

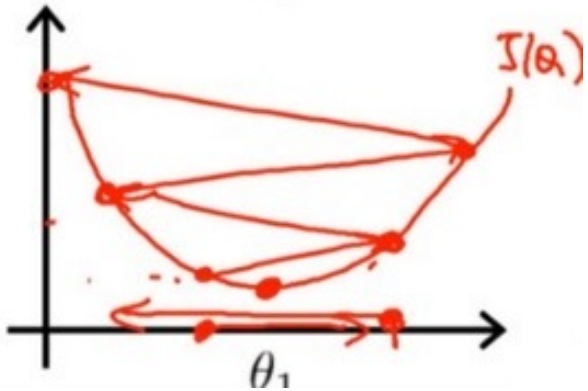
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

Если α слишком маленькая, то градиентный спуск может быть слишком медленным.



- Скорость обучения позволяет регулировать, насколько длинными будут наши шаги

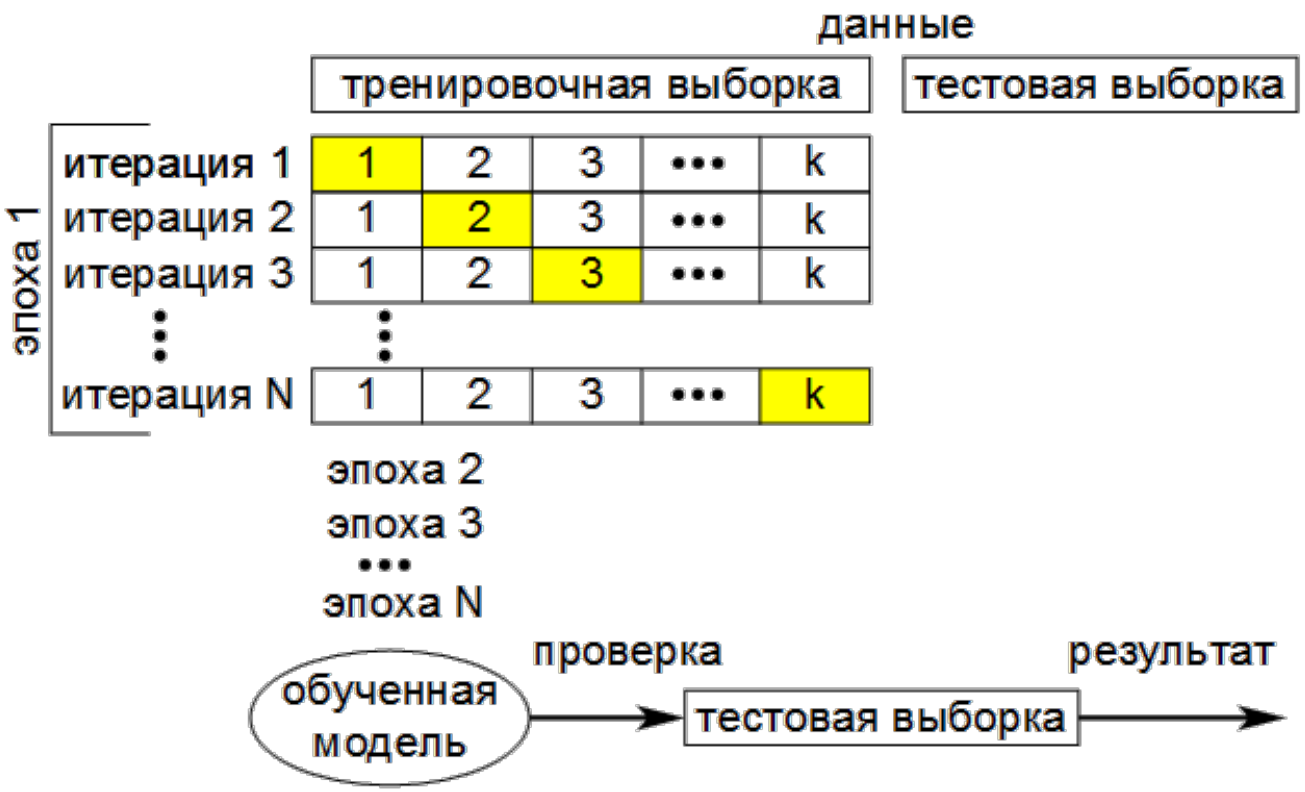
Если α слишком большая, то градиентный спуск может не попасть в точку минимума. Кроме того, сходимость может быть не достигнута.



 # lr - шаг обучения. Данный параметр можно изменять.
optimizer = optim.SGD(model.parameters(), lr=0.005)

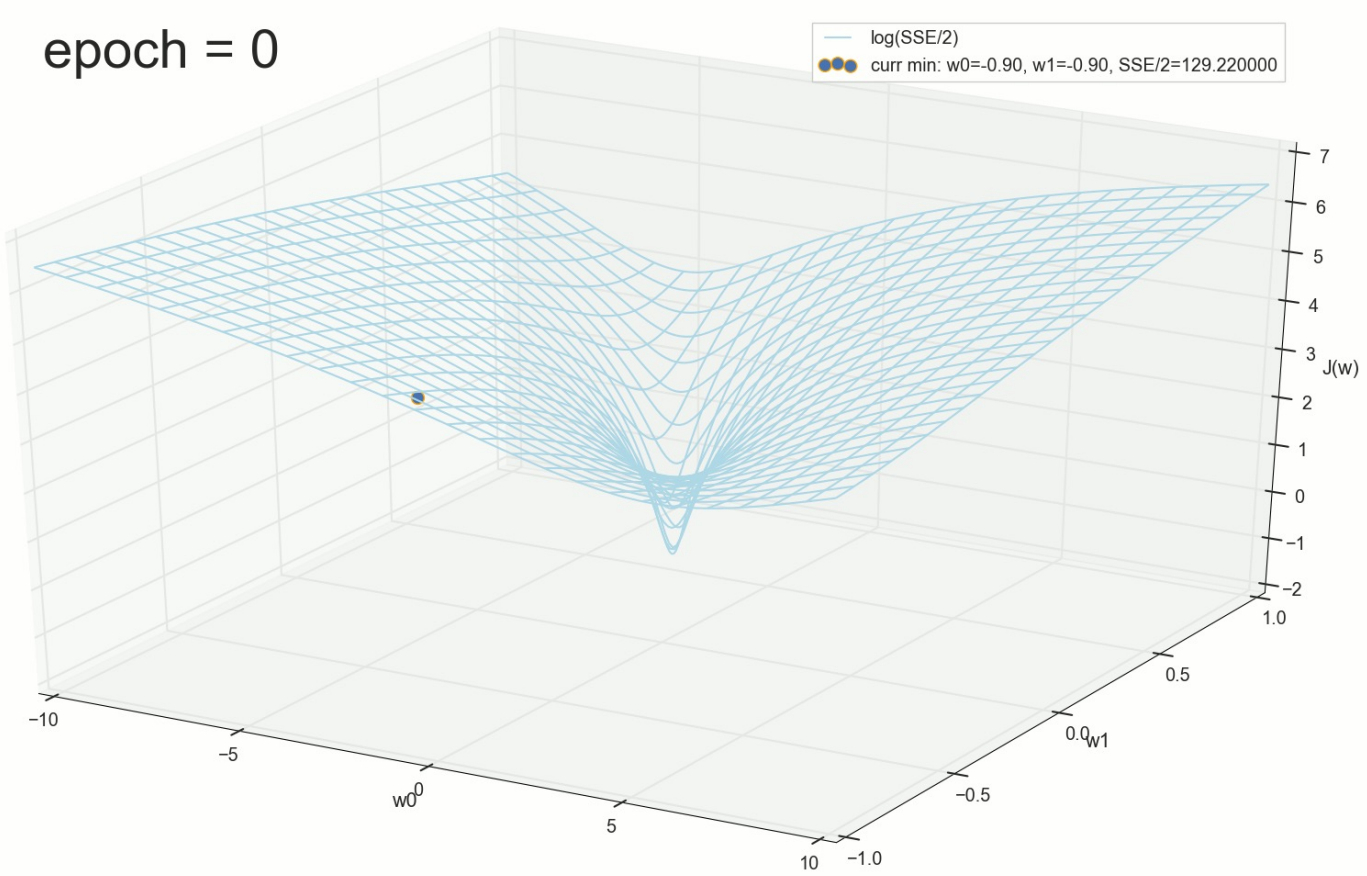
Итерации и эпохи

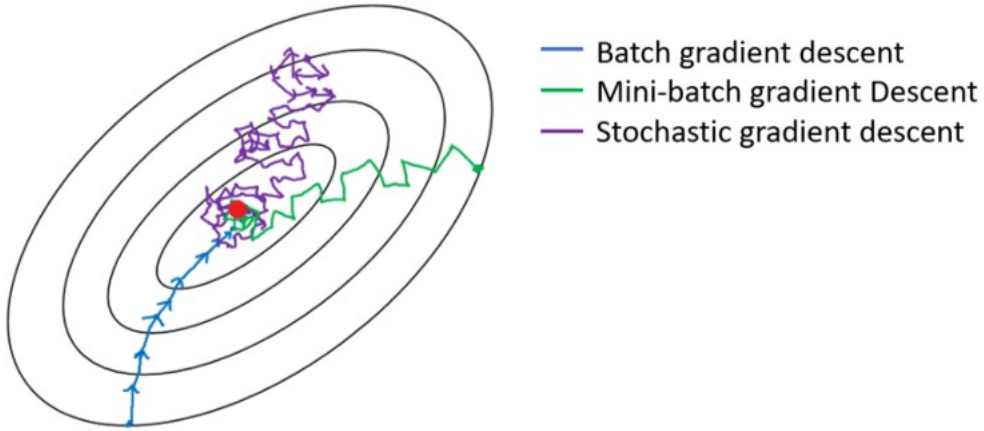
- Итерации – один шаг обучения
- Эпоха – обход всех экземпляров набора данных

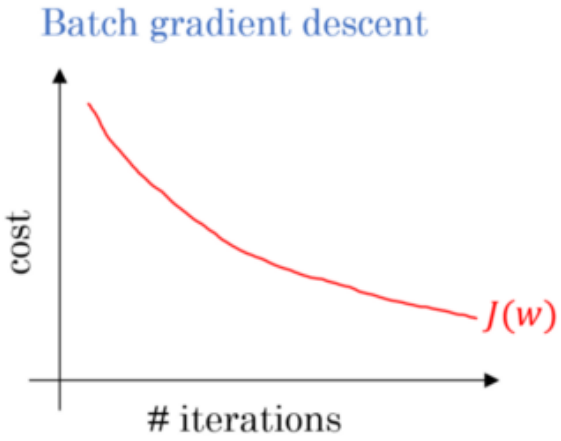


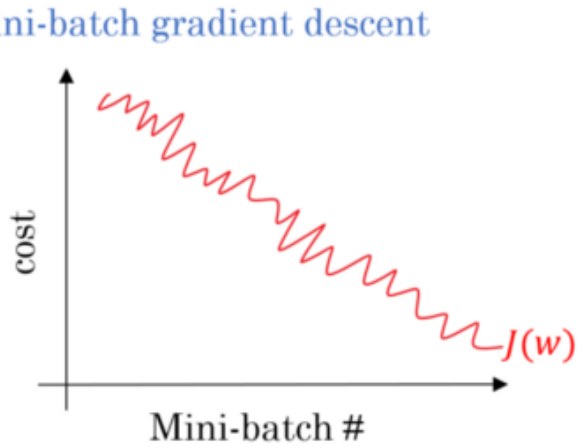
Батчи

- Функцию потерь считаем по одному примеру, но потом их складываем в одну
- Обучать на всем датасете – долго. Каждый раз берем небольшую порцию данных
- Но обучение становится хаотичнее

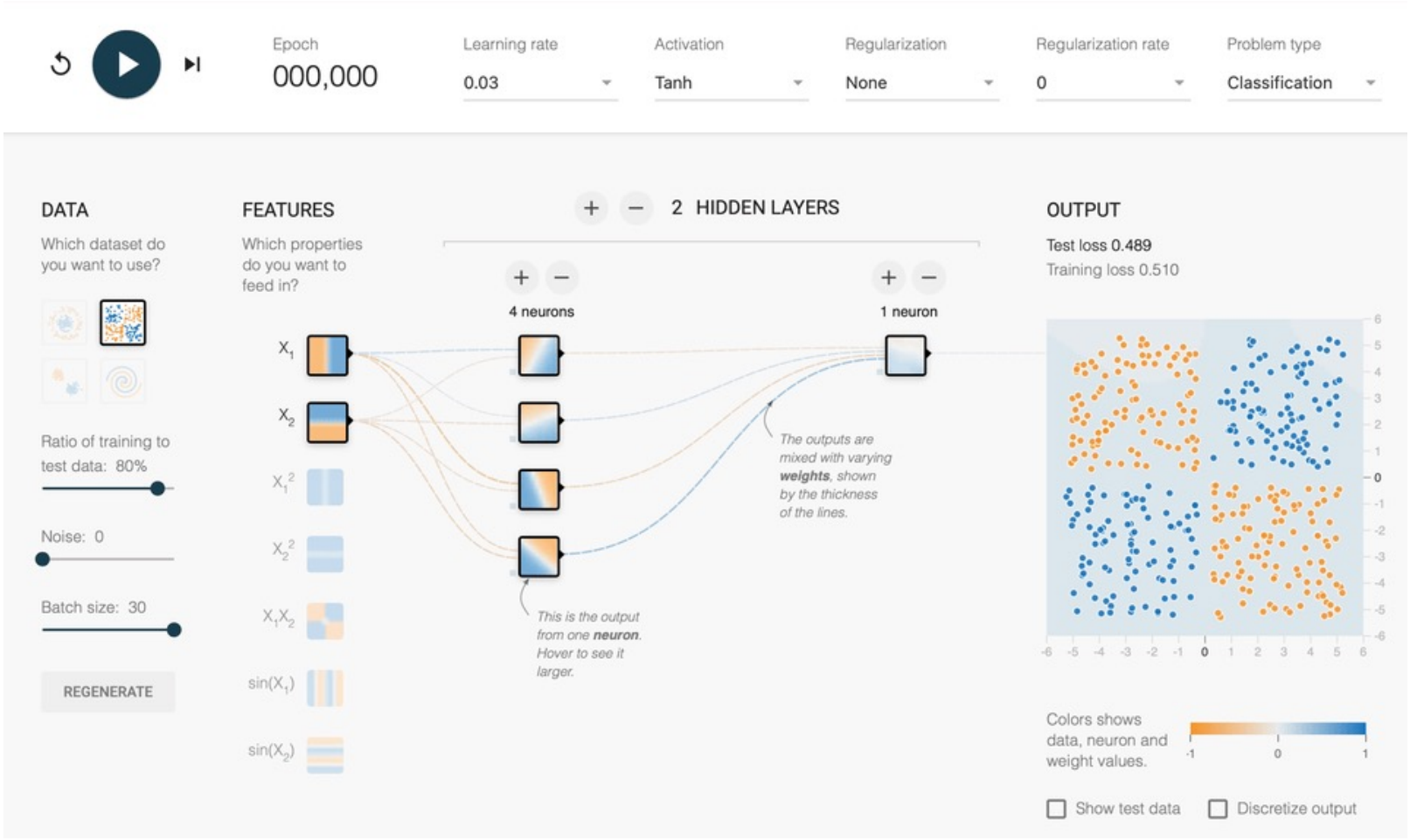






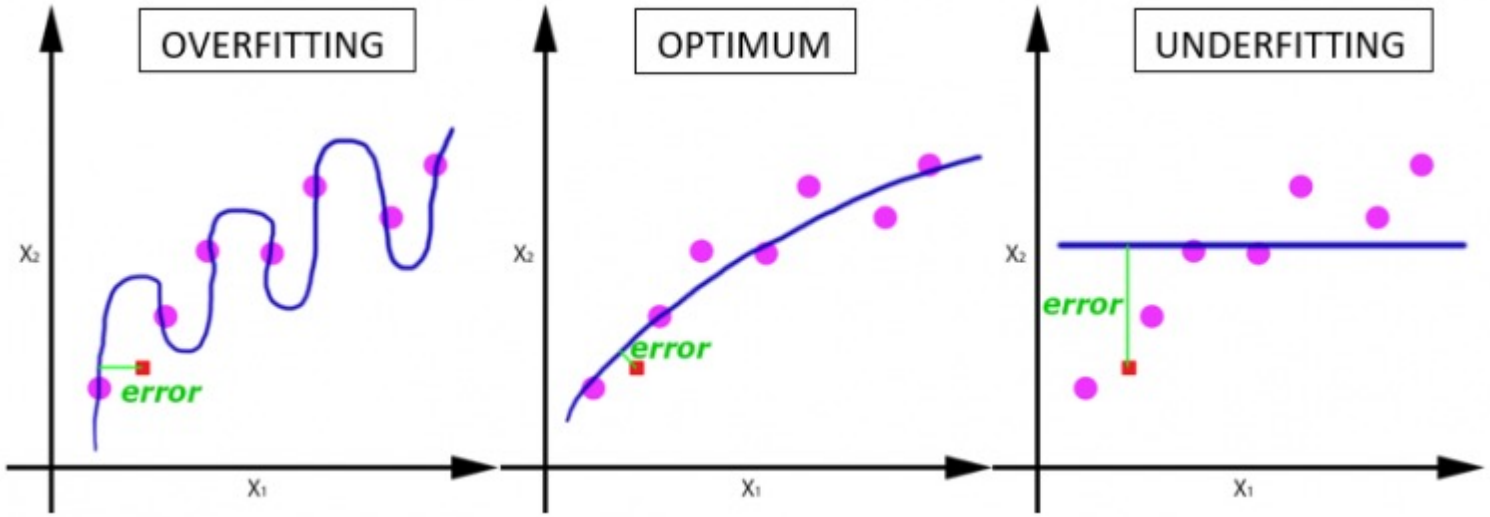


Визуализация обучения

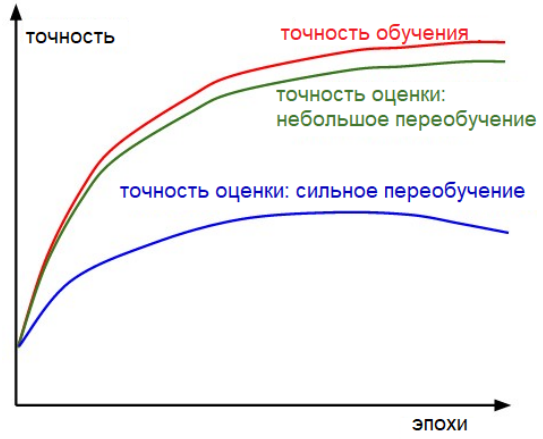


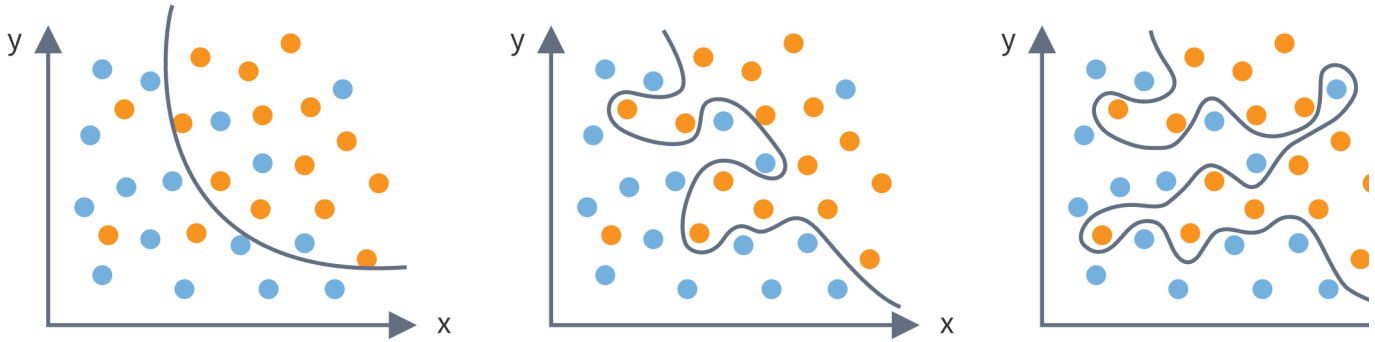
https://playground.tensorflow.org

Точность, переобучение

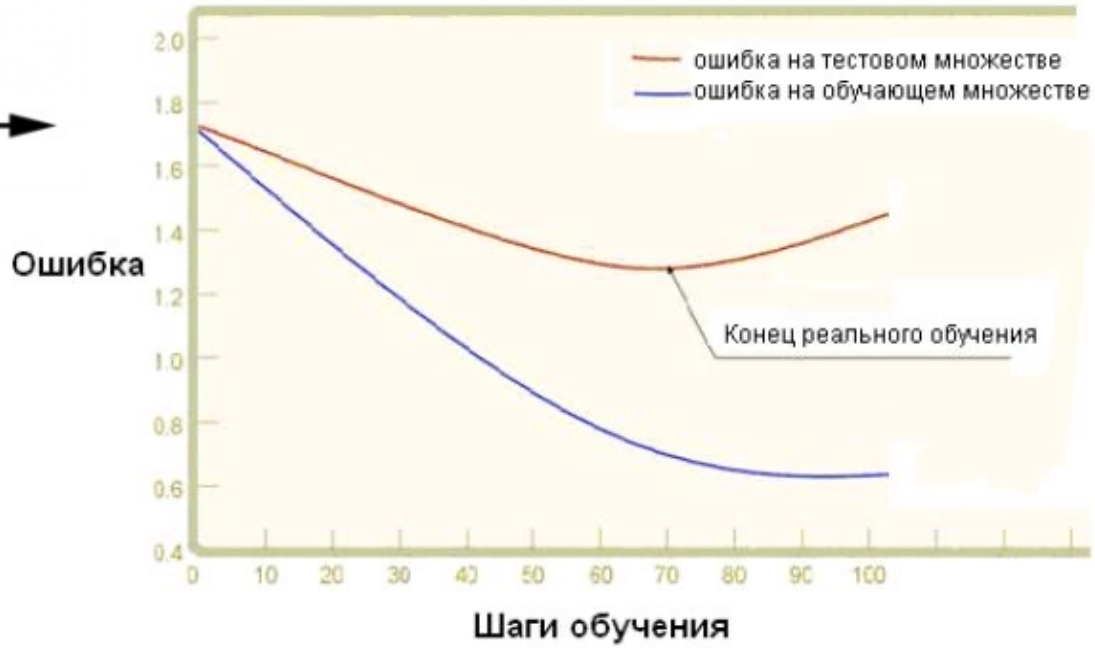


- Переобучение при долгом обучении слишком сложной модели



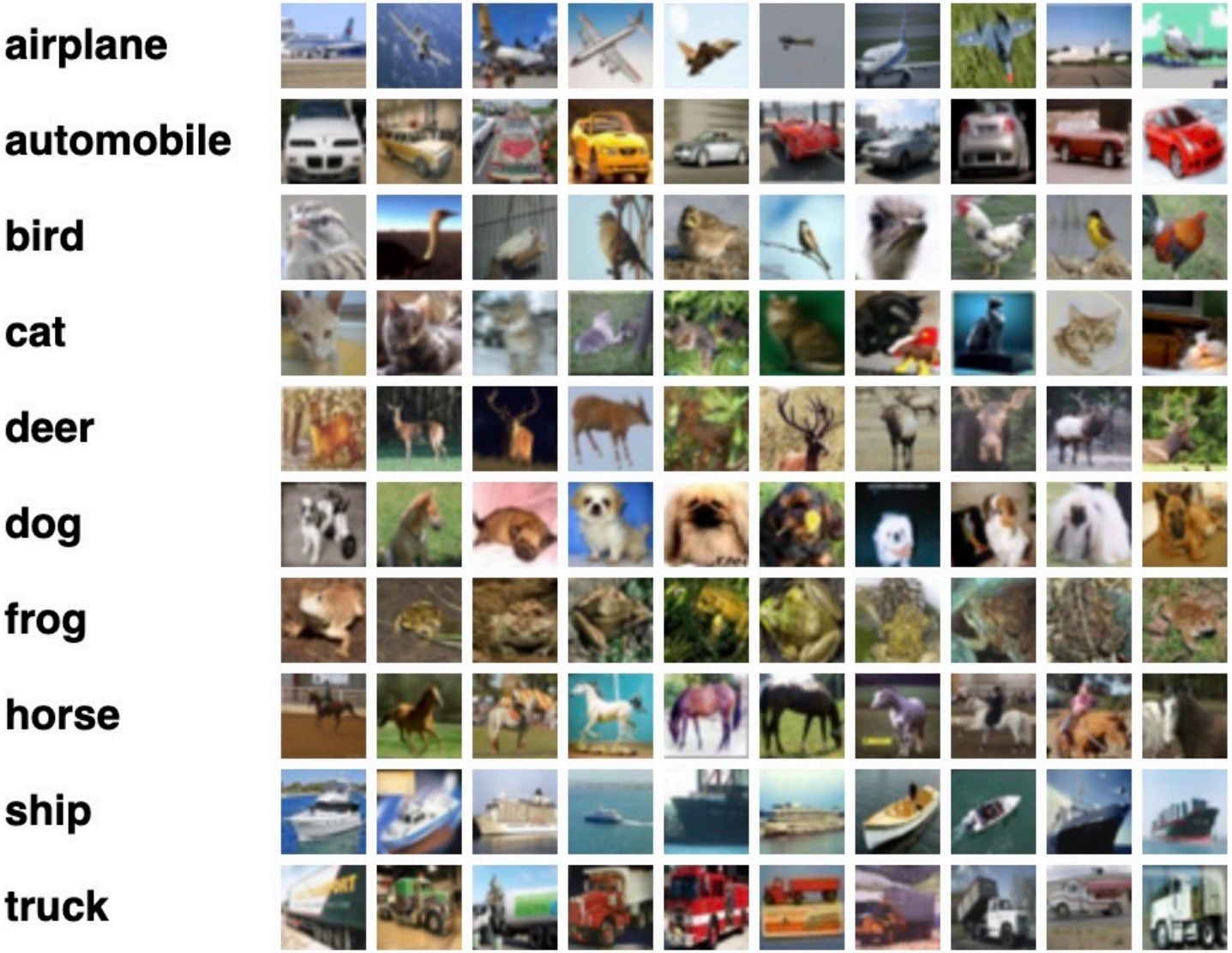


Недообучение | Оптимум | Переобучение



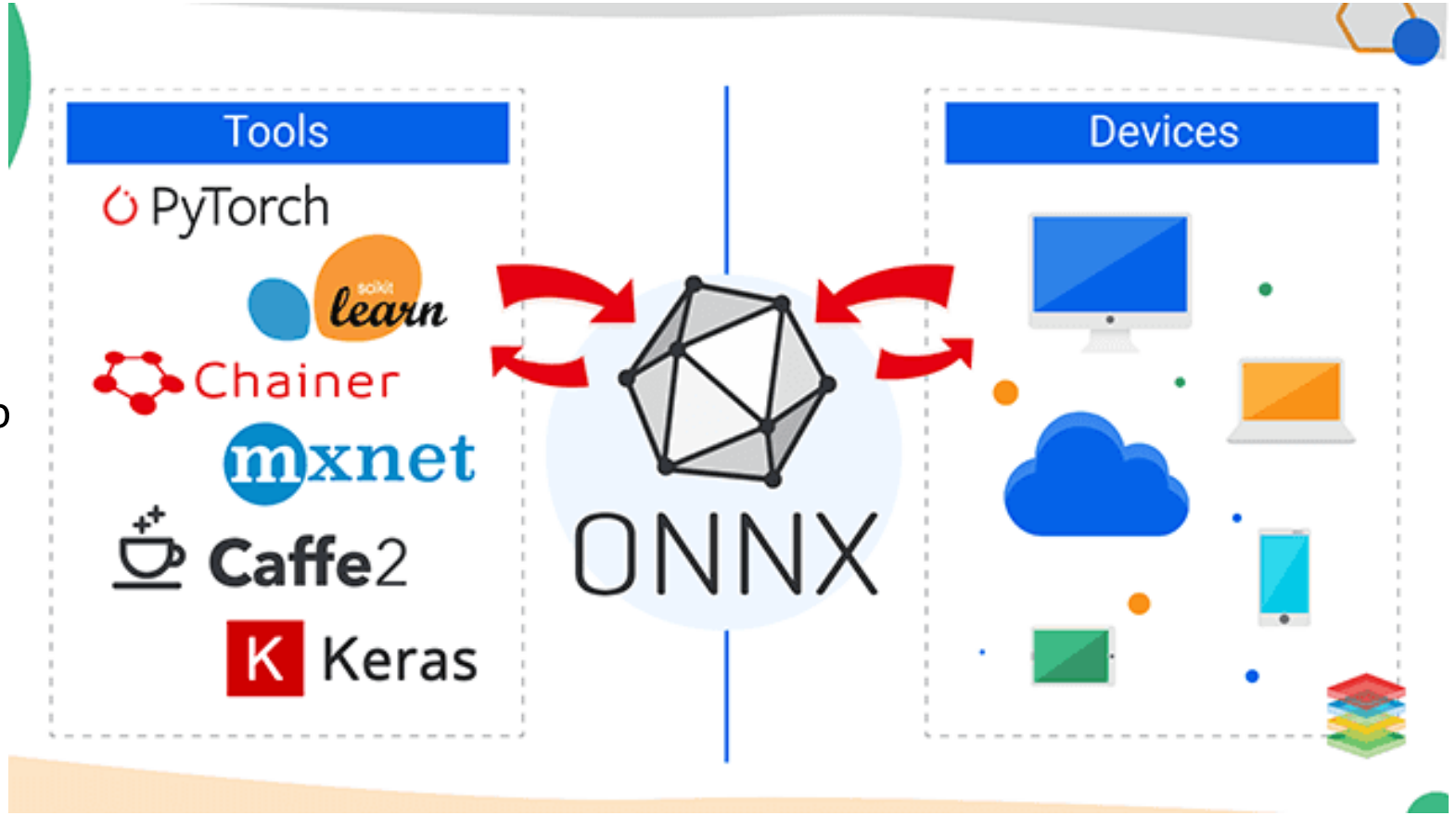
Cifar100

- Набор данных, состоящий из цветных изображений 100 классов
- Размер 32 на 32 пикселя
- 3 цвета



ONNX

- ONNX - библиотека для конвертации моделей между разными технологиями
- ONNX дает возможность исследователям и разработчикам выбрать нужную комбинацию инструментов для решения задачи
- ONNX.js позволяет запускать модели в браузере, то есть на стороне пользователя



ONNX пример

Step 3. Select class labels and get predictions

Выбрать файл cifar100_CNN.onnx Select ONNX file

Выбрать файл c25c94fe96_1000.jpg

Class label 54 Add 10 Random Undo Reset

0,50,54