



Кафедра молекулярных процессов и экстремальных состояний
вещества

Математические основы методов анализа результатов физического эксперимента

13. Проблемы обработки больших массивов
данных. Применение нейронных сетей в
анализе изображений в естествознании.

Коротеева Екатерина Юрьевна, ст. преп.
Дорощенко Игорь Александрович, ст. н. с.

Машинное обучение. Введение.

Машинное обучение

Машинное обучение – класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение за счёт применения решений множества сходных задач



Машинное обучение

Основные сферы применения машинного обучения:

- Робототехника
- Маркетинг
- Безопасность
- Финансовый сектор и страхование
- Медицина
- Добыча полезных ископаемых
- Биоинформатика
- Хемоинформатика
- ...

Машинное обучение

Основные компоненты машинного обучения:

- **данные**

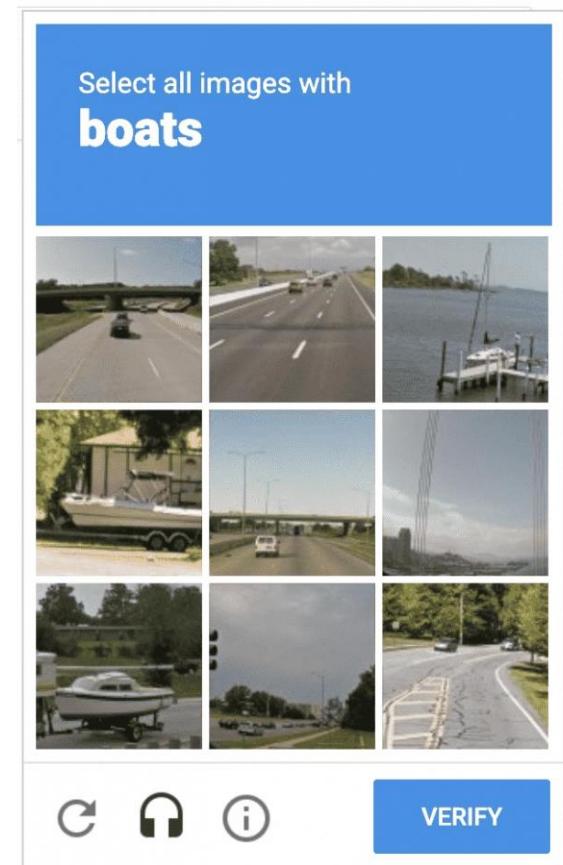
Собираются всевозможными способами. Чем больше данных, тем эффективней машинное обучение и точнее будущий результат.

- **признаки (*features*)**

Определяют, на каких параметрах строится машинное обучение

- **алгоритмы**

Метод машинного обучения (при условии наличия хороших данных) будет влиять на точность, скорость работы и размер готовой модели.



Машинное обучение

Основные компоненты машинного обучения:

- **данные**

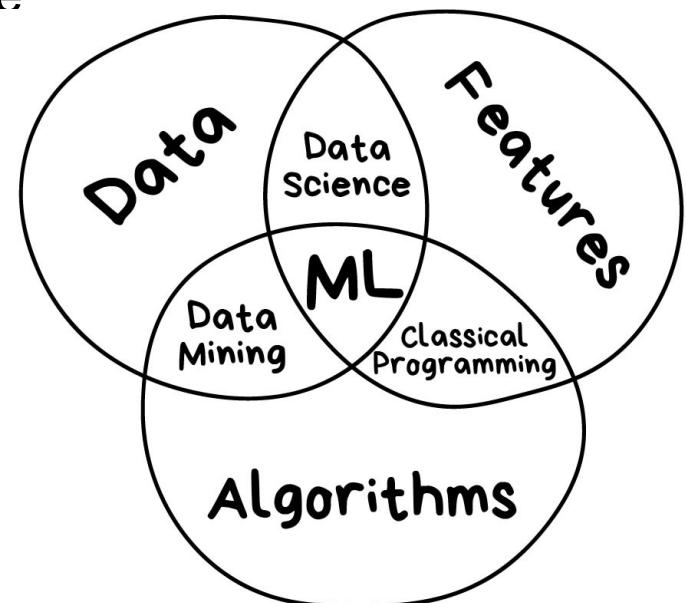
Собираются всевозможными способами. Чем больше данных, тем эффективней машинное обучение и точнее будущий результат.

- **признаки (features)**

Определяют, на каких параметрах строится машинное обучение

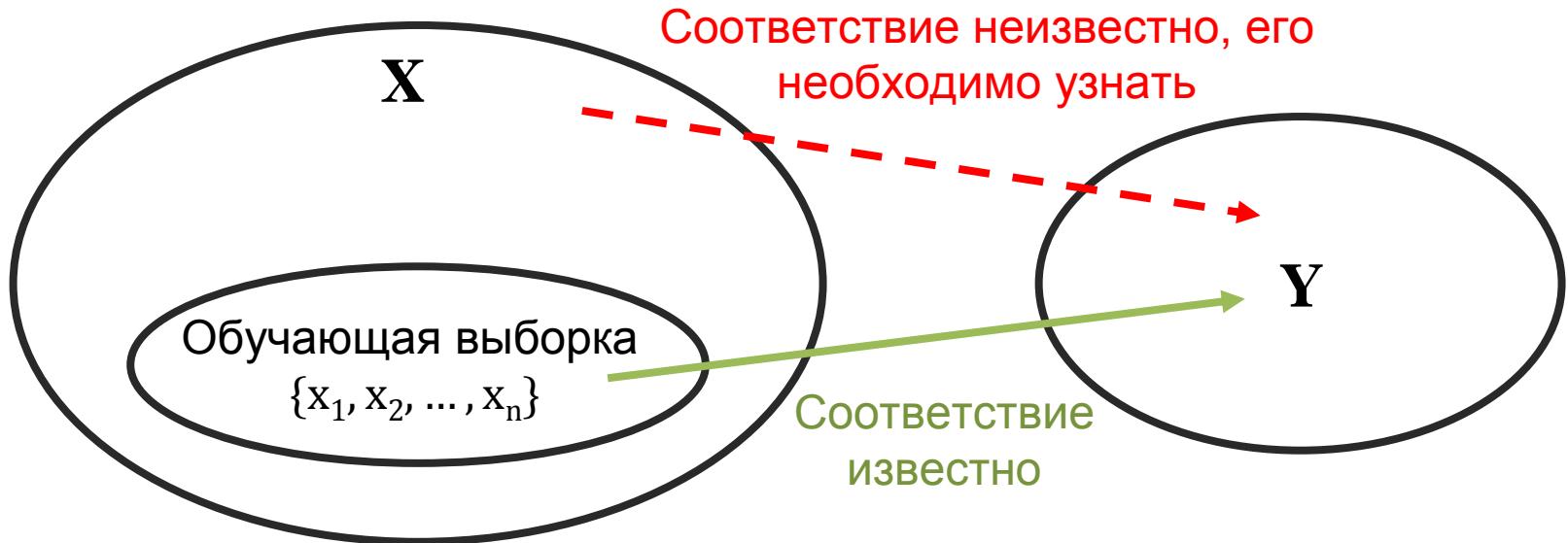
- **алгоритмы**

Метод машинного обучения (при условии наличия хороших данных) будет влиять на точность, скорость работы и размер готовой модели.



Машинное обучение. Постановка задачи.

- Пусть X — множество **объектов**, Y — множество **ответов** (строковые классы, числа), $y: X \rightarrow Y$ — искомая закономерность
- Подмножество $\{x_1, x_2, \dots, x_n\}$ множества X — обучающая выборка
- Нужно подобрать **алгоритм** $a: X \rightarrow Y$, приближающий функцию $y(x)$ на всём X



Машинное обучение. Постановка задачи.

- Пусть X – множество объектов (квартиры, пациенты, заемщики, email письма и т.д.)
- **Признак (feature)** – результат измерения некоторого параметра объекта $f(x)$

$f_1(x), f_2(x), \dots, f_n(x)$ – признаки объекта

$$\begin{bmatrix} f_1(x_1) & f_2(x_1) & \dots & f_k(x_1) \\ f_1(x_2) & f_2(x_2) & \dots & f_k(x_2) \\ \vdots & & & \vdots \\ f_1(x_n) & f_2(x_n) & \dots & f_k(x_n) \end{bmatrix}$$

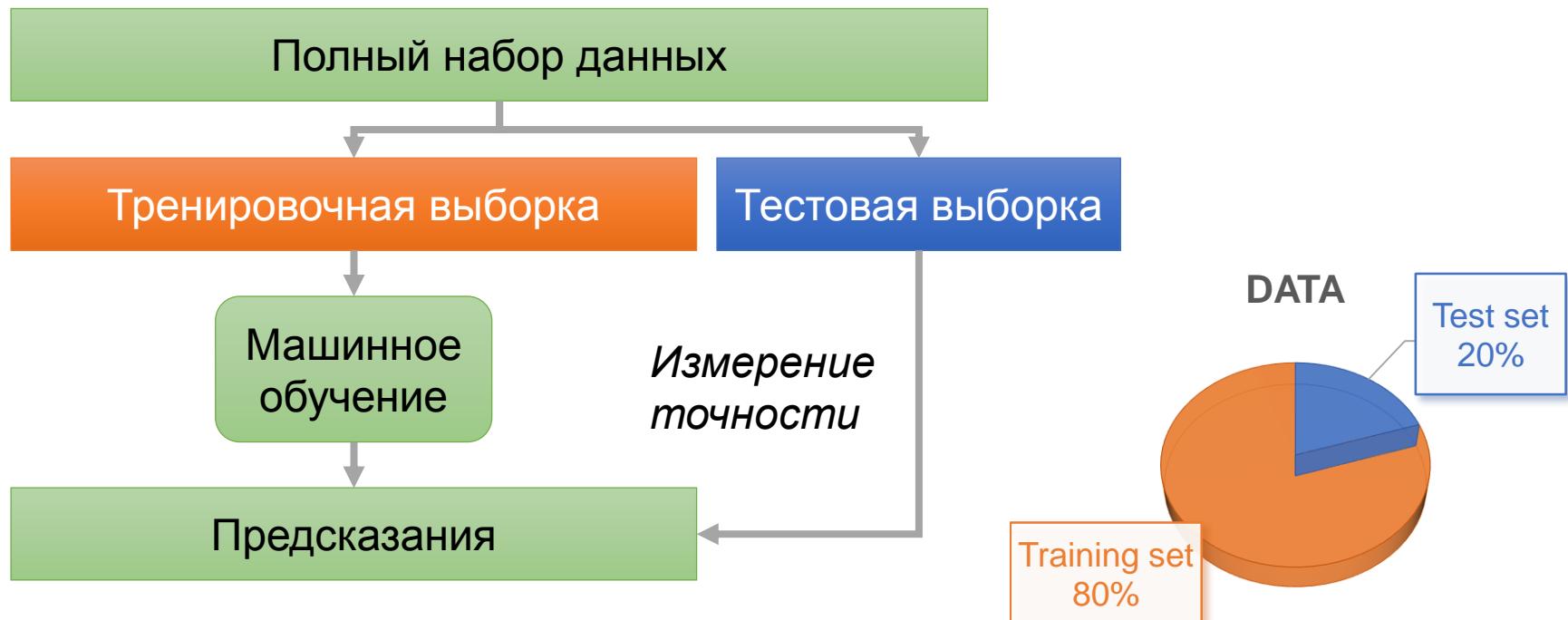
Матрица объектов-признаков

Признаки:

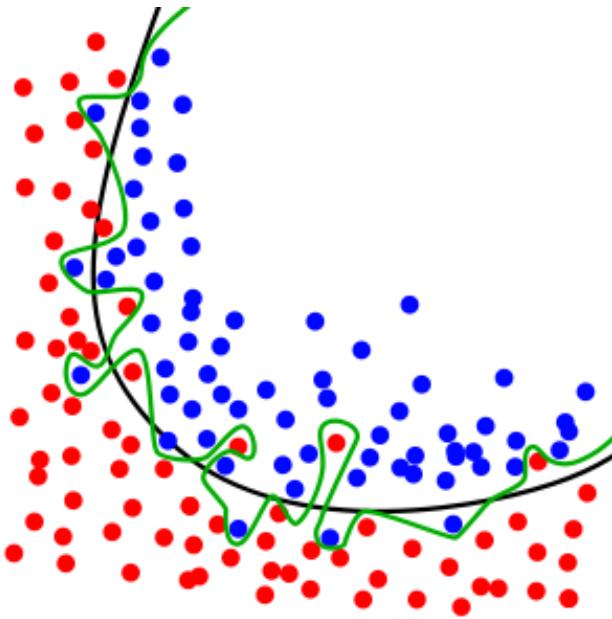
- числовые
- бинарные
- категориальные

Машинное обучение. Постановка задачи.

- Если в выборке достаточно данных, то ее имеет смысл разделить на 2 части:
тестовую и тренировочную (*test* and *train*).
- Модель обучается только на **тренировочной** выборке
- **Тестовая** выборка используется для проверки точности модели



Машинное обучение. Проблема переобучения.

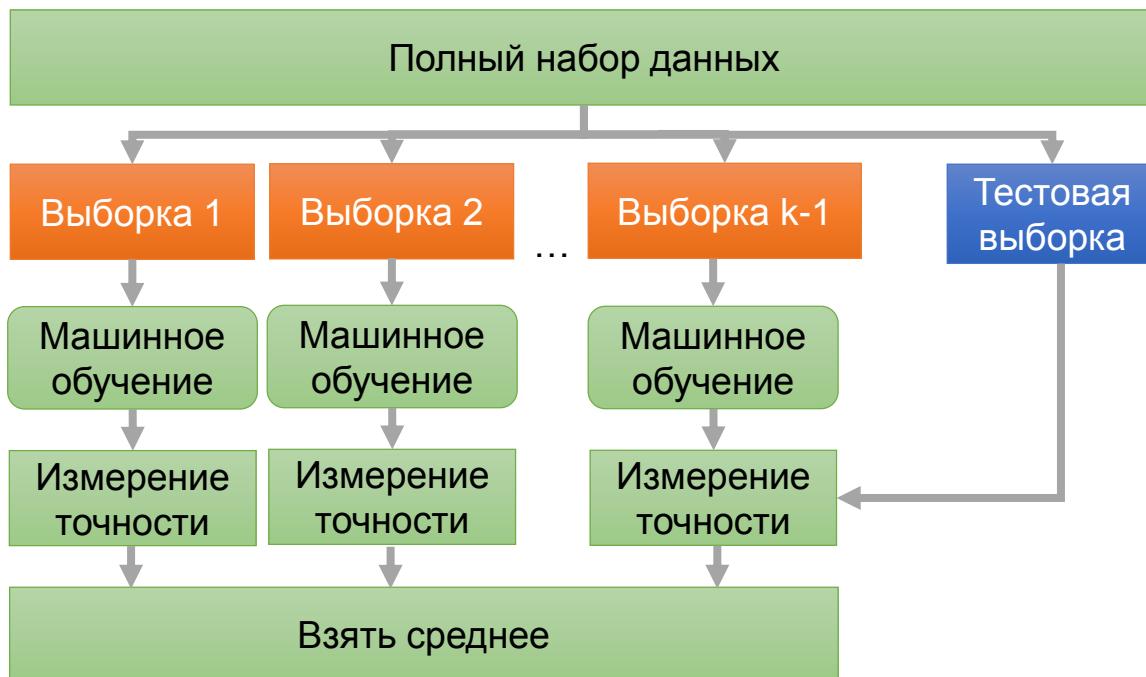


- Наборы данных должны быть достаточно большими и репрезентативными
 - Наборы данных должны выбираться случайным образом
 - Разделение на тренировочные и тестовые выборки – хороший способ справиться с переобучением
- ❖ Даже если разделение сделано, переобучение все еще может случиться, если наборы данных слишком малы или тестовый и тренировочный наборы данных очень похожи. Для решения проблемы применяется **кросс-валидация**.

Машинное обучение. Проблема переобучения.

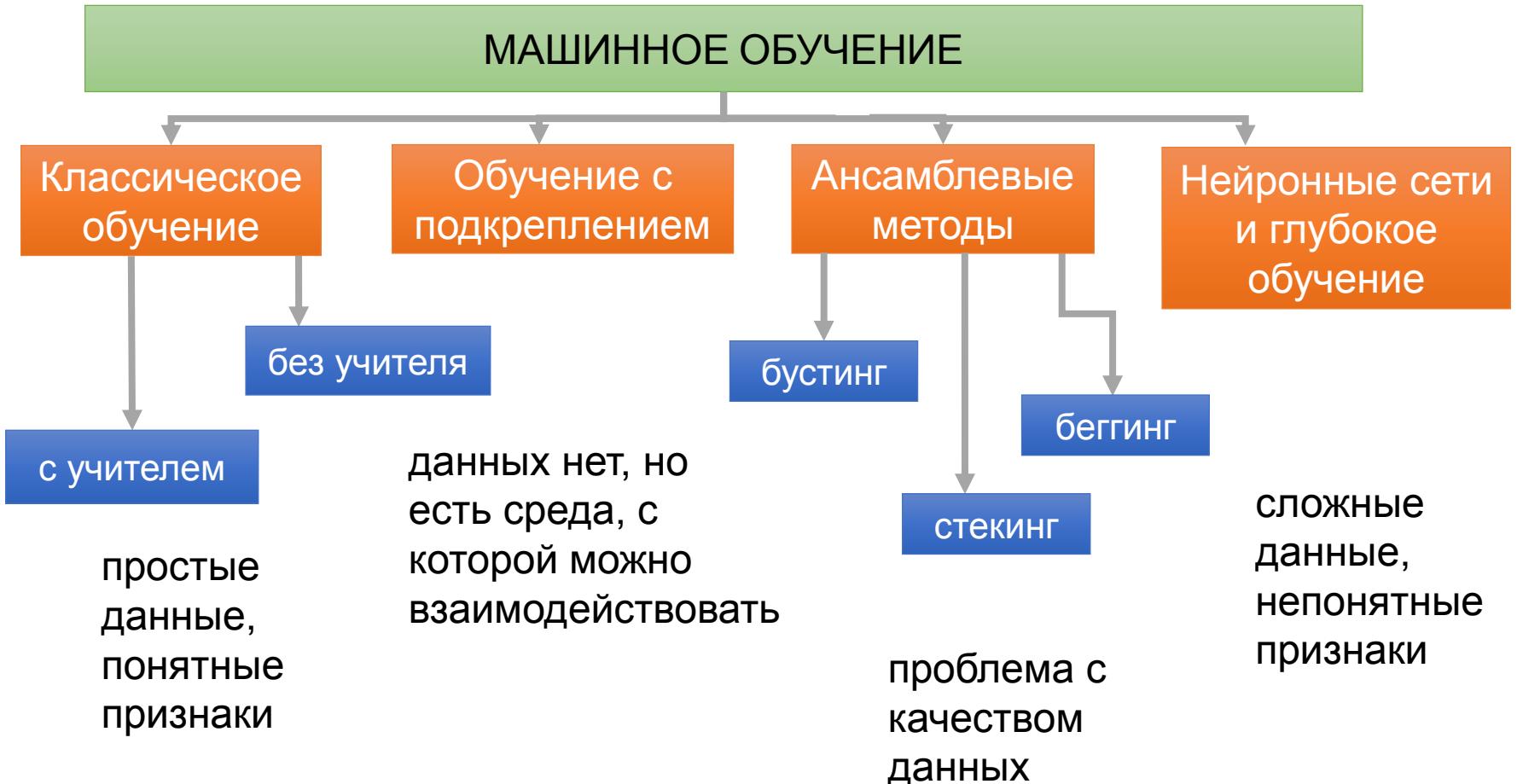
Кросс-валидация (cross-validation):

- Разделение данных на K случайных сегментов
- Выбор одного сегмента в качестве тестового



- Обучение на всех оставшихся K-1 наборах данных и вычисление их точности по сравнению с тестовым набором
- Расчет среднего по K-1 выборкам с использованием какой-либо метрики

Виды машинного обучения



Классическое обучение

Классическое обучение



Классическое обучение

- обучение с учителем (*supervised learning*)
«Верные» ответы даны (обучающая выборка)
- 1. Задача классификации:**
определить класс объекта, зная классы объектов из обучающей выборки
- Классификация изображений: дан набор объектов – фотографий - и каждому поставлен в соответствие класс «собака» или «кот».
К признакам можно отнести: количество и расположение границ и углов, гистограмму направленных градиентов и пр.
- Медицинская диагностика: дан набор объектов – пациенты. Классы – болен пациент или здоров. Признаки – пол (бинарный признак), наличие боли (бинарный признак), возраст (количественный признак), температура тела (количественный признак).



Класс «кот»



Класс «собака»

Классическое обучение

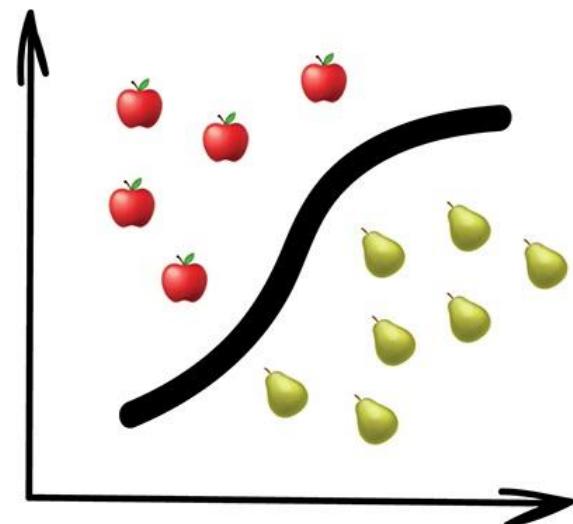
- обучение с учителем (*supervised learning*)
«Верные» ответы даны (обучающая выборка)

1. Задача классификации:

определить класс объекта, зная классы объектов из обучающей выборки

Применяется в:

- Спам-фильтры
- Определение языка
- Поиск похожих документов
- Анализ тональности
- Распознавание рукописных букв и цифр
- Определение подозрительных транзакций



Classification

Классическое обучение

- обучение с учителем (*supervised learning*)
«Верные» ответы даны (обучающая выборка)

1. Задача классификации:

определить класс объекта, зная классы объектов из обучающей выборки

Популярные алгоритмы:

- *Наивный Байес*
- *Деревья Решений*
- *Логистическая Регрессия*
- *K-ближайших соседей (k-NN)*
- *Машины Опорных Векторов (SVM)*



Классическое обучение

- обучение с учителем (*supervised learning*)
«Верные» ответы даны (обучающая выборка)

2. Задача регрессии:

определить число (из множества действительных чисел) по известным ответам обучающей выборки

Применяется в:

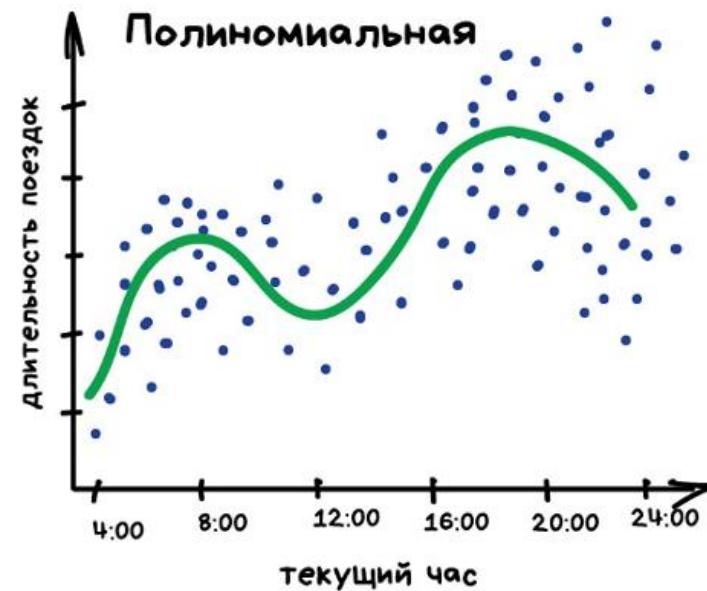
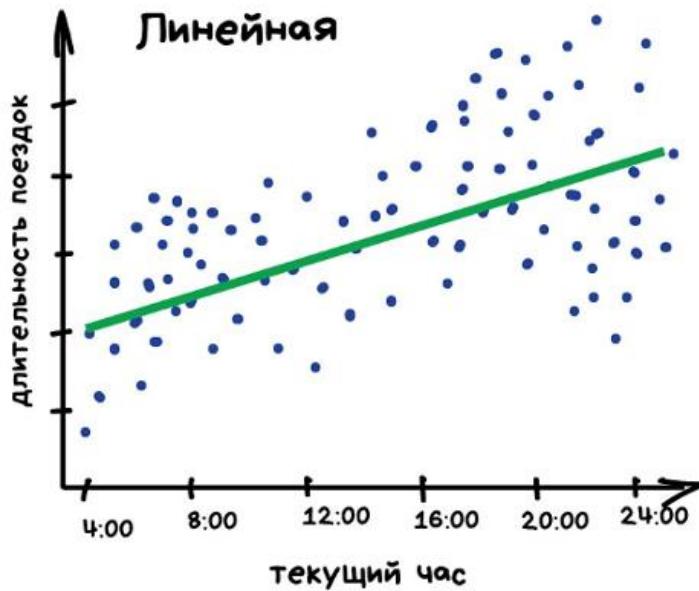
- Прогноз стоимости ценных бумаг
- Анализ спроса, объема продаж
- Медицинские диагнозы
- Прогнозирование
- Анализ трендов

Популярные алгоритмы: *Линейная и полиномиальная регрессии*

Классическое обучение

- обучение с учителем (*supervised learning*)
«Верные» ответы даны (обучающая выборка)
- 2. Задача регрессии:**
определить число (из множества действительных чисел) по известным ответам обучающей выборки

Предсказываем пробки



Классическое обучение

➤ обучение без учителя (*unsupervised learning*)

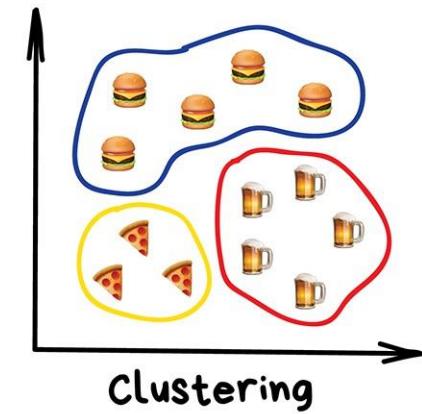
«Верных» ответов нет, алгоритм сам должен найти закономерности

1. Задача кластеризации:

разделить объекты на группы по некоторой закономерности. Как правило, закономерность изначально не формализована

Применяется в:

- Сегментация рынка (типов покупателей, лояльности)
- Объединение близких точек на карте
- Сжатие изображений
- Анализ и разметки новых данных
- Детекторы аномального поведения



Популярные алгоритмы: *Метод K-средних*, *Mean-Shift*, *DBSCAN*

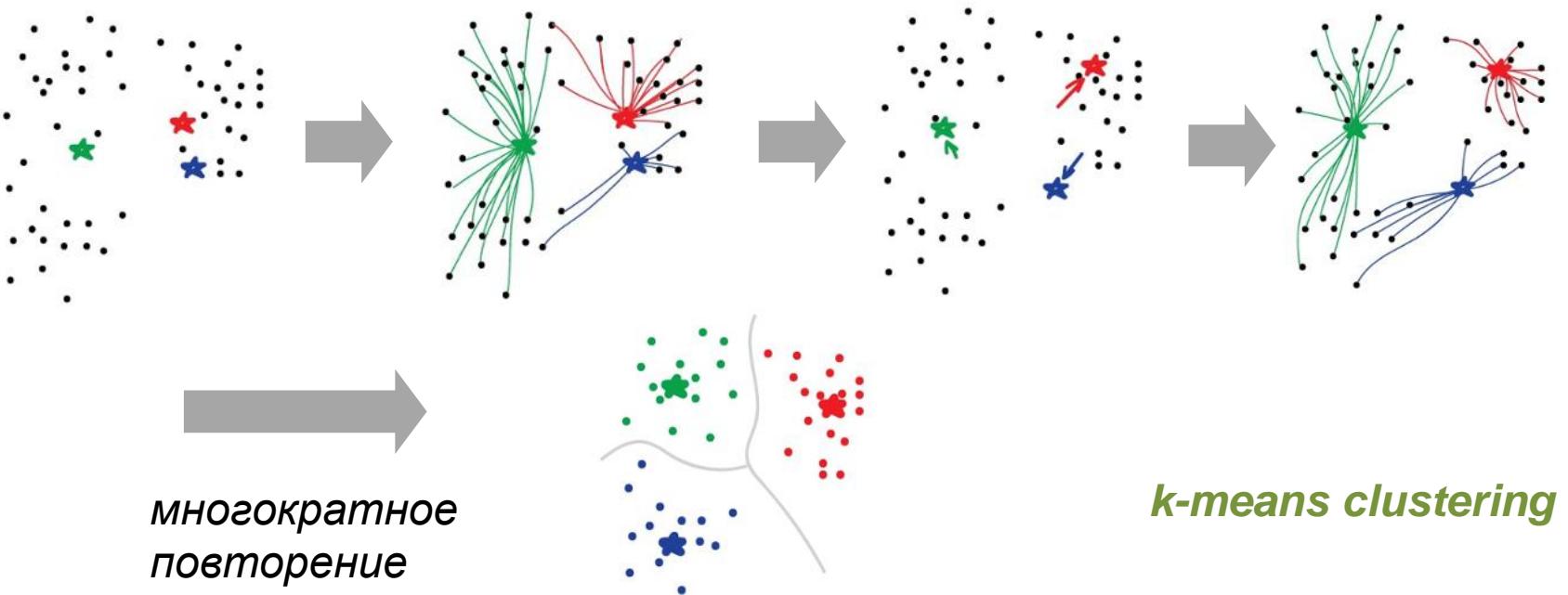
Классическое обучение

- обучение без учителя (*unsupervised learning*)

«Верных» ответов нет, алгоритм сам должен найти закономерности

1. Задача кластеризации:

разделить объекты на группы по некоторой закономерности. Как правило, закономерность изначально не формализована

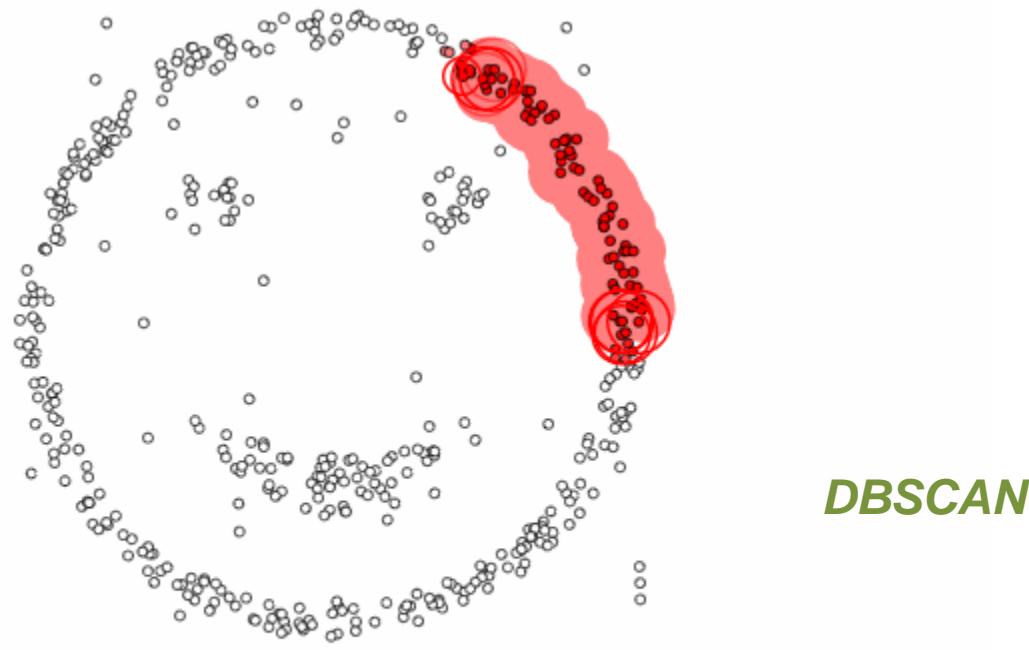


Классическое обучение

- обучение без учителя (*unsupervised learning*)
«Верных» ответов нет, алгоритм сам должен найти закономерности

1. Задача кластеризации:

разделить объекты на группы по некоторой закономерности. Как правило, закономерность изначально не формализована



Классическое обучение

- обучение без учителя (*unsupervised learning*)

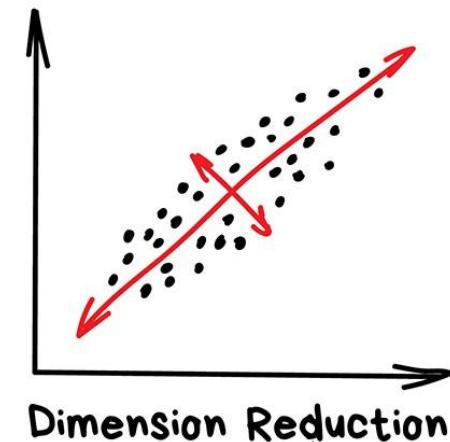
«Верных» ответов нет, алгоритм сам должен найти закономерности

2. Задача понижения размерности:

преобразование данных, состоящее в уменьшении числа признаков путём выделения признаков более высокого уровня

Применяется в:

- Рекомендательные Системы (*)
- Красивые визуализации
- Определение тематики и поиска похожих документов
- Риск-менеджмент



Классическое обучение

- обучение без учителя (*unsupervised learning*)

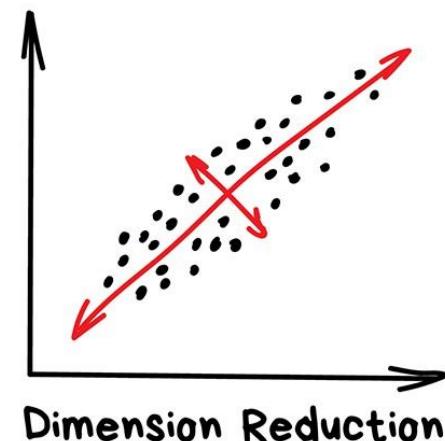
«Верных» ответов нет, алгоритм сам должен найти закономерности

2. Задача понижения размерности:

преобразование данных, состоящее в уменьшении числа признаков путём выделения признаков более высокого уровня

Популярные алгоритмы:

- Метод главных компонент (PCA)
- Сингулярное разложение (SVD)
- Латентное размещение Дирихле (LDA)
- Латентно-семантический анализ (LSA, pLSA, GLSA)



Классическое обучение

- обучение без учителя (*unsupervised learning*)

«Верных» ответов нет, алгоритм сам должен найти закономерности

2. Задача понижения размерности:

преобразование данных, состоящее в уменьшении числа признаков путём выделения признаков более высокого уровня

Сингулярное разложение (SVD)
матрицы M на произведение
трех матриц



$$M_{m \times n} = U_{m \times m} \Sigma_{m \times n} V^*_{n \times n}$$

Diagram illustrating the Singular Value Decomposition (SVD) of a matrix M into three matrices: U , Σ , and V^* . The original matrix M is shown as a grey grid. It is decomposed into U (orthogonal columns), Σ (diagonal singular values), and V^* (orthogonal rows). The matrices U and V^* are shown as colored blocks, while Σ is a diagonal matrix.

$$U \quad U^* = I_m$$
$$V \quad V^* = I_n$$

Классическое обучение

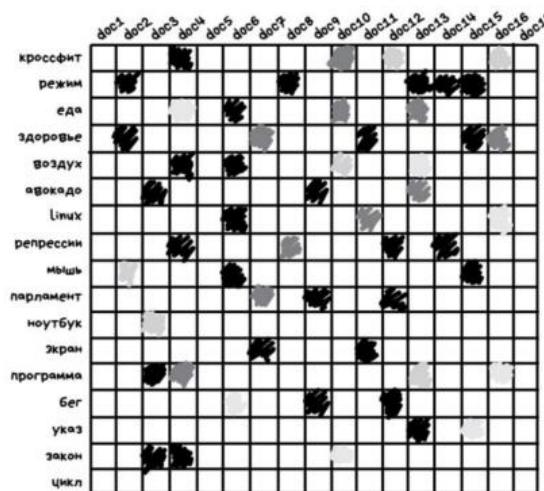
- обучение без учителя (*unsupervised learning*)

«Верных» ответов нет, алгоритм сам должен найти закономерности

2. Задача понижения размерности:

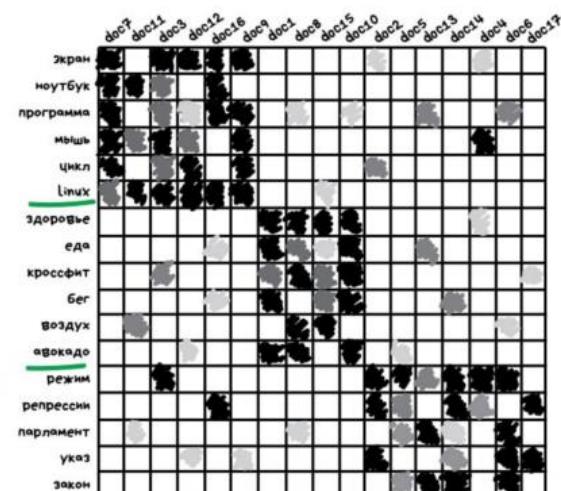
преобразование данных, состоящее в уменьшении числа признаков путём выделения признаков более высокого уровня

Латентно-
семантический
анализ –
разделение
документов по
темам



1. Строим матрицу как часто каждое слово встречается в каждом документе (чёрнее - чаще)

2. Раскладываем
→
SVD



3. Получаем наглядные кластера по тематикам (даже если слова не встречались вместе)

Классическое обучение

➤ обучение без учителя (*unsupervised learning*)

«Верных» ответов нет, алгоритм сам должен найти закономерности

3. Задача поиска ассоциативных правил: обнаружение закономерностей в данных

Применяется в:

- Прогноз акций и распродаж
- Анализ товаров, покупаемых вместе
- Расстановка товаров на полках
- Анализ паттернов поведения на веб-сайтах

Популярные алгоритмы: *Apriori*, *Euclat*, *FP-growth*

Обучение с подкреплением

Обучение с подкреплением

Обучение с подкреплением (Reinforcement learning) - способ машинного обучения, в ходе которого испытуемая система (агент) обучается, взаимодействуя с некоторой средой. Откликом среды на принятые решения являются *сигналы подкрепления*

Применяется для:

- Самоуправляемых автомобилей
- Роботов пылесосов
- Игр
- Автоматической торговли
- Управления ресурсами предприятий



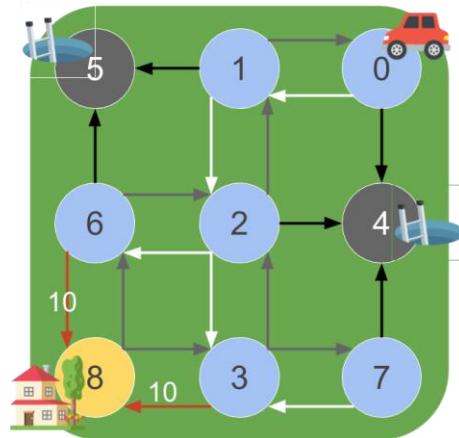
Популярные алгоритмы: *Q-Learning*, *SARSA*, *DQN*, *A3C*,
Генетический Алгоритм

Обучение с подкреплением

Q-Learning – метод обучения с подкреплением, не формирующий модель окружающей среды (model-free).

На основе подкрепления от среды агент может сравнивать функцию полезности Q доступных действий.

Q-Learning применяется для ситуаций, которые можно представить в виде марковского процесса принятия решений.



Q-table initialised at zero

	UP	DOWN	LEFT	RIGHT
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0

After few episodes

	UP	DOWN	LEFT	RIGHT
0	0	0	0	0
1	0	0	0	0
2	0	2.25	2.25	0
3	0	0	5	0
4	0	0	0	0
5	0	0	0	0
6	0	5	0	0
7	0	0	2.25	0
8	0	0	0	0

Eventually

	UP	DOWN	LEFT	RIGHT
0	0	0	0.45	0
1	0	1.01	0	0
2	0	2.25	2.25	0
3	0	0	5	0
4	0	0	0	0
5	0	0	0	0
6	0	5	0	0
7	0	0	2.25	0
8	0	0	0	0

Ансамблевые методы

Ансамблевые методы

Ансамблевые методы — это парадигма машинного обучения, где несколько моделей (т.н. «слабые ученики») обучаются для решения одной и той же проблемы и объединяются для получения лучших результатов.

Применяются для:

- Всего, где подходят классические алгоритмы (но работают точнее)
- Поисковые системы (*)
- Компьютерное зрение
- Распознавание объектов

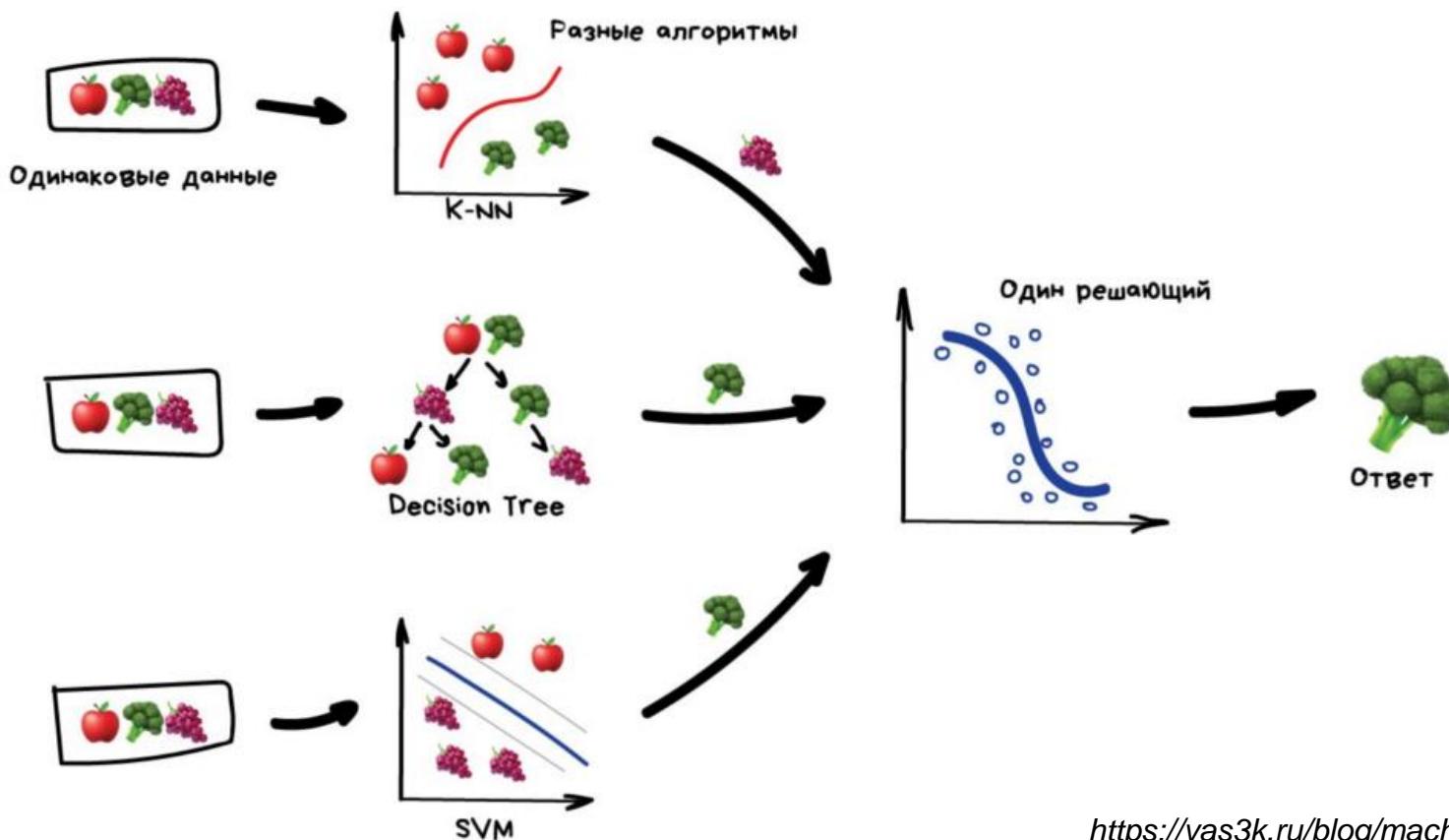


Популярные алгоритмы: *Random Forest*, *Gradient Boosting*

Ансамблевые методы

1. Стекинг (Stacking):

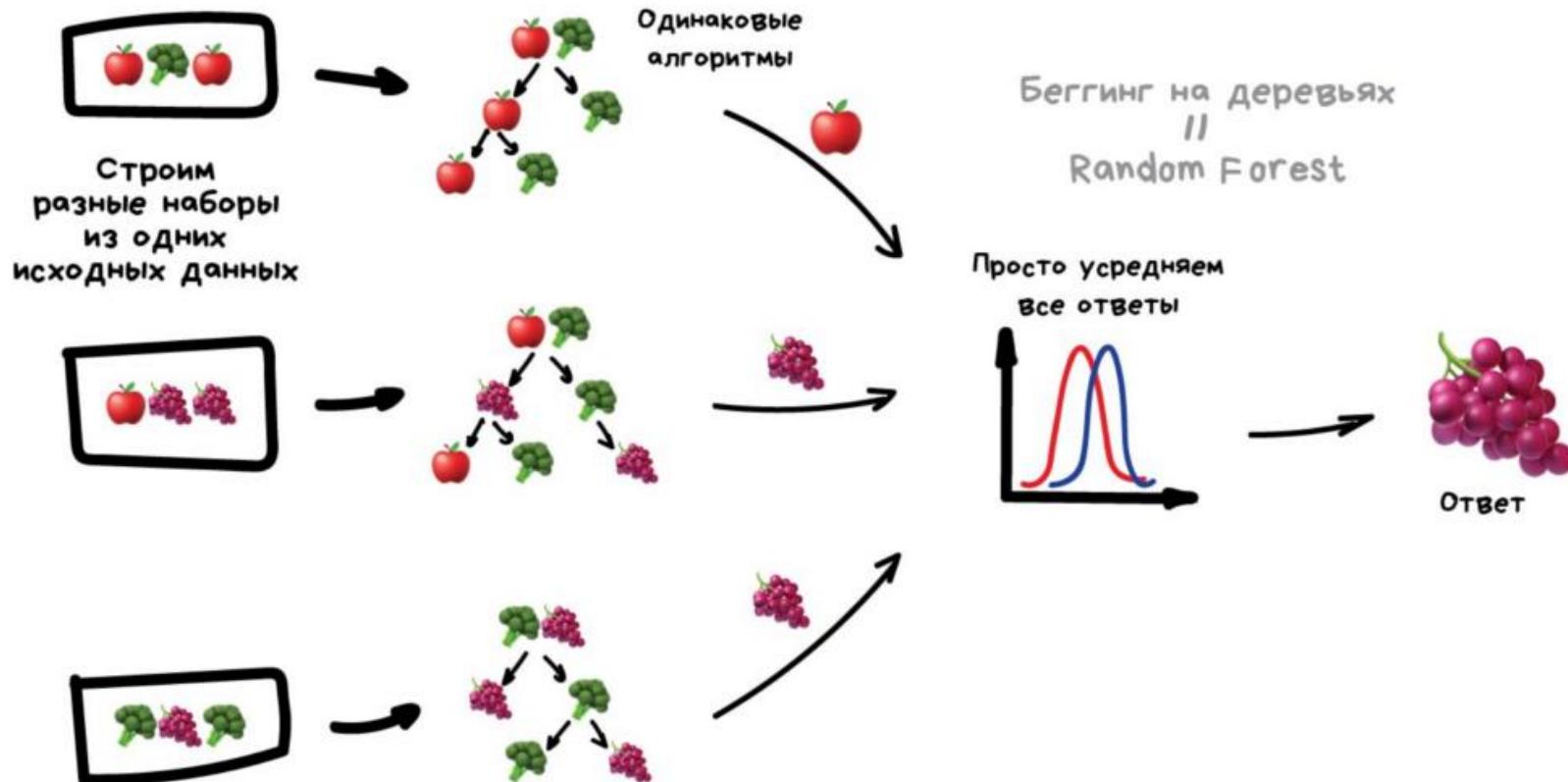
обучают несколько разнородных алгоритмов («слабых учеников») на одних данных, затем результаты подают на вход последнему алгоритму, принимающему окончательно решение



Ансамблевые методы

2. Бэггинг (Bootstrap AGGREGATING):

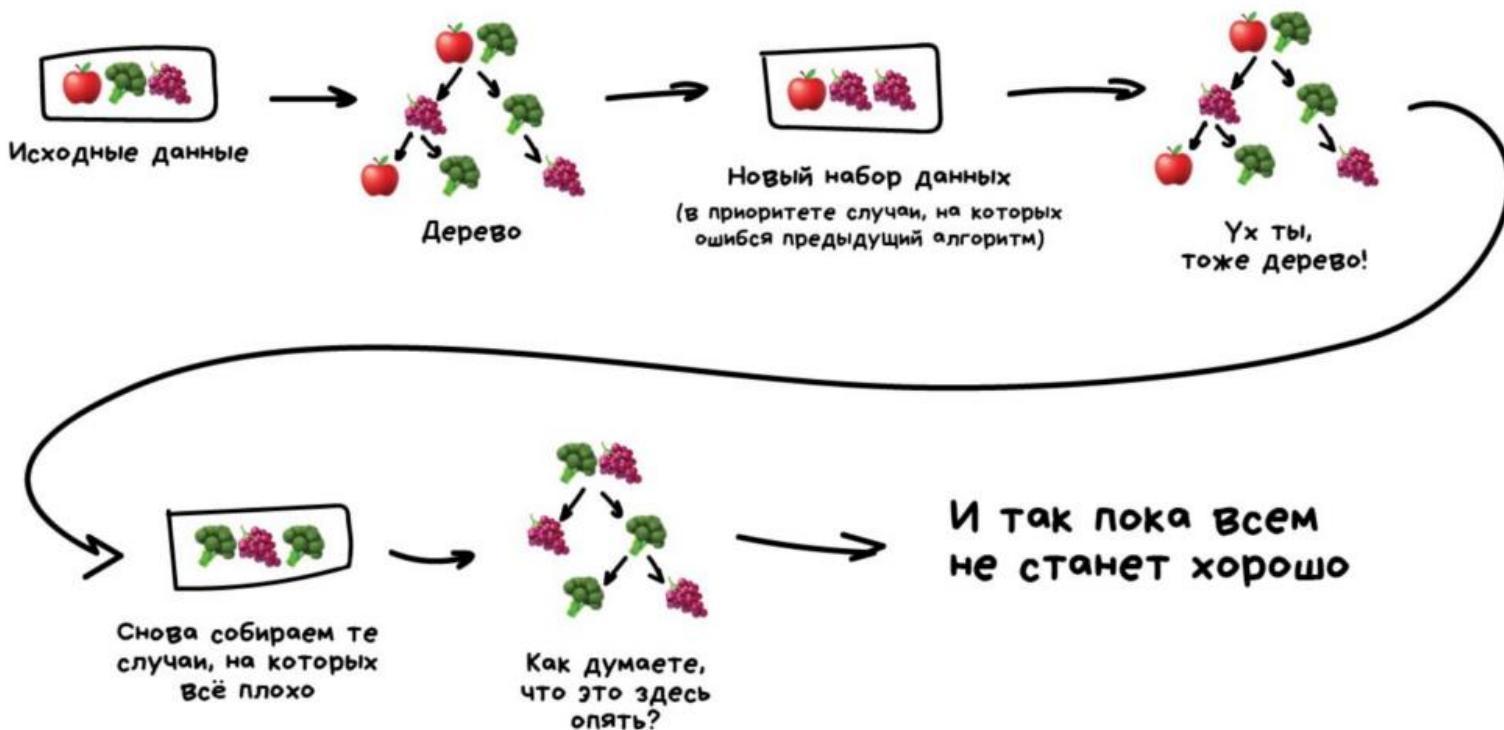
обучают один алгоритм на случайных выборках из исходных данных, затем результаты усредняют



Ансамблевые методы

3. Бустинг (Boosting):

обучают (обычно) однородных слабых учеников последовательно адаптивным способом (в каждую новую выборку берется часть тех данных, на которых предыдущий алгоритм отработал неправильно)



Нейросети и глубокое обучение

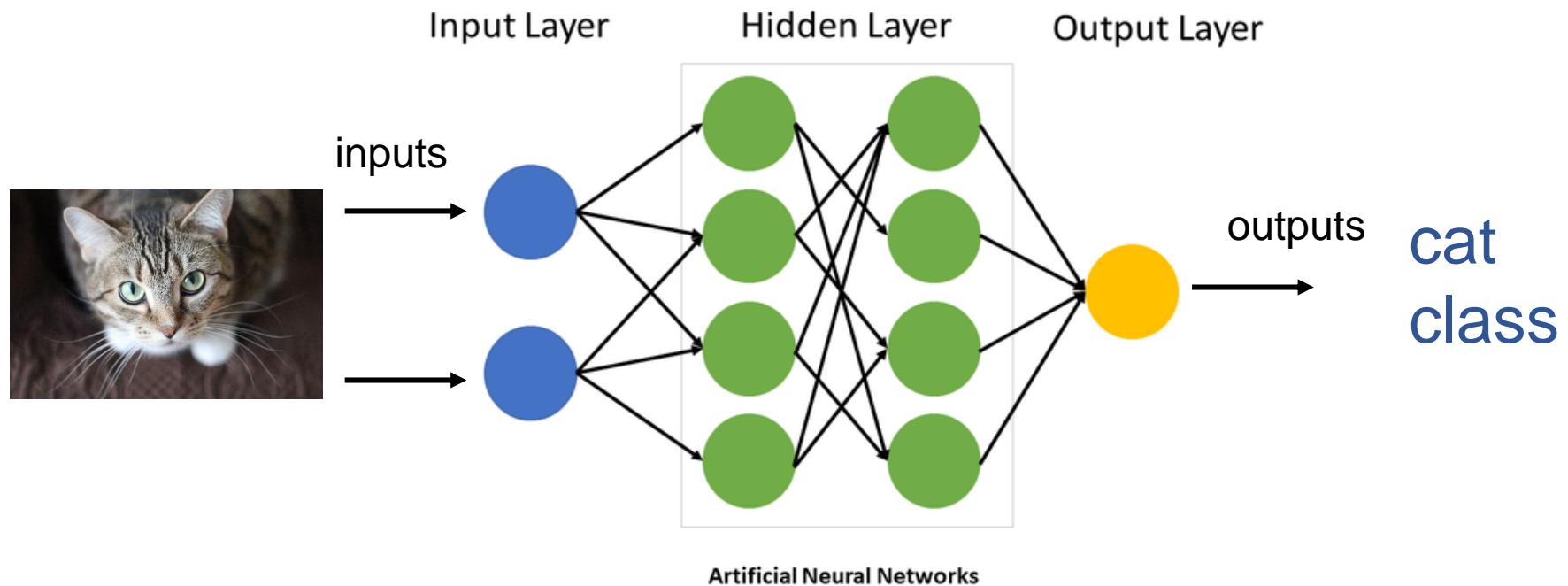
Нейросети и глубокое обучение

Библиотеки и ПО



Нейросети и глубокое обучение

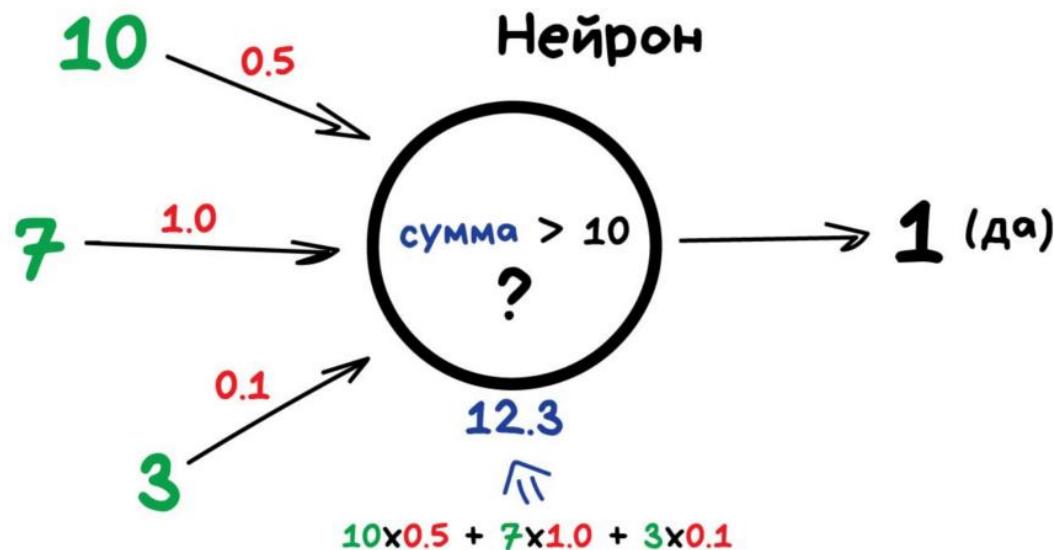
Нейронная сеть представляет собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов)



Модель с 1 нейроном

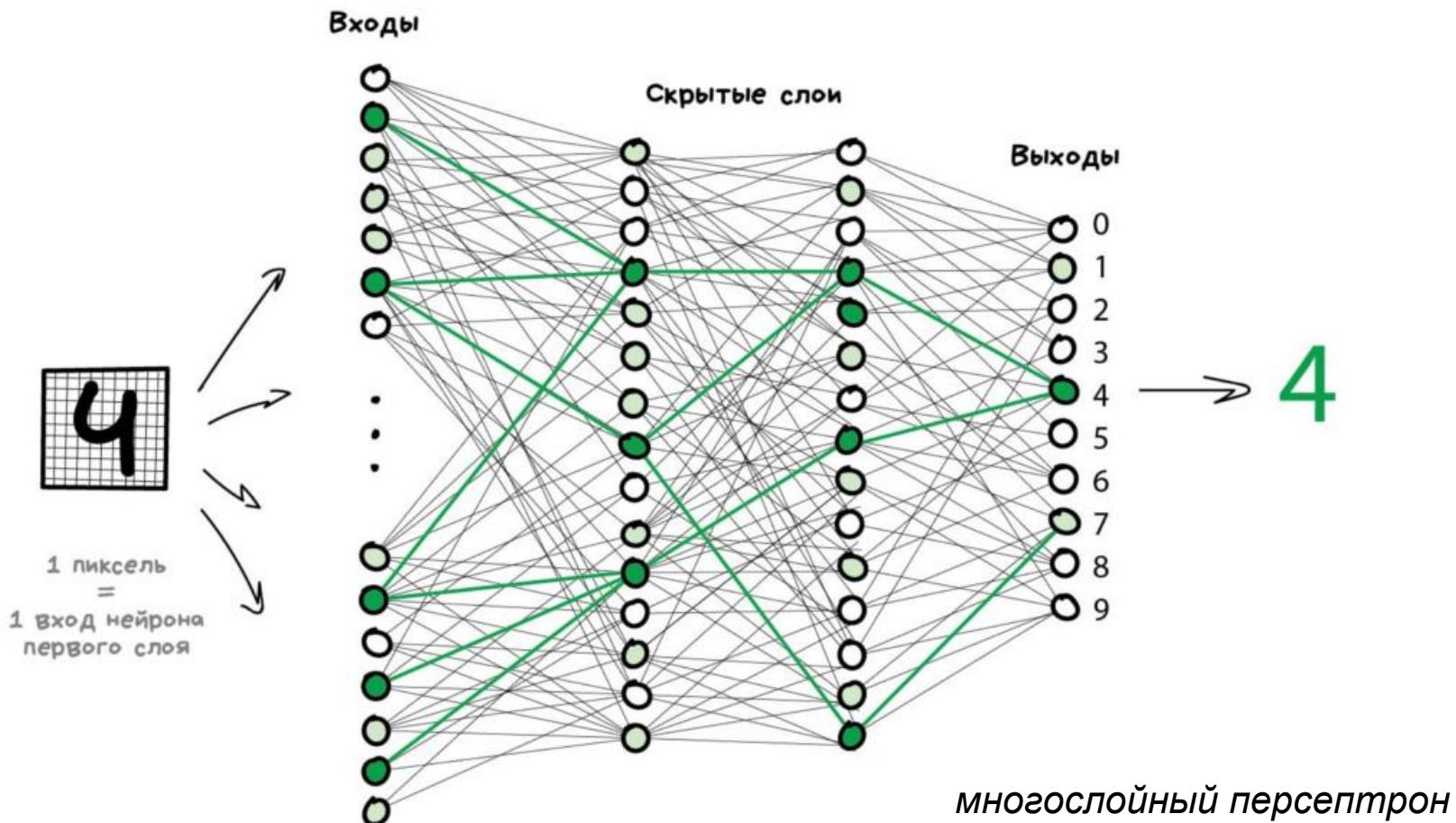
Модель нейрона:

- Нейрон обрабатывает сигналы от каждого источника и выдает один выходной сигнал
- Нейрон можно представить как нелинейную функцию (функцию активации) от единственного аргумента — линейной комбинации всех входных сигналов



- Обучение нейросети заключается в нахождении коэффициентов связей между нейронами

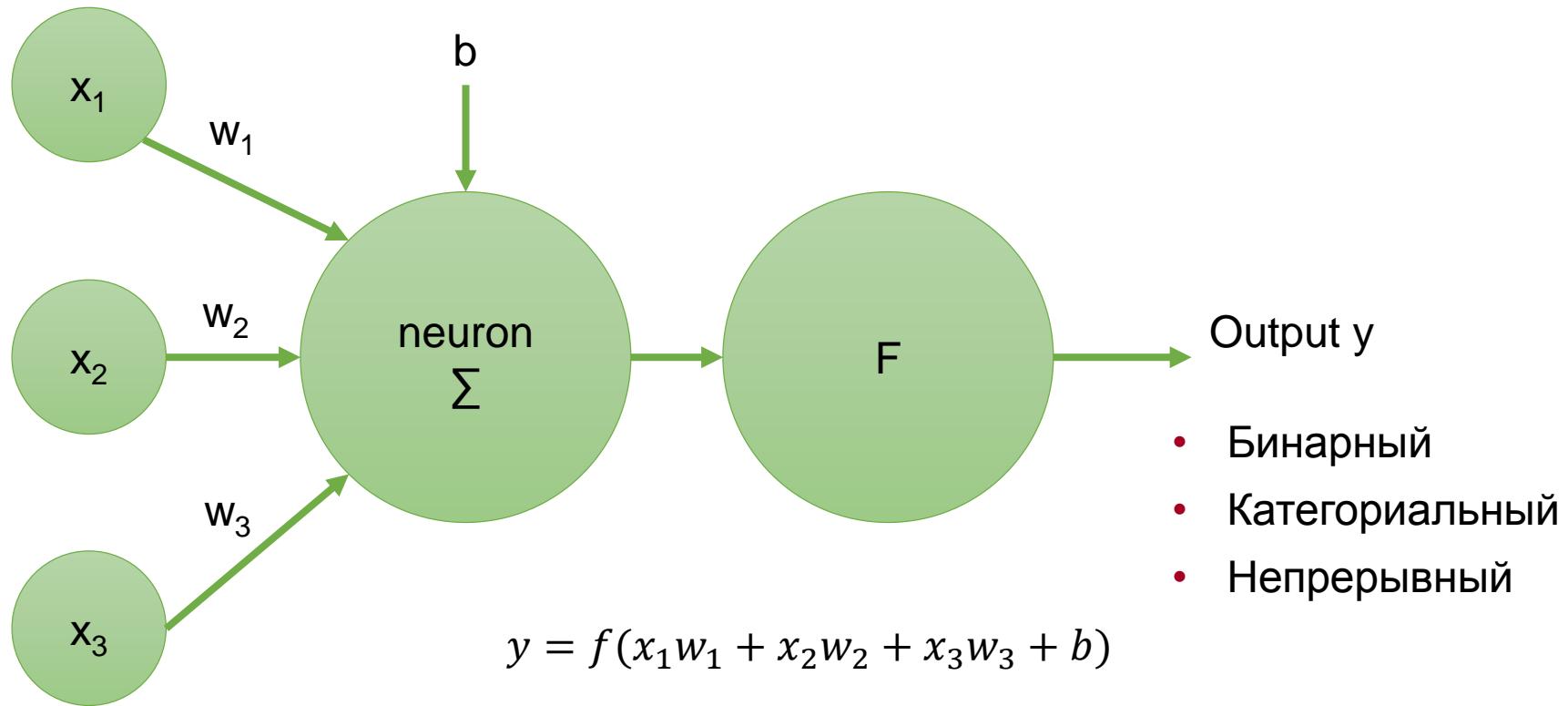
Нейросети и глубокое обучение



Принцип «черного ящика»: внутренняя структура и распределение весов, как правило, сложны для человеческого анализа и задаются автоматически в процессе обучения.

Модель с 1 нейроном

- F – функция активации (activation function).
Смещение b позволяет сдвинуть кривую функции активации вверх или вниз.
 - Веса w – числа, которые задают уровень влияния каждого входного сигнала.

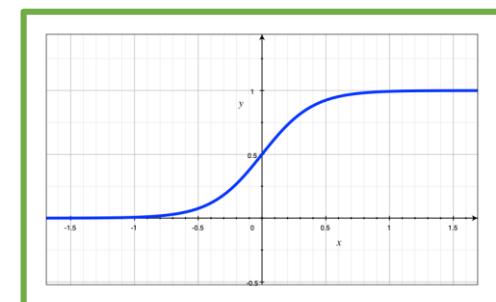
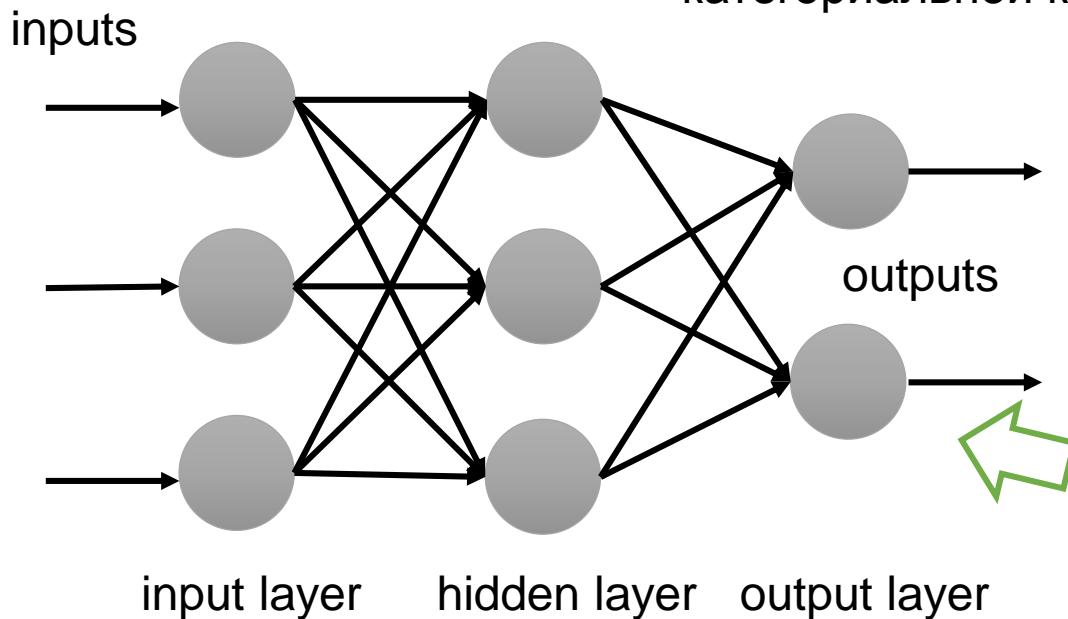


Модель с 1 нейроном

Пример 1. Функция активации F - sigmoid

$$y = \frac{1}{1 + e^{-(a+bx)}}$$

- Преобразует входной аргумент в число в диапазоне [0, 1]
- Преобразует большие отрицательные числа в 0, а большие положительные в 1
 - Как правило, используется в последнем слое (output layer), например, для бинарной или категориальной классификации

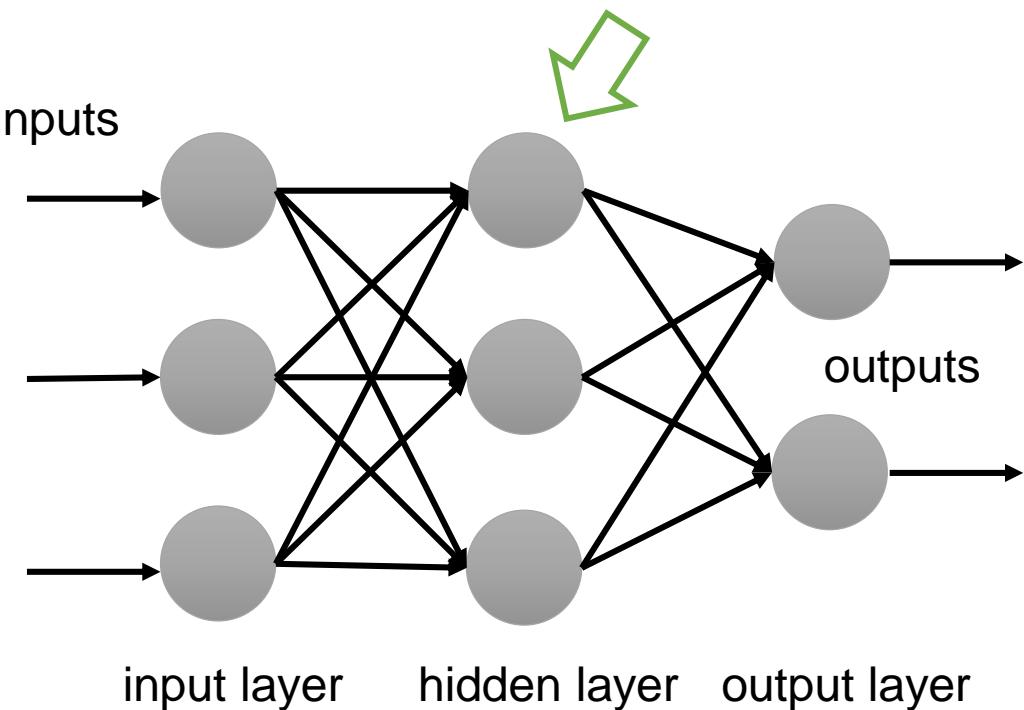
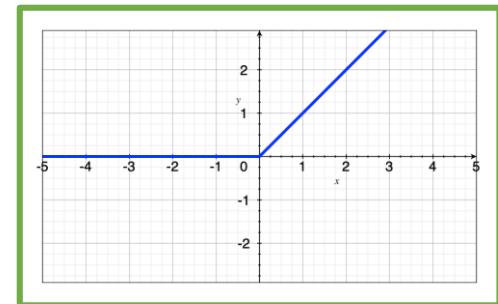


Модель с 1 нейроном

Пример 2. Функция активации F – $ReLU$ (*Rectified linear unit*)

$$ReLU(x) \triangleq \max(0, x)$$

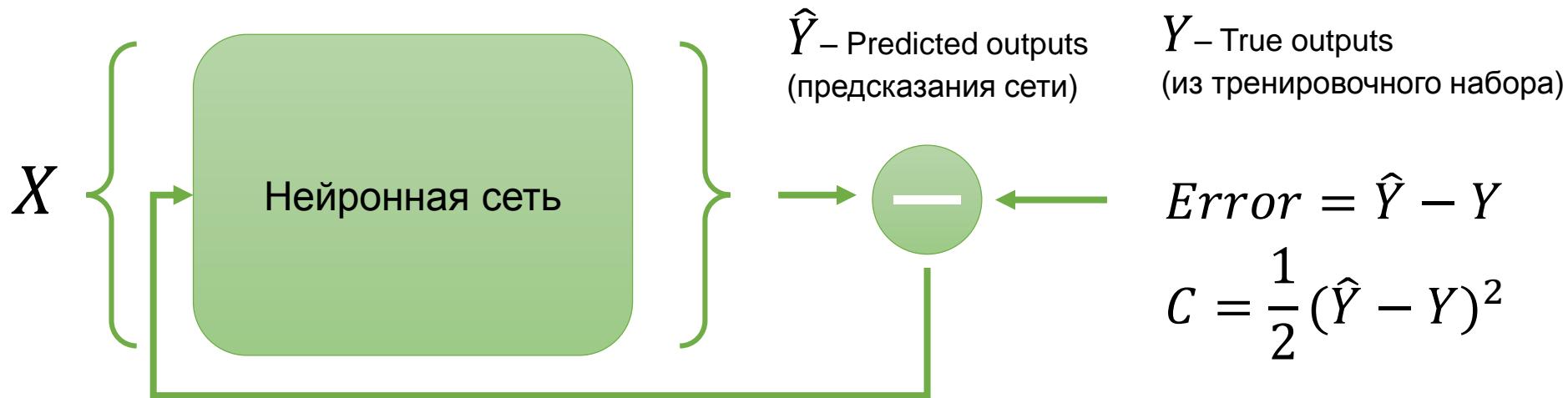
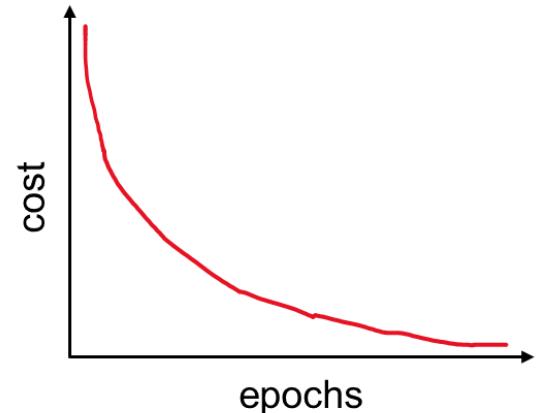
- Если $x < 0$, то $y = 0$, если $x \geq 0$, то $y = x$
- Эффективна с точки зрения вычислений
- В отличие от сигмоида, позволяет сохранить сильные градиенты, не ограничена $[0, 1]$
- Как правило, используется в скрытых слоях (hidden layers)



Модель с 1 нейроном

NN: Supervised learning

- Даны данные для обучения $\{X(n), Y(n)\}$
- Дано количество шагов обучения (epochs) до сходжения
- С (cost function) – мера расхождения между истинным значением параметра и его оценкой



Обновление весов w

\hat{Y} – Predicted outputs
(предсказания сети)

Y – True outputs
(из тренировочного набора)

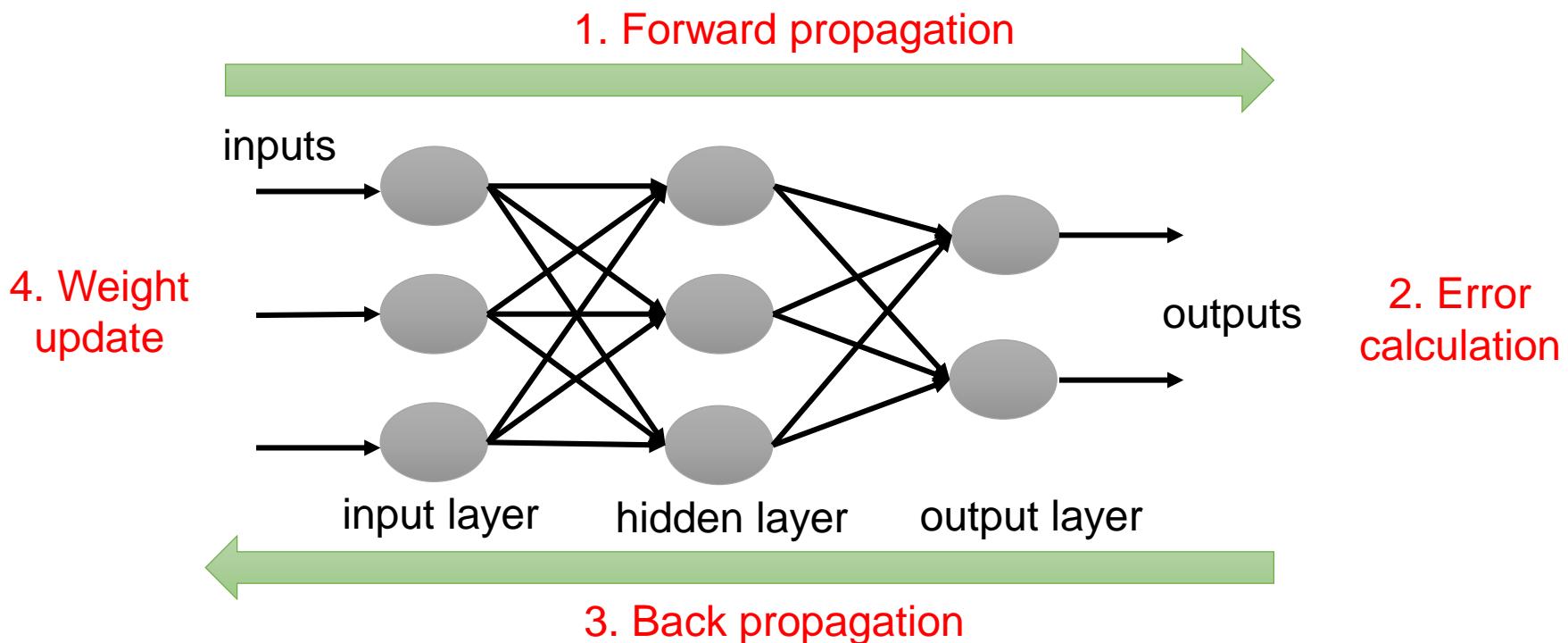
$$Error = \hat{Y} - Y$$

$$C = \frac{1}{2}(\hat{Y} - Y)^2$$

Модель с 1 нейроном

Метод обратного распространения ошибки (backpropagation):

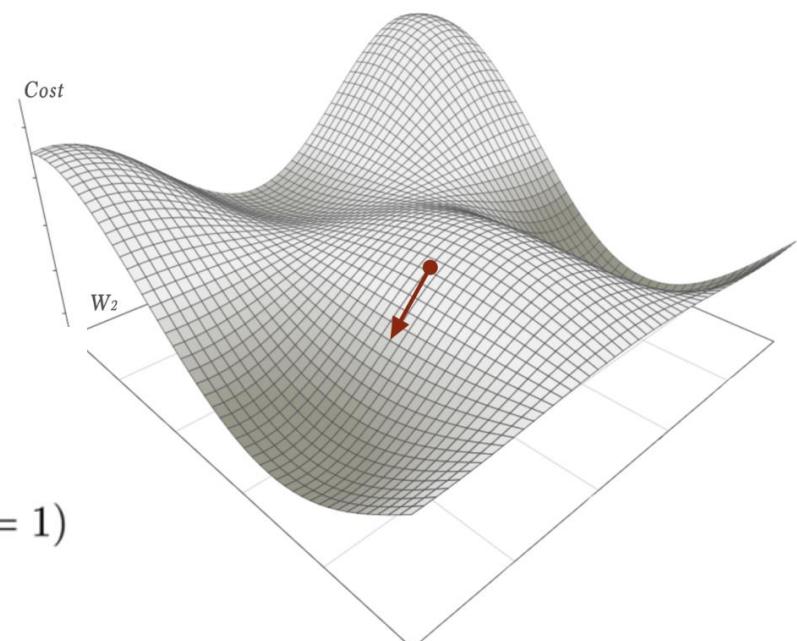
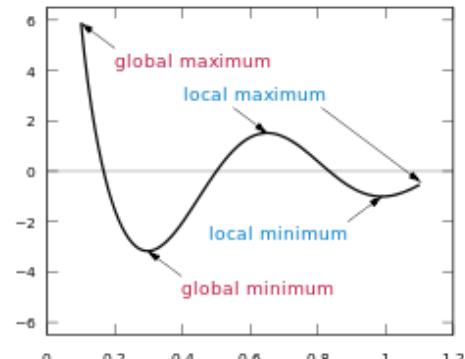
- Шаги 1, 2 – первый проход (начальные веса заданы случайным образом), вычисление ошибки первого выходного значения
- Шаги 3, 4 – расчет градиентов весов, выбор доли от градиента весов для вычитания из веса (learning rate). Чем это значение больше, тем быстрее учится нейронная сеть, чем меньше, тем точнее.



Модель с 1 нейроном

Метод градиентного спуска (gradient descend):

- Осуществляет спуск вниз по поверхности ошибки, непрерывно подстраивая веса в направлении к минимуму
- Типы градиентного спуска: пакетный (batch) и стохастический (stochastic)
- Проблема попадания в локальные минимумы
- Проблема выбора шага



Gradient descent algorithm

```
repeat until convergence {  
     $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$     (for  $j = 0$  and  $j = 1$ )  
}
```

Модель с 1 нейроном

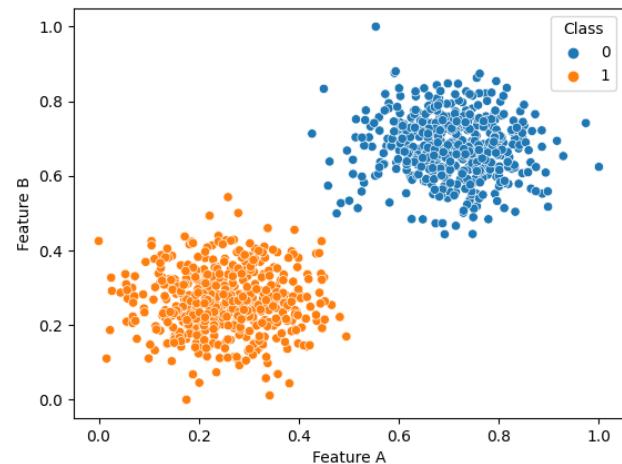
Пример задачи классификации (1 нейрон)

Данные для обучения – таблица с 3 столбцами:
Feature A, Feature B и Class.
Первые 2 параметра подаются на вход, Class –
на выход.

```
model.add(Dense(1, input_dim=2, activation='sigmoid'))
```

```
Model: "sequential"
-----
Layer (type)          Output Shape       Param #
=====
dense (Dense)         (None, 1)           3
=====
Total params: 3
Trainable params: 3
Non-trainable params: 0
-----
```

Модель содержит 3 параметра обучения:
Feature A, Feature B и bias



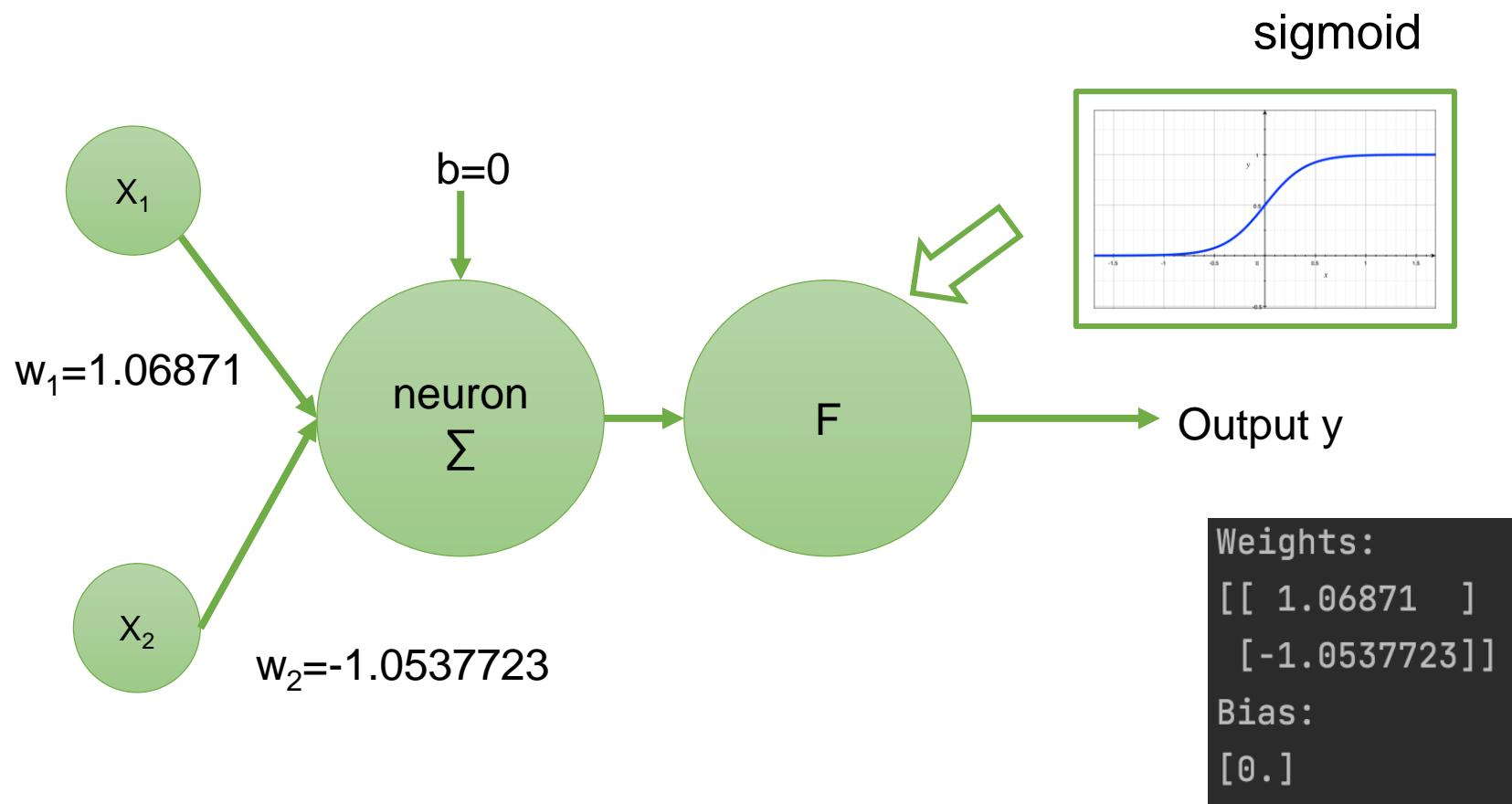
Weights:
[[1.06871]
 [-1.0537723]]
Bias:
[0.]

Начальные значения
входных параметров

Модель с 1 нейроном

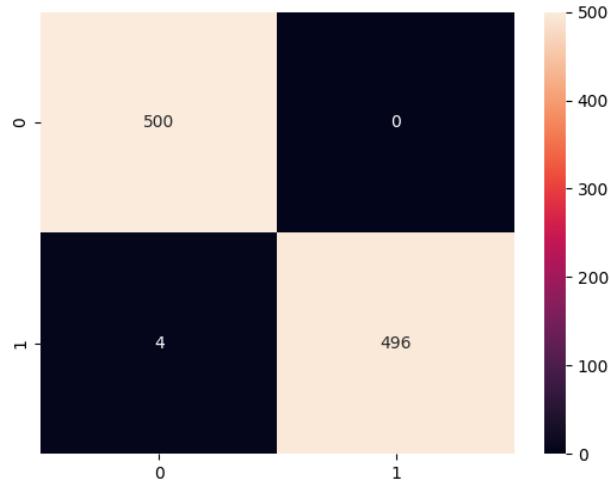
Пример задачи классификации (1 нейрон)

В результате создана модель:



Модель с 1 нейроном

Пример задачи классификации (1 нейрон)



```
Epoch 93/100  
32/32 - 0s - loss: 0.2980  
Epoch 94/100  
32/32 - 0s - loss: 0.2959  
Epoch 95/100  
32/32 - 0s - loss: 0.2938  
Epoch 96/100  
32/32 - 0s - loss: 0.2917  
Epoch 97/100  
32/32 - 0s - loss: 0.2896  
Epoch 98/100  
32/32 - 0s - loss: 0.2876  
Epoch 99/100  
32/32 - 0s - loss: 0.2855  
Epoch 100/100
```

Предсказания

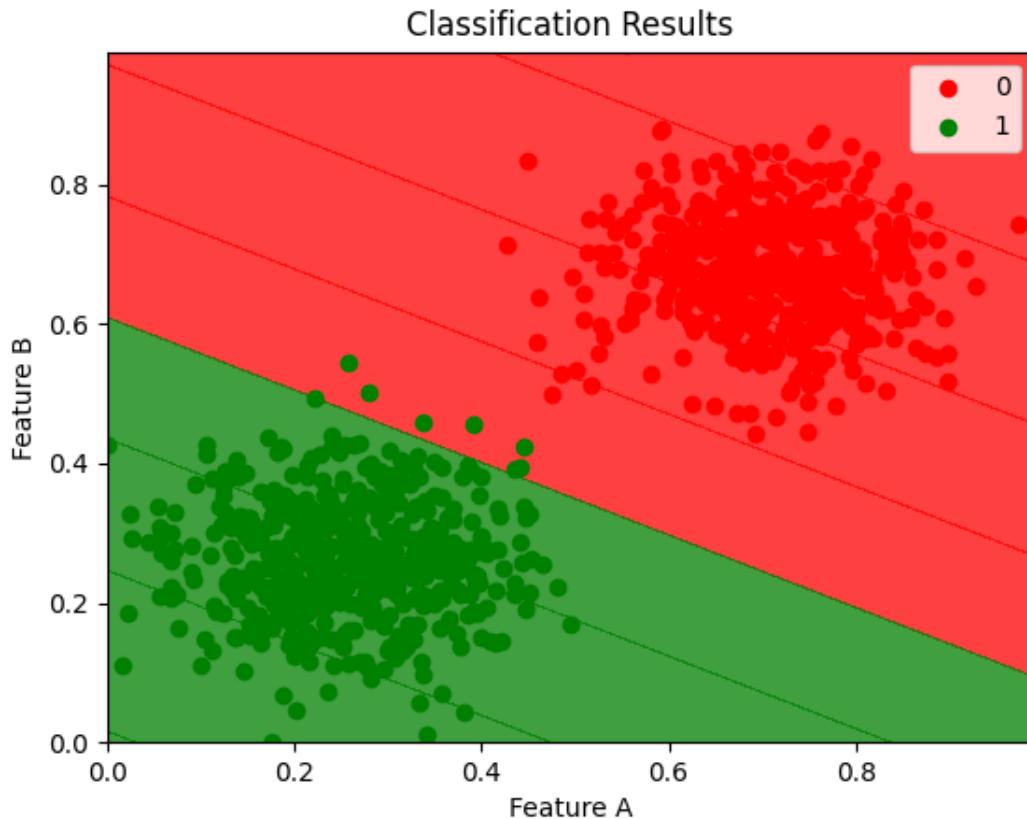
Истинные значения
(из обучающей выборки)

		+	
		+	-
+	+	True	False
	-	False	True

- финальные метрики обучения;
конечный $loss = 0.2855$ – достаточно близко к 0 (от 0 до 1 – хороший результат).

Модель с 1 нейроном

Пример задачи классификации (1 нейрон)

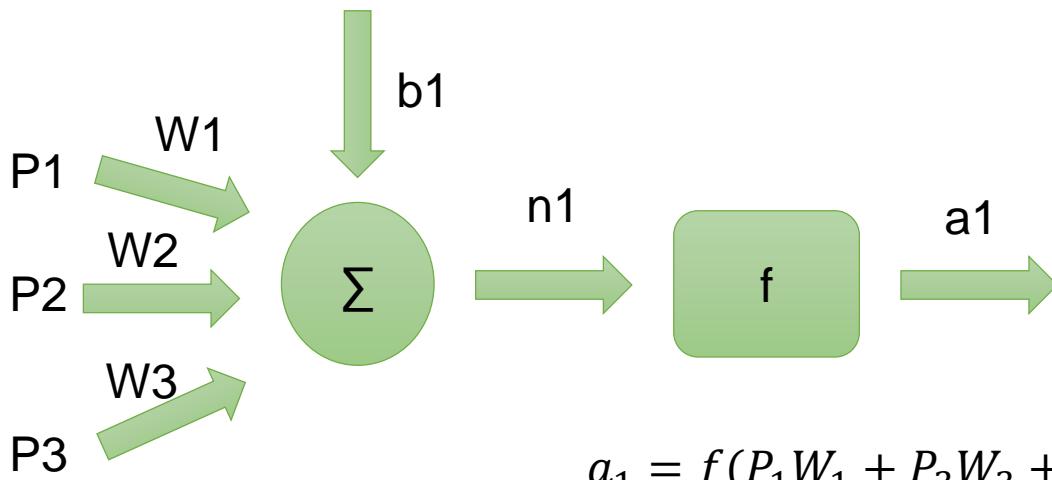


```
Final weights: [[-1.2145058]
[-2.3369439]]
Final biases: [1.423021]
```

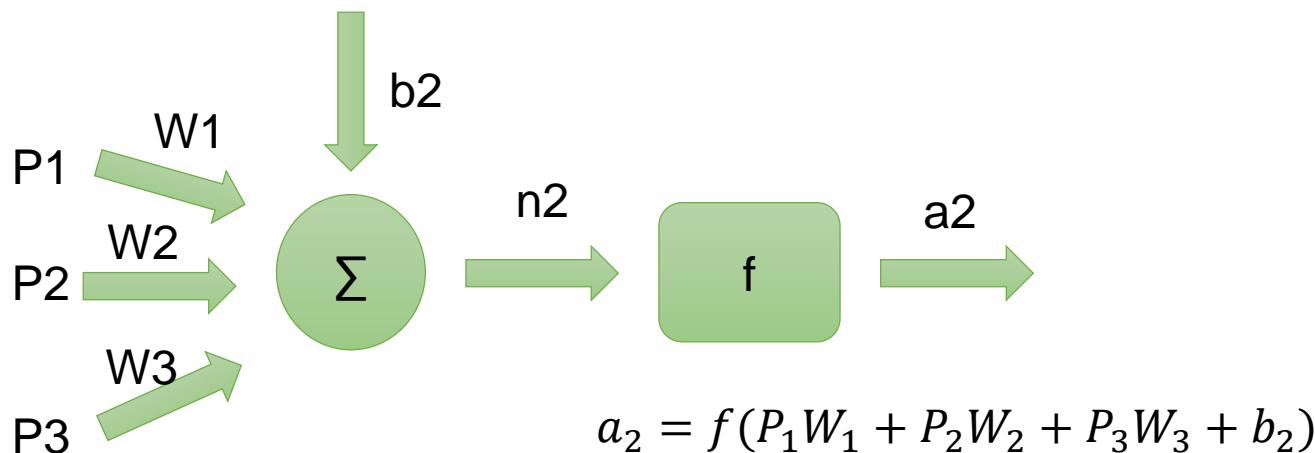
Таким образом,
произведена
классификация с
использованием всего 1
нейрона

Модель с 2 и более нейронами

Модель из двух нейронов:



- 8 параметров:
6 weights
2 biases



Модель с 2 и более нейронами

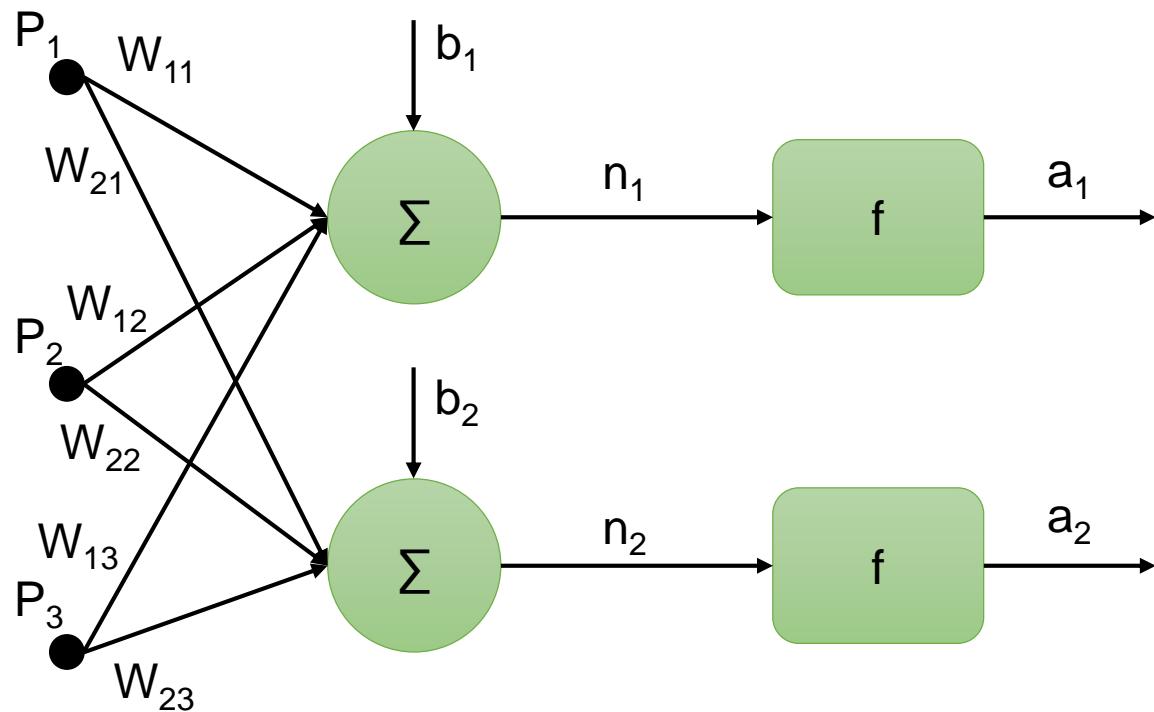
Матричное представление модели:

$$a = f(W \times P + b)$$

$$P = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

$$W = \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$



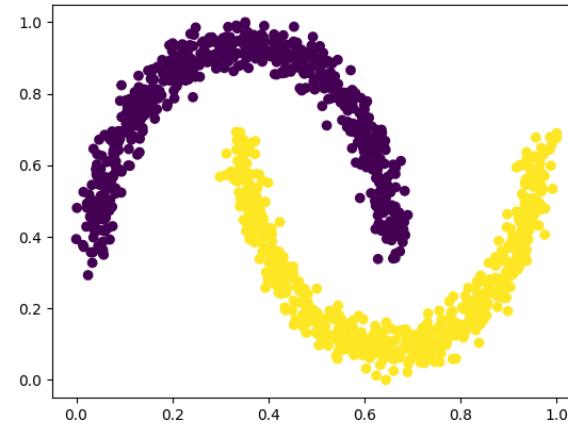
Модель с 2 и более нейронами

Пример задачи классификации (> 2 нейронов)

Набор данных:

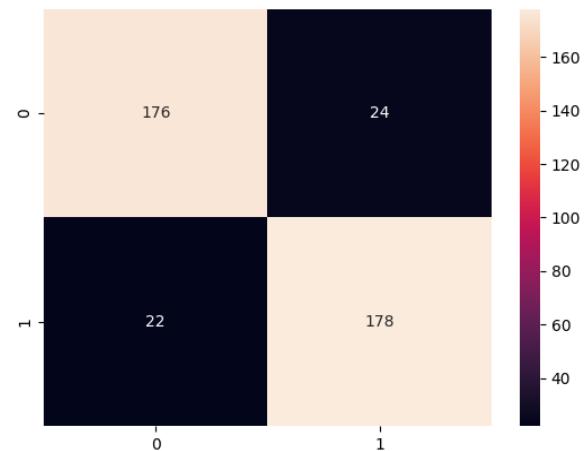
```
X: [[ 1.78966939 -0.18948138]
 [ 0.79103993 -0.52969594]
 [ 0.04339676  0.95076195]
 ...
 [ 0.25900547 -0.23609462]
 [ 1.94171606  0.15901203]
 [ 0.96271111  0.12114898]]
y: [1 1 0 ... 1 1 0]
```

Визуализация данных после нормализации



Результат обучения модели с 1 нейроном:

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1)	3
Total params:	3	
Trainable params:	3	
Non-trainable params:	0	



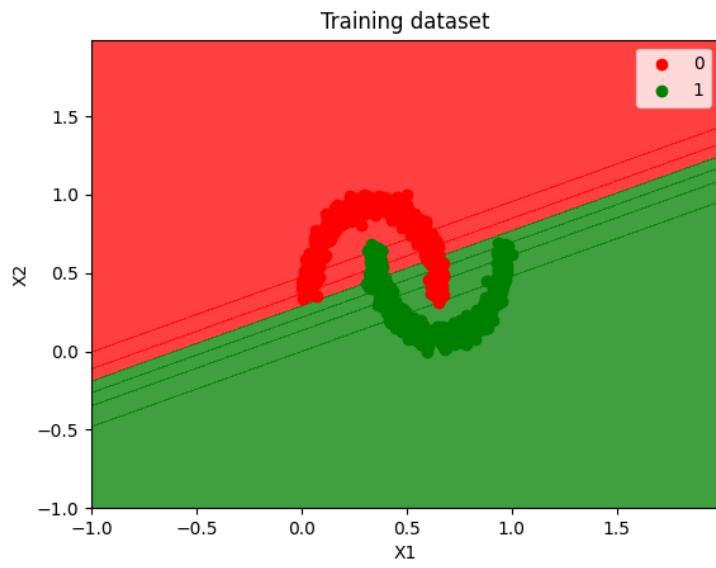
Модель с 2 и более нейронами

Пример задачи классификации (> 2 нейронов)

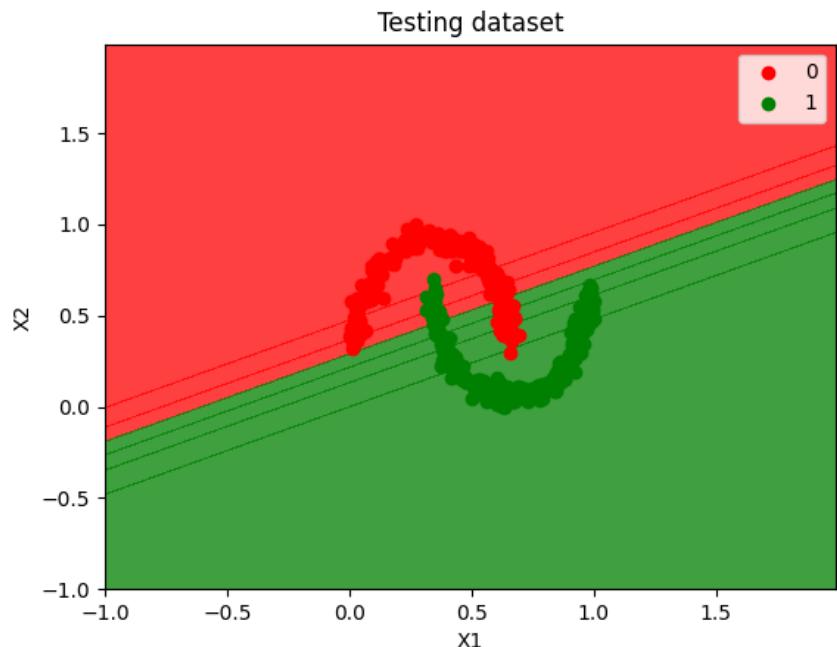
Результат обучения модели с 1 нейроном:

Очевидна проблема с поиском нелинейных закономерностей

	precision	recall	f1-score	support
0	0.89	0.88	0.88	200
1	0.88	0.89	0.89	200
accuracy			0.89	400
macro avg	0.89	0.89	0.88	400
weighted avg	0.89	0.89	0.88	400



Weights: [[3.9705787]
[-8.267138]]
Biases: [2.1937928]



Модель с 2 и более нейронами

Пример задачи классификации (> 2 нейронов)

Модель с 20 нейронами:

Model: "sequential"		
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 20)	60 (2 inputs + 1 bias) * 20 neurons = 60 params
dense_1 (Dense)	(None, 20)	420 (20 inputs * 20 neurons) + 20 biases = 420 params
dense_2 (Dense)	(None, 20)	420
dense_3 (Dense)	(None, 20)	420
dense_4 (Dense)	(None, 1)	21
Total params: 1,341		
Trainable params: 1,341		
Non-trainable params: 0		

Код для создания модели

```
model = Sequential()
# слой с 20 нейронами, 2 inputs, функция активации ReLU (ReLU keeps gradients)
model.add(Dense(20, input_dim=2, activation='relu'))
# создадим еще несколько слоев (hidden layers)
model.add(Dense(20, activation='relu'))
model.add(Dense(20, activation='relu'))
model.add(Dense(20, activation='relu'))

# output layer, 1 output, функция активации - sigmoid, так как всего 2 класса, 0 и 1!
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam')
model.fit(X_train_scaled, y_train, epochs=500, verbose=2)
```

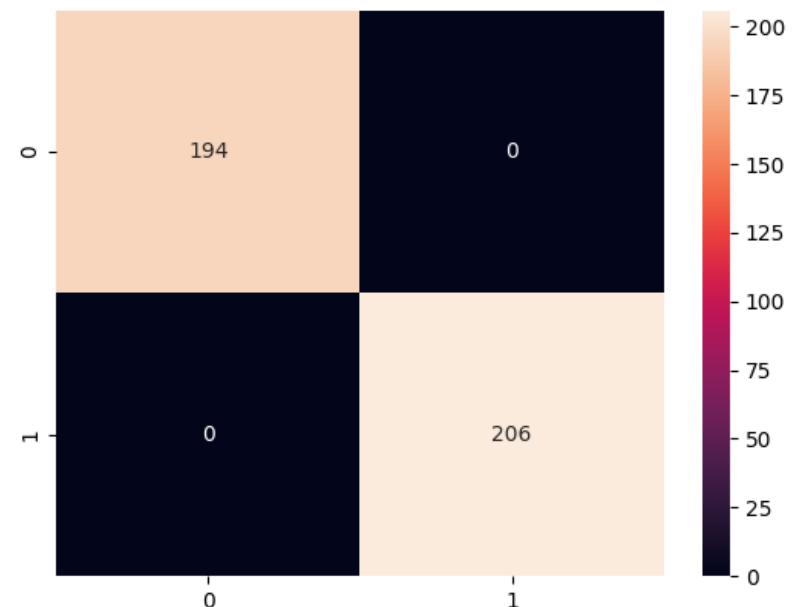
Модель с 2 и более нейронами

Пример задачи классификации (> 2 нейронов)

Модель с 20 нейронами:

```
Epoch 492/500
38/38 - 0s - loss: 7.4229e-08
Epoch 493/500
38/38 - 0s - loss: 8.3337e-08
Epoch 494/500
38/38 - 0s - loss: 6.6203e-08
Epoch 495/500
38/38 - 0s - loss: 7.0056e-08
Epoch 496/500
38/38 - 0s - loss: 7.0266e-08
Epoch 497/500
38/38 - 0s - loss: 6.2120e-08
Epoch 498/500
38/38 - 0s - loss: 8.7400e-08
Epoch 499/500
38/38 - 0s - loss: 6.5154e-08
Epoch 500/500
38/38 - 0s - loss: 6.4599e-08
Model: "sequential"
```

Последние шаги (epochs) обучения – очень хорошая точность (ошибка между предсказанием и истинным значением близка к 0)

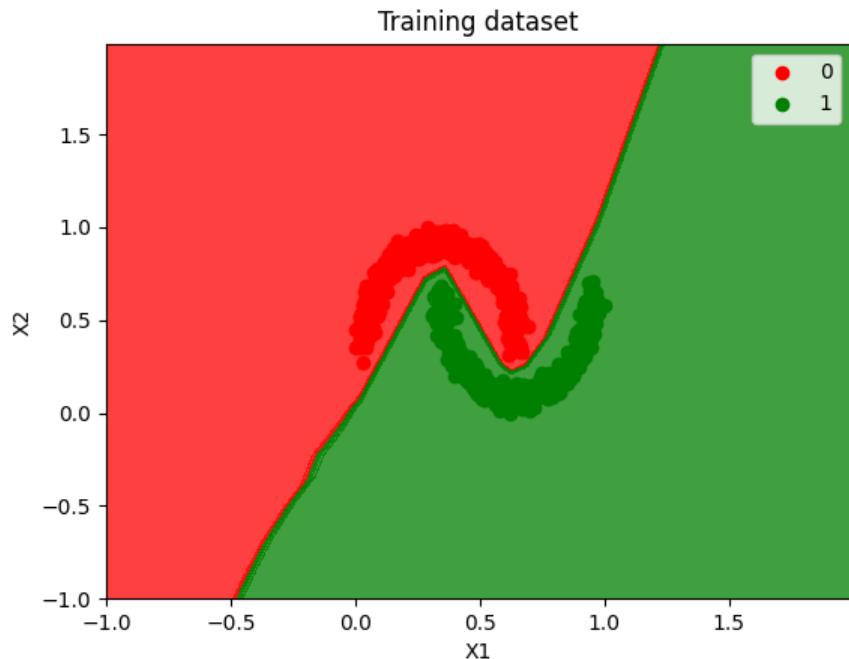


100% верных предсказаний

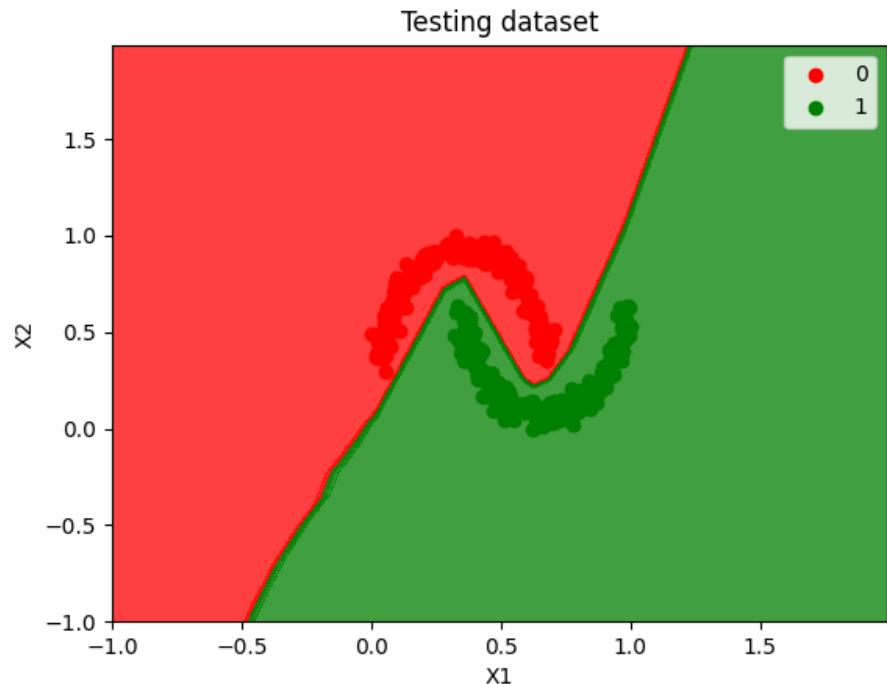
Модель с 2 и более нейронами

Пример задачи классификации (> 2 нейронов)

Модель с 20 нейронами: результат



```
Weights: [[-0.18157451 -0.48329517 -0.00817692 -0.37296087 -0.03079598 0.3712045  
0.29927674 0.46091178 -0.01598659 0.03724555 -0.16424356 -0.4481405  
0.76686907 -0.19153017 1.3170782 -0.25186026 -0.16383328 0.6739355  
0.53877664 0.24115191]  
[ 0.48988914 -0.13867095 -0.42077857 -0.03952771 0.03663329 -0.20680152  
-0.37113672 0.27266717 0.58020455 -0.10597386 0.2908407 -0.27777678  
0.09520844 -0.2367307 -0.07958822 0.08417594 0.07622404 -0.25642174  
-0.06479647 0.49631816]]  
Biases: [ 0.16664119 0. 0. 0. -0.04399716 0.30417368  
-0.07859182 -0.3264709 0.1285506 -0.02770058 0.10056048 0.  
-0.02461845 0. -0.41210383 0.29248142 0.19290875 0.13955025  
0.01855338 -0.08333492]
```



Нейронная сеть в состоянии находить сложные нелинейные закономерности!

Сверточные нейронные сети (CNN)

Classical Computer Vision

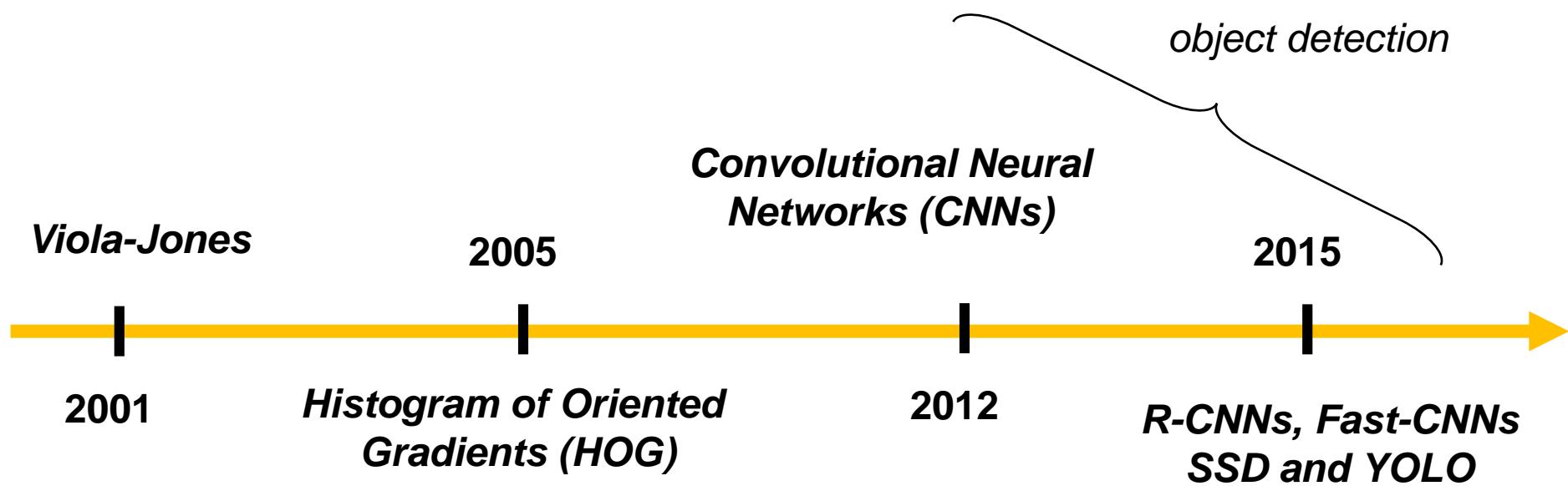
- Image Segmentation: Blob Analysis, Color Thresholding.
- Feature-based object detection (Scan and Match): HOG, SURF, etc.

Machine Learning

- Support Vector Machines (SVM)
- Viola-Jones Algorithms

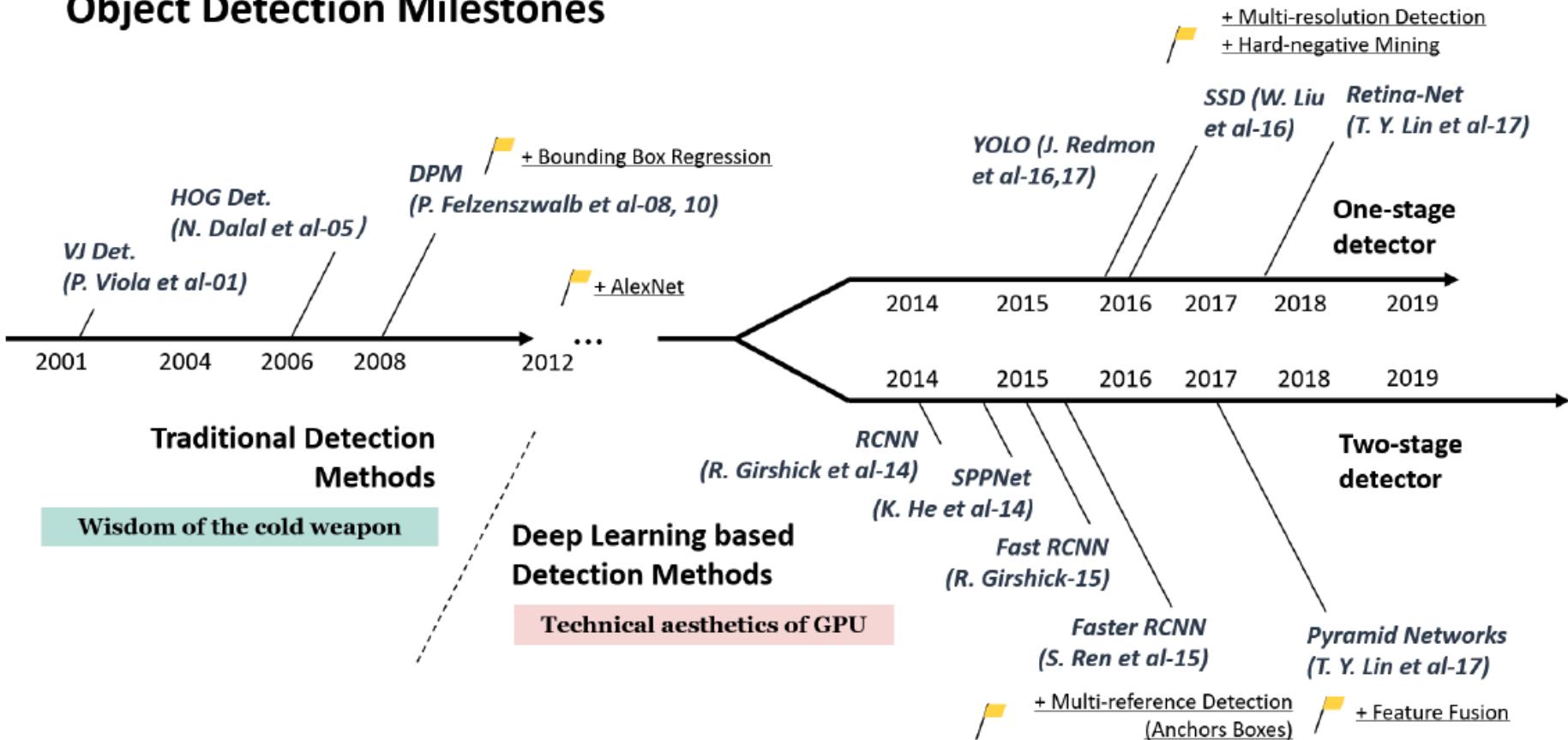
Deep Learning

- Convolutional Neural Networks: R-CNN, YOLO, SSD, etc



Сверточные нейронные сети (CNN)

Object Detection Milestones



Сверточные нейронные сети (CNN)

Типы задач компьютерного зрения:

- **Classification**

классификация изображения по типу объекта, которое оно содержит

- **Semantic segmentation**

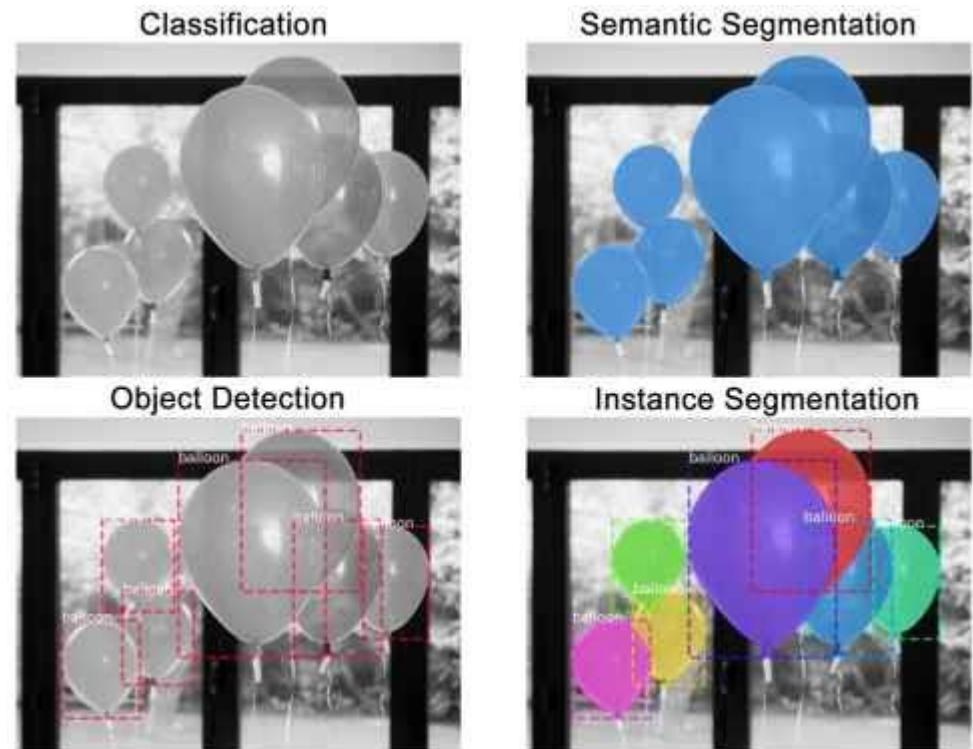
определение всех пикселей объектов определённого класса или фона на изображении.

- **Object detection**

обнаружение всех объектов указанных классов и определение охватывающей рамки для каждого из них

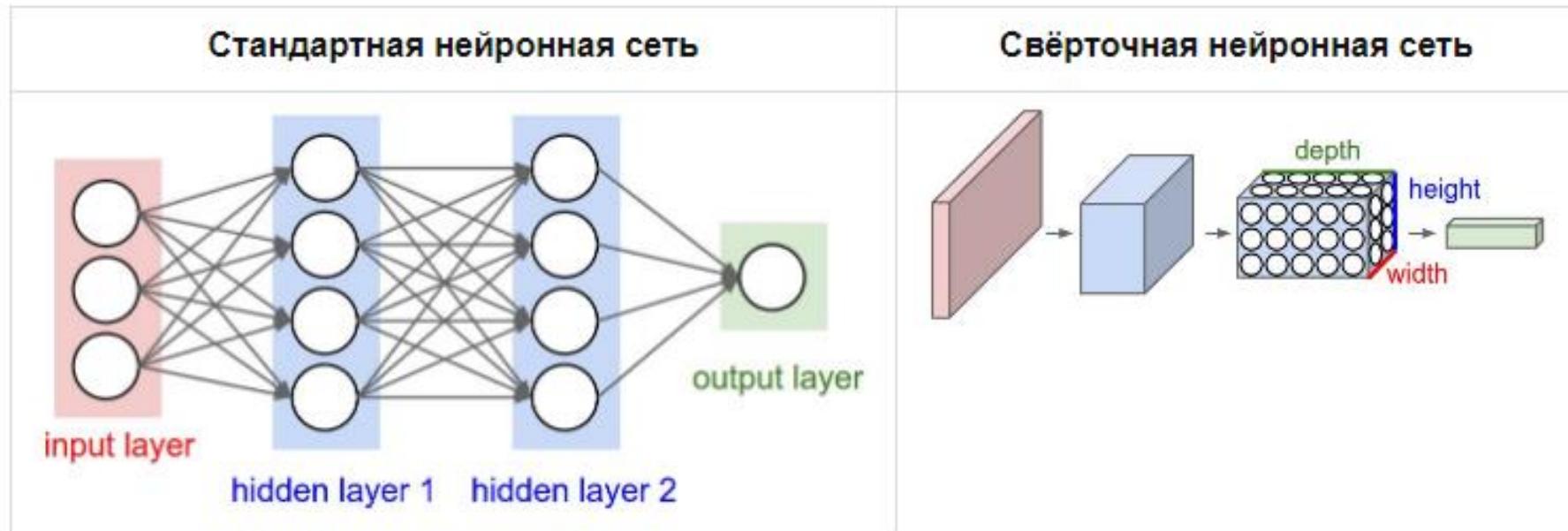
- **Instance segmentation**

определение пикселей, принадлежащих каждому объекту каждого класса по отдельности



Сверточные нейронные сети (CNN)

- CNN получили широкое распространение с развитием графических процессоров (в 2010-х)
- Позволяют находить объекты на изображениях
- Используются окна опроса (sliding windows), аналогично операции свёртки, которые подаются на вход нейронной сети.
- В отличие от обычных нейронных сетей, слои в свёрточной нейронной сети располагают нейроны в 3 измерениях — ширине, высоте, глубине



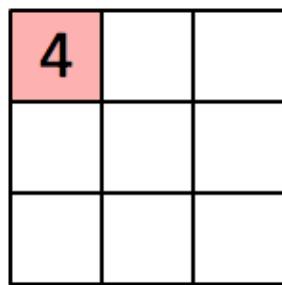
Сверточные нейронные сети (CNN)

CNN: операция свертки

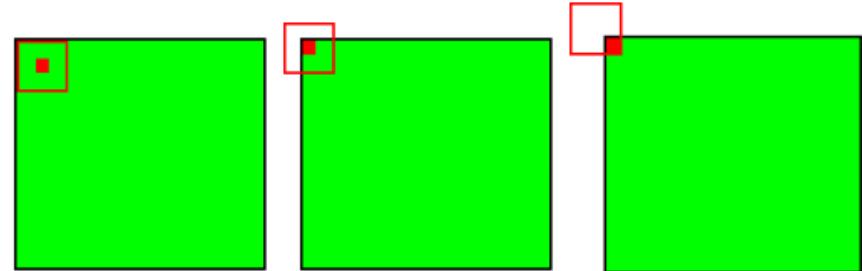
Свёртка делается, чтобы извлечь высокоуровневые признаки (например, края и линии входного изображения).

1 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	0	0
0 <small>x0</small>	1 <small>x1</small>	1 <small>x0</small>	1	0
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	1
0	0	1	1	0
0	1	1	0	0

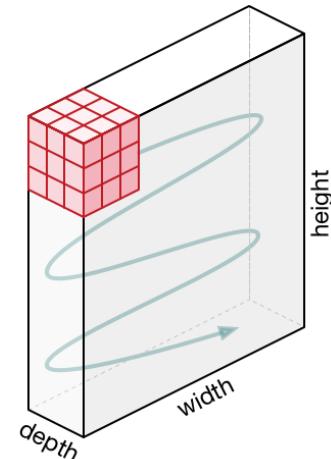
Image



Convolved Feature

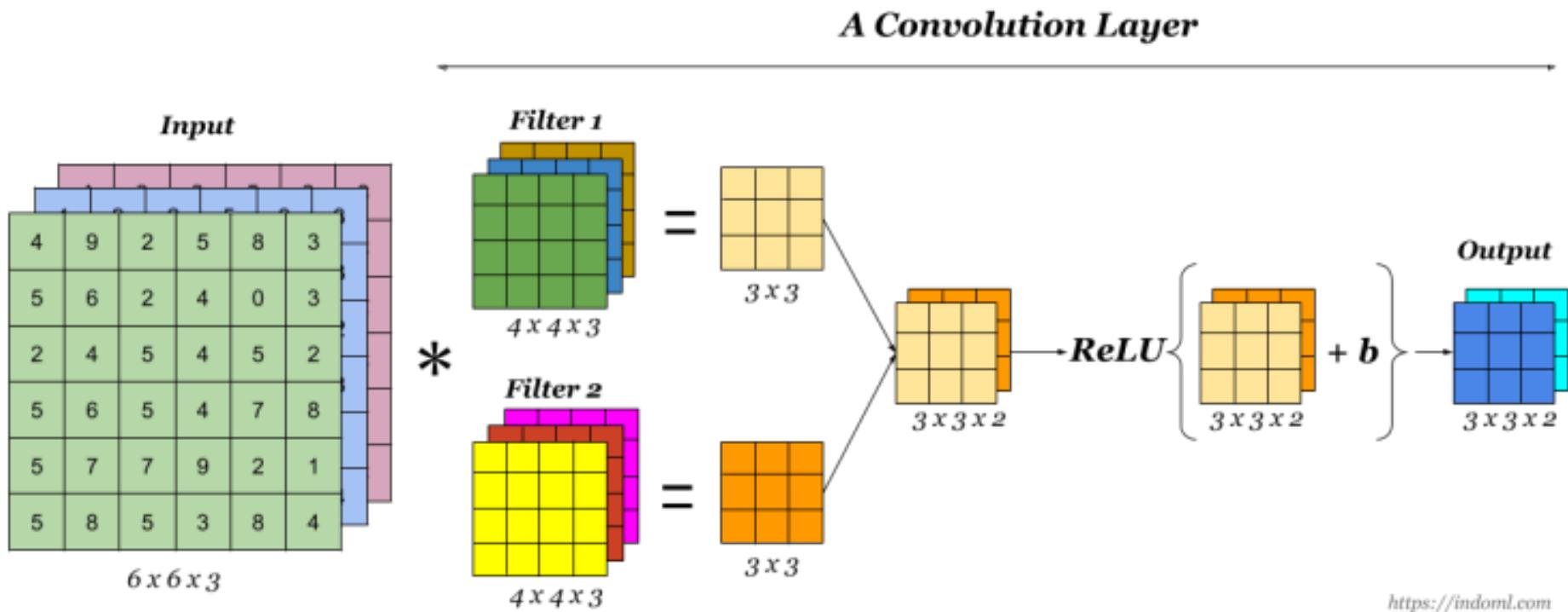


методы обработки краев исходной матрицы: *valid*, *same*, *full*



Сверточные нейронные сети (CNN)

CNN: сверточные слои

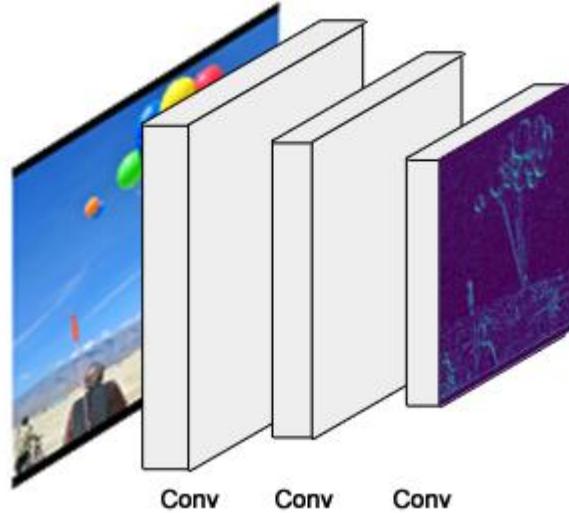


пример 1 сверточного слоя с двумя фильтрами

Сверточные нейронные сети (CNN)

CNN: сверточные слои

При добавлении слоев архитектура CNN постепенно адаптируется к высокоуровневым признакам



признаки более высокого уровня

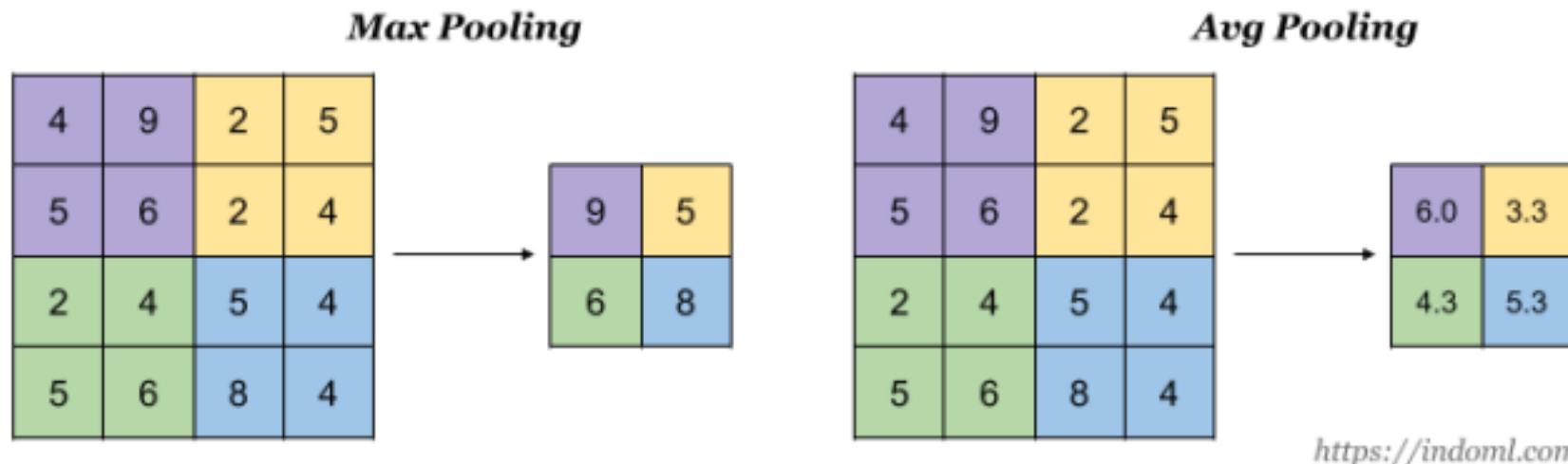
$$\begin{aligned} q &= \text{O}_1 + \text{I}_2 \\ 4 &= \text{I}_3 + \text{C}_4 + \text{I}_2 \\ \hline \text{O}_1 &= \text{O}_1 + \text{C}_2 + \text{I}_3 + \dots \\ \text{C}_1 &+ \text{C}_2 + \text{I}_3 + \dots \end{aligned}$$

низкоуровневые
признаки

Сверточные нейронные сети (CNN)

CNN: слои объединения (max pooling)

Цель слоя – уменьшение размерности карт предыдущего слоя.

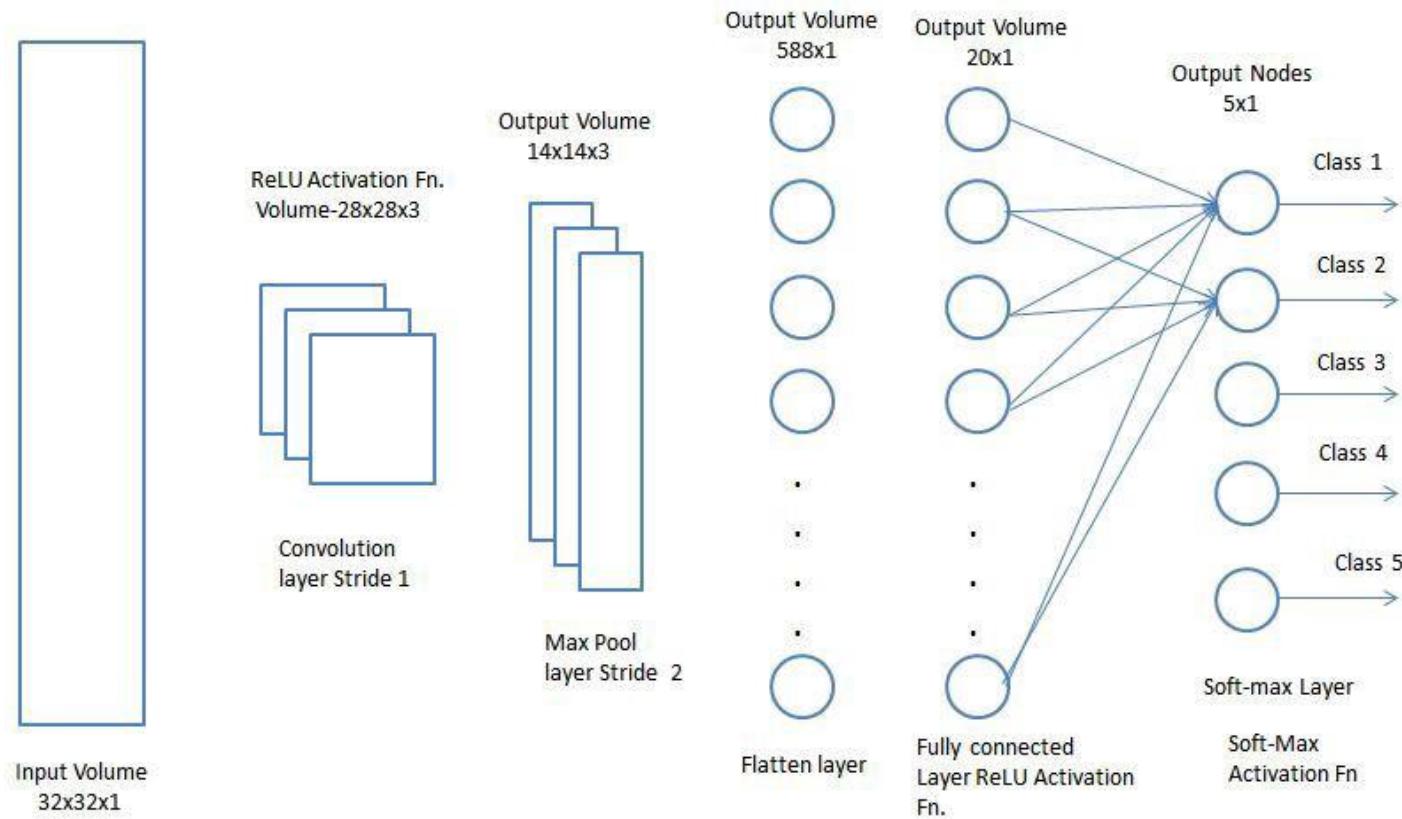


Данный слой снижает чувствительность к точному пространственному расположению признаков изображения

Сверточные нейронные сети (CNN)

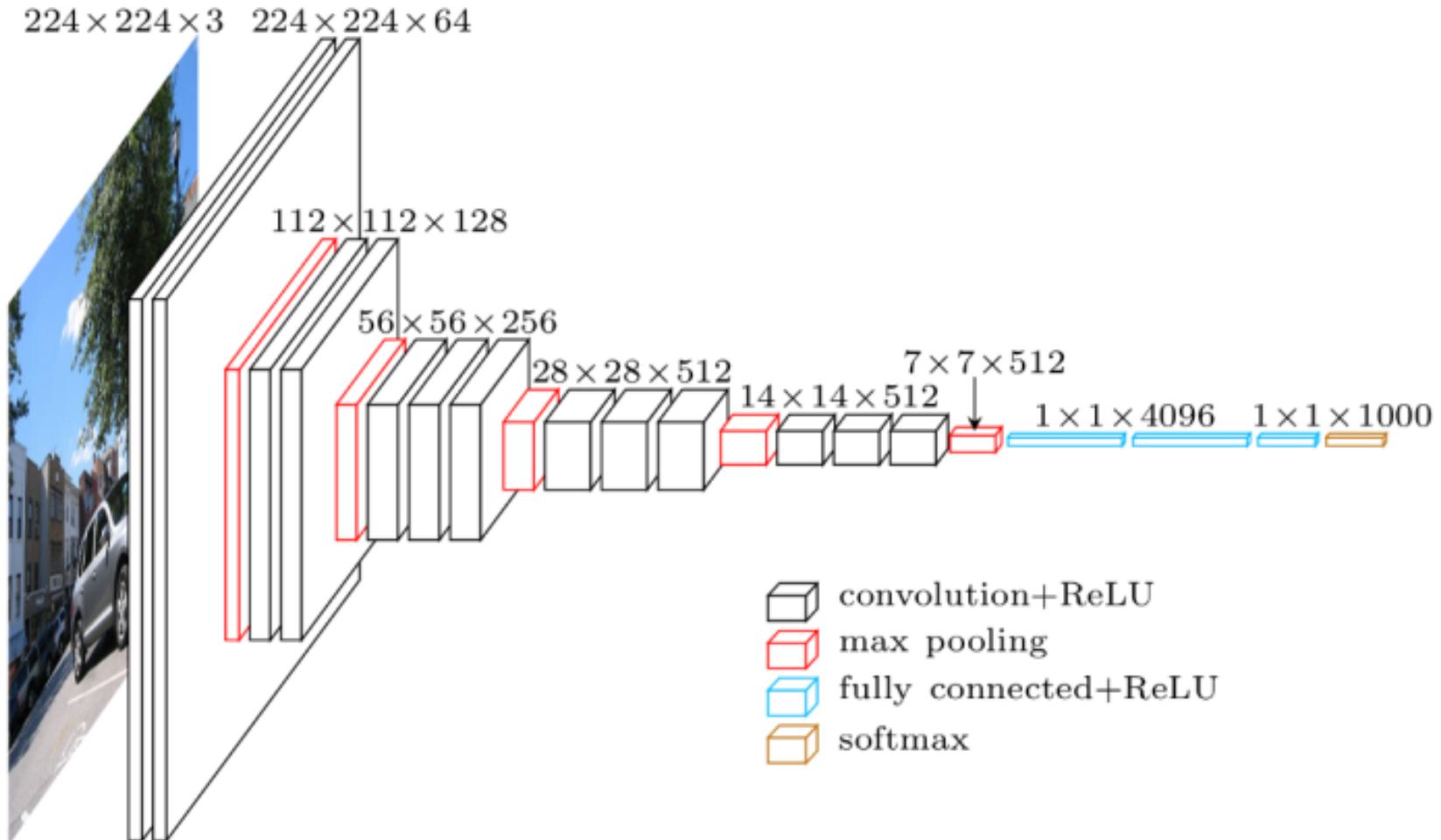
CNN: полносвязный слой

Карта признаков, созданная сверточным слоем, сводится к вектору, который подается на полностью связанный слой, так что он захватывает сложные отношения между объектами высокого уровня.



Сверточные нейронные сети (CNN)

классическая CNN: VGG-16 (2014)



Сверточные нейронные сети (CNN)

Проблема: «лишние» вычисления при проходе окна опроса по всему изображению, т.к. не все области искомый объект.

Решение: **R-CNNs (Region Based Convolutional Neural Networks)**.

Они позволяют значительно ускорить время распознавания объектов на изображениях (2015-2016 гг).

- R-CNN
- Fast R-CNNs
- Faster R-CNNs

Библиотека
Detection (Facebook)

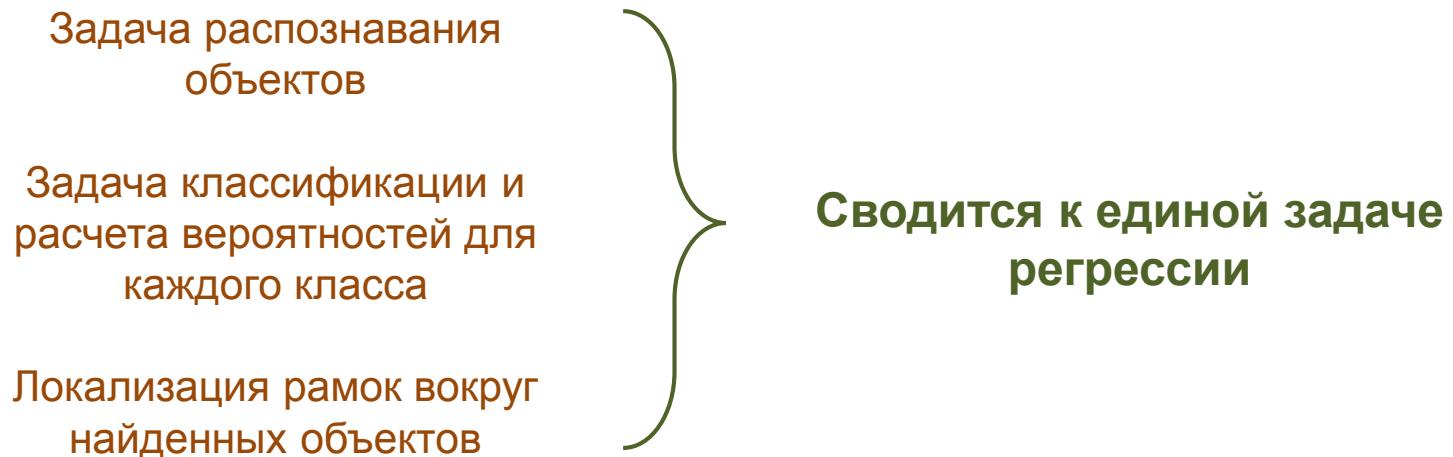


Алгоритм You Only Look Once (YOLO)

Алгоритм YOLO был изобретен в 2015 г.

Он решает несколько основных проблем с R-CNN:

- R-CNNs очень точные, но работают слишком медленно, что не позволяет их использовать для распознавания в реальном времени
- Для работы с R-CNN необходимо сначала обучить дополнительные компоненты алгоритма: CNN, SVM, линейная регрессия для окон опроса и др.



Алгоритм You Only Look Once (YOLO)

- Разделение изображения на сетку размером $S \times S$ (как правило, размером 13×13 или 19×19 пикселей)
- Каждая ячейка может обнаружить до B рамок или частей рамок (как правило $B = 5$)
- Каждому объекту соответствует своя рамка с координатами центра (x, y)
- Таким образом, каждая ячейка предсказывает до B рамок и вероятностей для каждого класса C
- Класс C характеризует вероятность того, что данная рамка соответствует объекту определенного класса



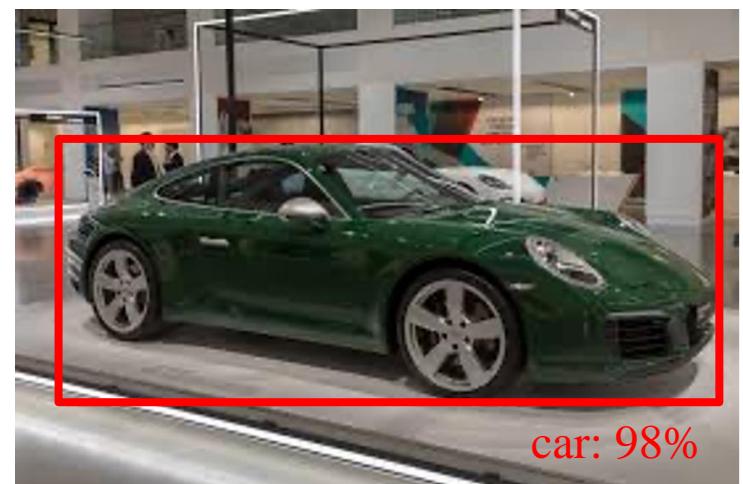
Алгоритм You Only Look Once (YOLO)

Алгоритм YOLO предсказывает следующие параметры:

(x , y , w , h , confidence, class)

- (x , y) – координаты центра рамки относительно размера ячейки
- (w , h) – размеры рамки относительно размера изображения
- Все значения, как правило, нормализованы и находятся в диапазоне [0, 1]

Обучение YOLO проводилось на наборе
данных **PASCAL VOC**
(<http://host.robots.ox.ac.uk/pascal/VOC/>)



Алгоритм You Only Look Once (YOLO)

Вероятность обнаружения
объекта данного класса

Intersection over Union

$$\text{confidence} = \overbrace{p(\text{object})}^{\text{предсказание}} \times \overbrace{\text{IoU}(\text{prediction}, \text{ground truth})}^{\text{истинное положение рамки (из обучающей выборки)}}$$

$$\text{IoU} = \frac{\text{Площадь пересечения рамок}}{\text{Общая площадь рамок}}$$

- **IoU** близко к 0: ошибка большая, неточные предсказания
- **IoU** близко к 1: ошибка незначительна, точные предсказания



Алгоритм You Only Look Once (YOLO)

Алгоритм YOLO: быстрое кодирование

- Количество классов может быть различным, но как правило их 20 (в соответствии с PASCAL VOC dataset); то есть, нейросеть может распознавать 20 типов различных объектов
- Используется бинарная категоризация.
Каждый класс мы можем представить в виде *one-hot encoding*: одномерный массив из 20 элементов – нулей и единиц, где 0 - отсутствие класса, 1 – наличие.
- Вектор предсказания содержит 25 значений:

1 1 1 1 1 20
(x , y , w , h , confidence, class)

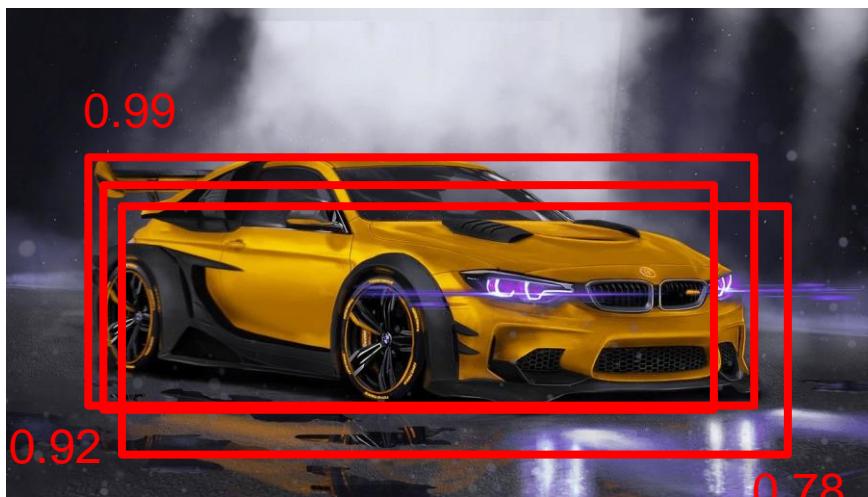
- Общее число предсказаний для всего изображения:

$$S \times S \times (B \times \text{predictedVectorSize}) + S \times S \times C$$

Алгоритм You Only Look Once (YOLO)

Алгоритм YOLO: Non-maximum suppression (NMS)

- Одна из часто встречающихся проблем при распознавании объектов – обнаружение одного и того же объекта несколько раз. NMS позволяет решить эту проблему.
 - Алгоритм NMS находит рамку с наибольшей достоверностью
 - Удаляет оставшиеся рамки с большим значением IoU (около 1).



Алгоритм You Only Look Once (YOLO)

Алгоритм YOLO: Anchor boxes

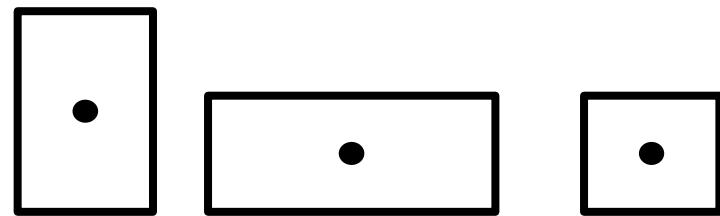
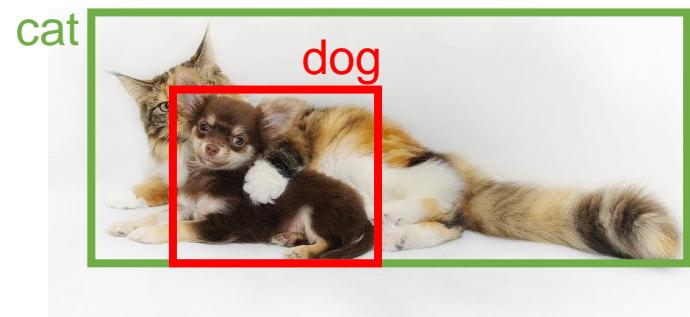
- Проблема: каждая ячейка может обнаруживать только 1 объект. Что делать, если центры (x, y) 2-х и более объектов находятся в одной ячейке?
- Решение: каждой ячейке присваивается несколько anchor boxes (в предположении, что различные объекты имеют разные формы) .

$$y = \begin{bmatrix} x \\ y \\ w \\ h \\ \text{conf} \\ \text{class} \end{bmatrix}$$

1 ячейка без
использования anchor
boxes

$$y = \begin{bmatrix} x \\ y \\ w \\ h \\ \text{conf} \\ \text{class} \\ x \\ y \\ w \\ h \\ \text{conf} \\ \text{class} \end{bmatrix}$$

для 2-х anchor
boxes в одной
ячейке

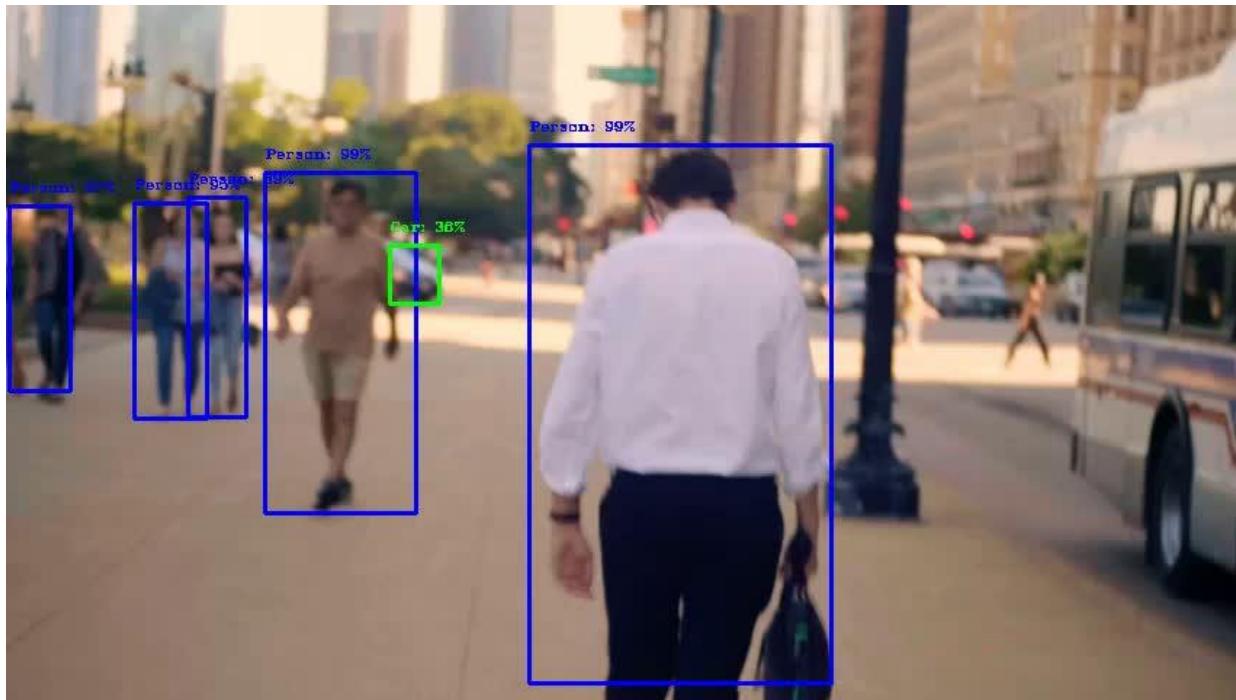


примеры форм anchor boxes

Алгоритм You Only Look Once (YOLO)

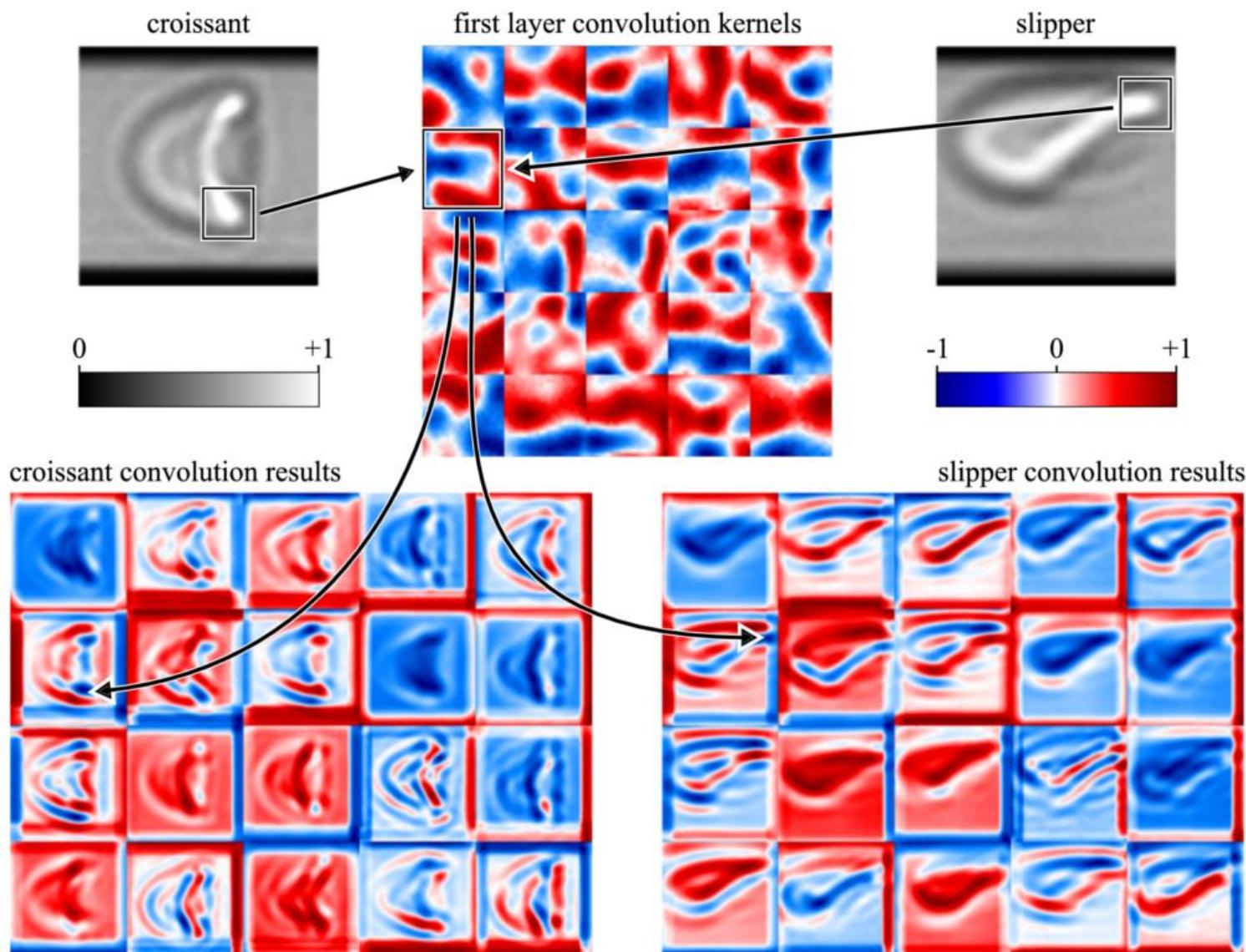
Алгоритм YOLO: практика

- Скачиванием предобученную модель и файл с настройками с сайта <https://pjreddie.com/darknet/yolo/>
`neural_network = cv2.dnn.readNetFromDarknet('yolov3.cfg', 'yolov3.weights')`
- Нейронная сеть обучена на базе данных COCO: <https://cocodataset.org>



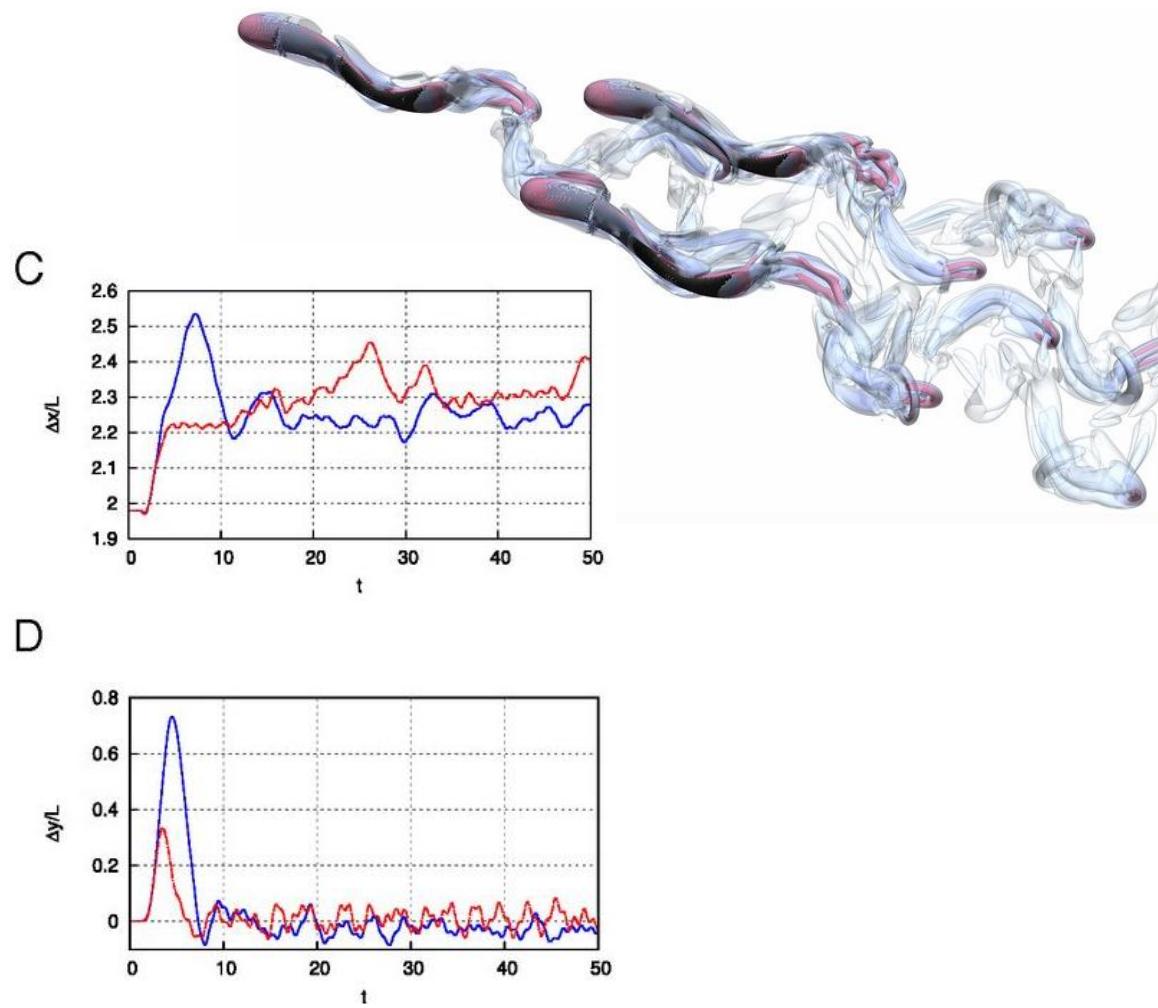
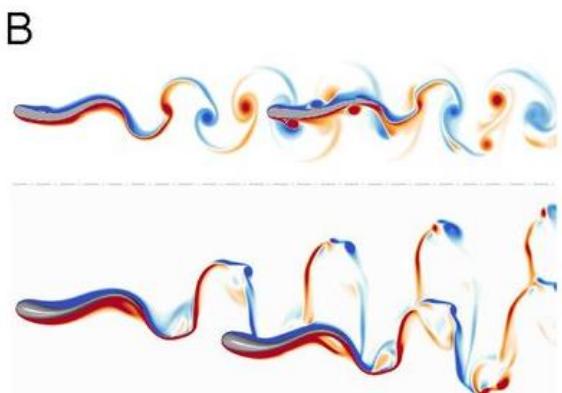
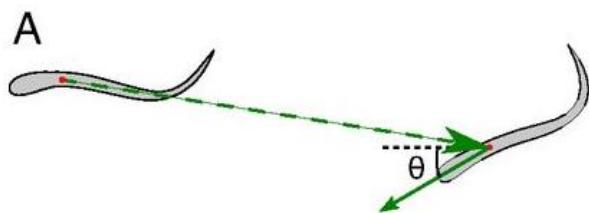
Применение нейронных сетей в анализе изображений в физическом эксперименте

Нейронные сети в физическом эксперименте



Kihm A, Kaestner L, Wagner C, Quint S (2018) Classification of red blood cell shapes in flow using outlier tolerant machine learning. *PLoS Comput Biol* 14(6): e1006278.

Нейронные сети в физическом эксперименте



Verma S, Novati G, Koumoutsakos P. 2018. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *PNAS* 115:5849–54

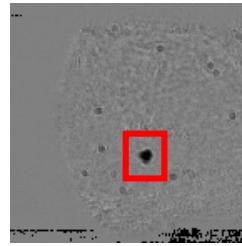
Нейронные сети в физическом эксперименте

CoreML: настройка модели для течения в ударной трубе

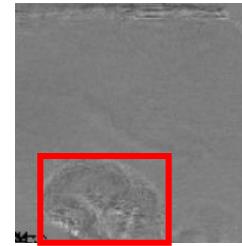
классы изображений



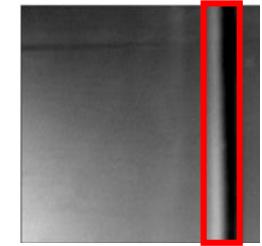
"empty"



"particle"



"plume"



"shock"

Ввод

Image (Color 416 × 416)

iou_threshold : 0.45

maximum allowed overlap (as intersection-over-union ratio) for any pair of output bounding boxes

confidence_threshold: 0.25

minimum confidence score for an output bounding box

Выход

Confidence:

MultiArray (Double N × 3)
Boxes × Class confidence

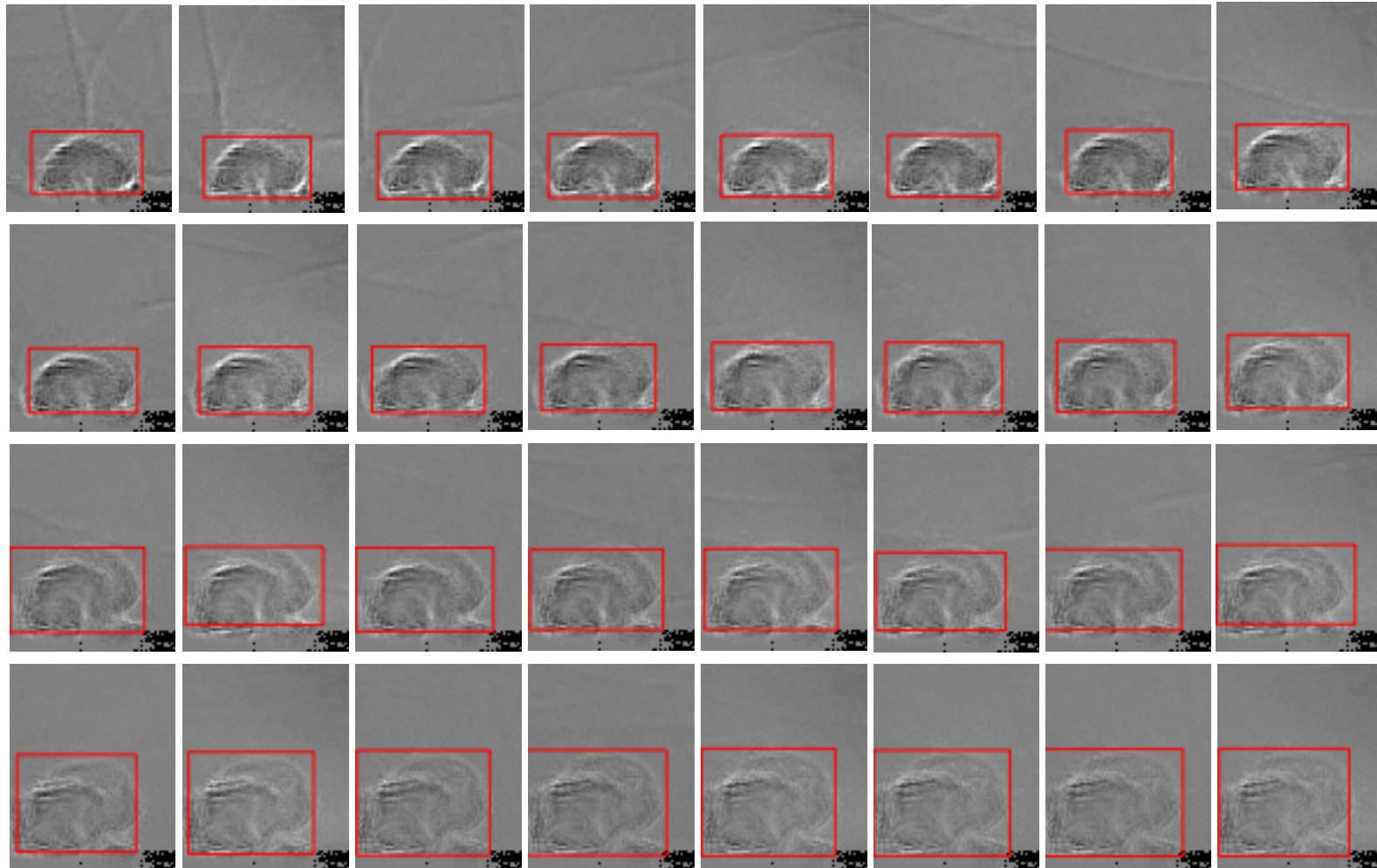
Coordinates:

MultiArray (Double N × 4)
Boxes × [x, y, width, height] (relative to image size)

N – количество распознанных объектов

Нейронные сети в физическом эксперименте

CoreML: обработка всплыивания термика

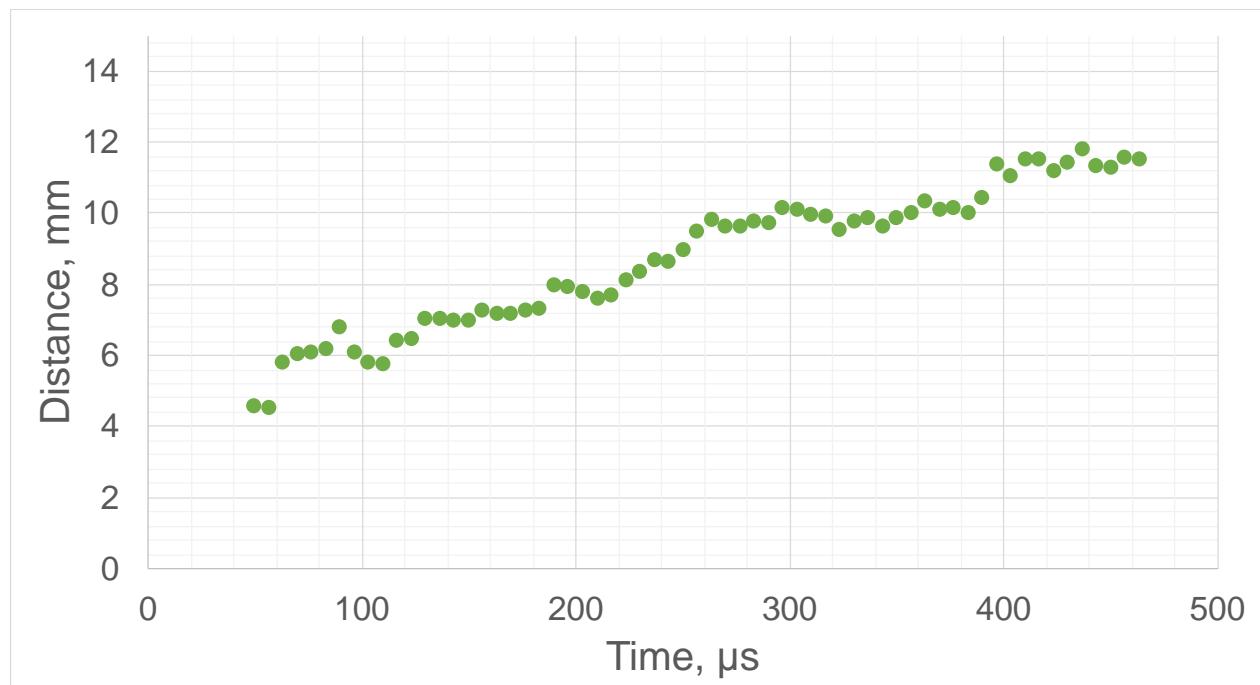


Нейронные сети в физическом эксперименте

CoreML: обработка всплывания конвективного термика

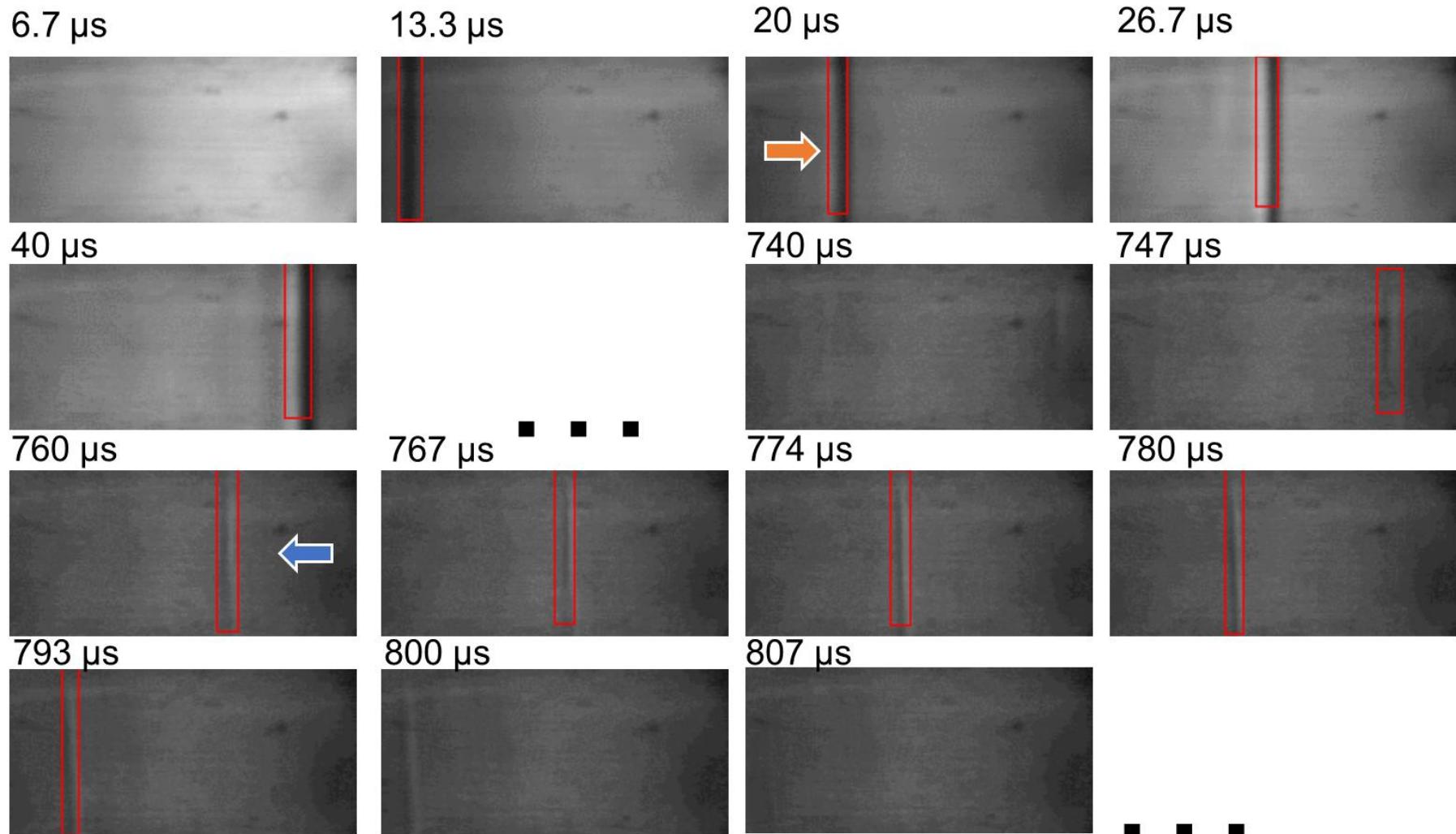


$P = 93$ Торр;
150000 кадров / с



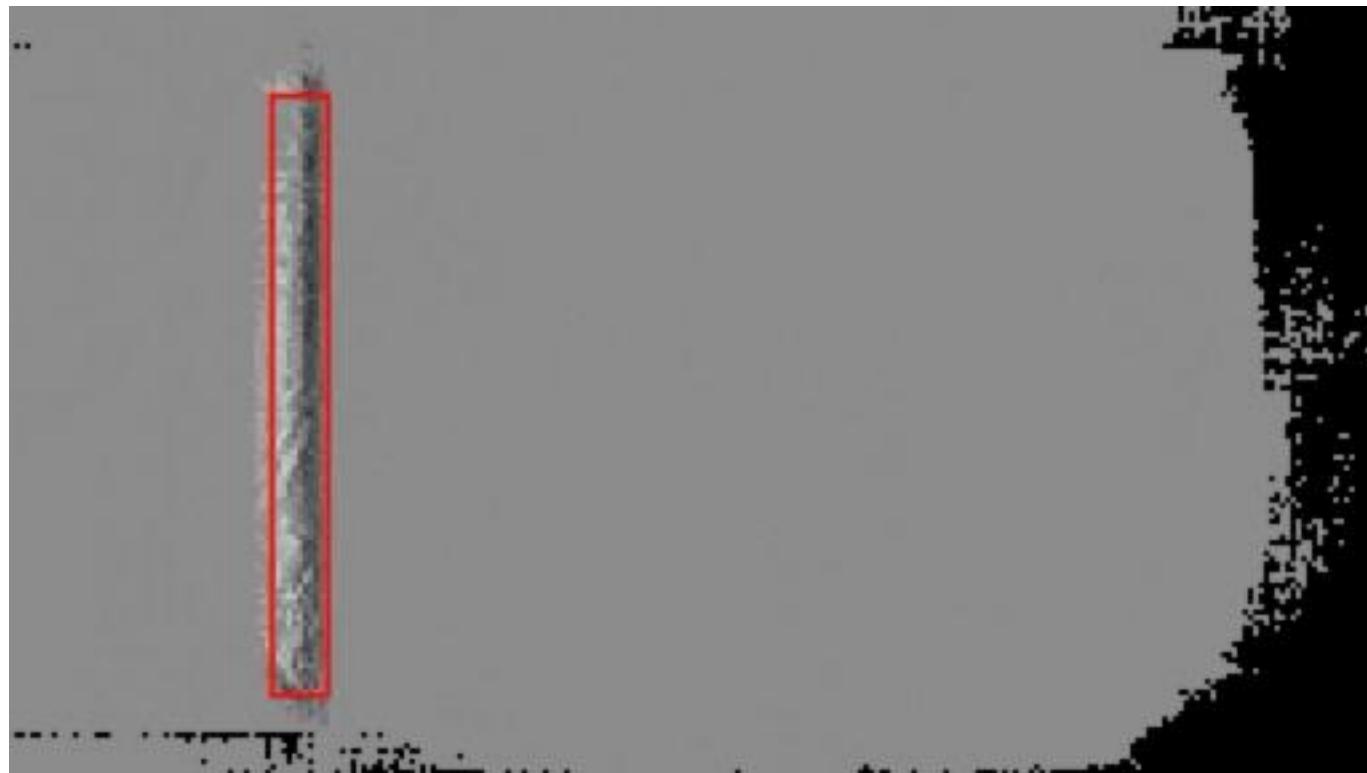
Нейронные сети в физическом эксперименте

CoreML: обработка прямой и отраженной ударной волны



Нейронные сети в физическом эксперименте

CoreML: обработка прямой и отраженной ударной волны



Некоторые ссылки:

- <https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46>
- <https://arxiv.org/pdf/1506.02640.pdf>
- https://vas3k.ru/blog/machine_learning/
- <https://indoml.com/>
- <https://arxiv.org/pdf/1505.06798.pdf>
- <https://image-net.org/>
- <http://host.robots.ox.ac.uk/pascal/VOC/>
- <https://arxiv.org/pdf/1311.2524.pdf>
- <https://www.section.io/engineering-education/understanding-loss-functions-in-machine-learning/>
- <https://pjreddie.com/darknet/yolo/>
- <https://cocodataset.org>