Титульный лист материалов по дисциплине

(заполняется по каждому виду учебного материала)

ДИСЦИПЛИНА	**Технологии извлечения знаний из больших данных** (полное наименование дисциплины без сокращений)
ИНСТИТУТ	ИКБ
КАФЕДРА	**Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»** полное наименование кафедры)
ВИД УЧЕБНОГО МАТЕРИАЛА	**Лекция** (в соответствии с пп.1-11)
ПРЕПОДАВАТЕЛЬ	**Никонов В.В.** (фамилия, имя, отчество)
СЕМЕСТР	**3 семестр 2023/2024 уч. года** (указать семестр обучения, учебный год)

Глубокое обучение и нейросети

Определение нейронной сети

Что такое нейронная сеть и как она работает

Идея искусственных нейронныхсетей пришла из верхнеуровневого понимания работы человеческого мозга.

Человеческий мозг – удивительно сложный объект, способный решать очень большое количество задач одновременно. В нем около 8.6×10^{10} нейронов, но ещё больше ($\sim1.5\times10^{14}$) связей между ними.

Нейрон представляет из себя элемент, который вычисляет выходной сигнал (по определенному правилу) из совокупности входных сигналов. То есть основная последовательность действий одного нейрона такая:

0. Прием сигналов от предыдущих элементов сети
1. Комбинирование входных сигналов
2. Вычисление выходного сигнала
3. Передача выходного сигнала следующим элементам нейронной сети

Модель

Рассмотрим пример простейшей математической модели естественного нейрона. На рисунке 1 ниже в центре размещен рассматриваемый нейрон, а слева изображены нейроны, которые посылают в него входной сигнал. Каждый из четырёхнейронов слева посылает сигнал, обозначаемый x_1, x_2, x_3, x_4. Каждый сигнал умножается на коэффициент w_1, w_2, w_3, w_4, характеризующий качество связи между двумя нейронами.

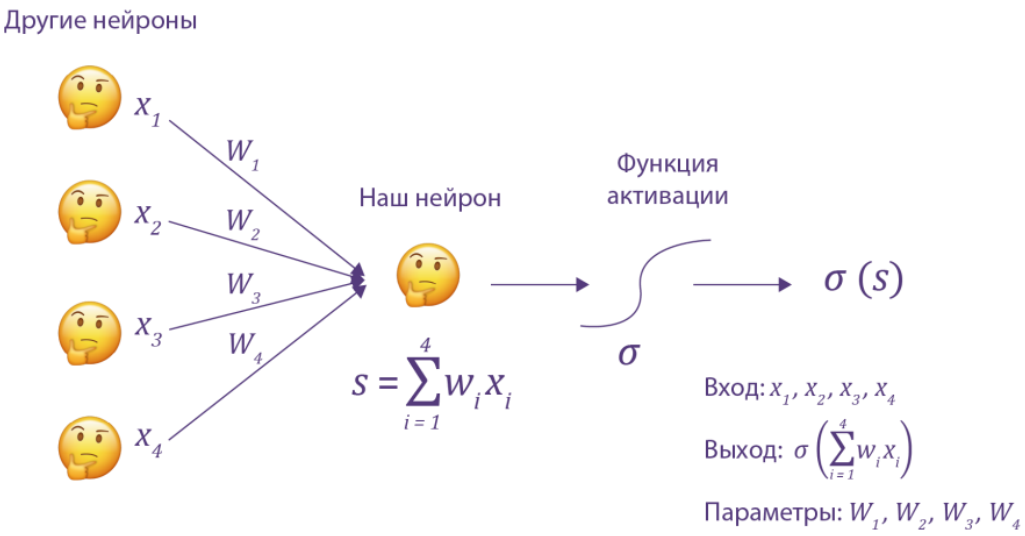


Рисунок 1: Пример простейшей нейросети

Рассматриваемый нейрон собирает все взвешенные сигналы ($w_i x_i$) и использует их в качестве входа для функции активации (примеры приведен в Таблице 1).

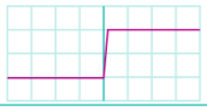
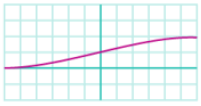
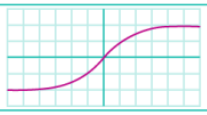
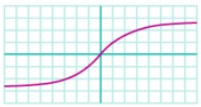
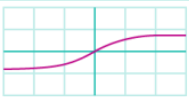
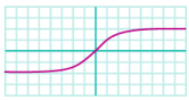
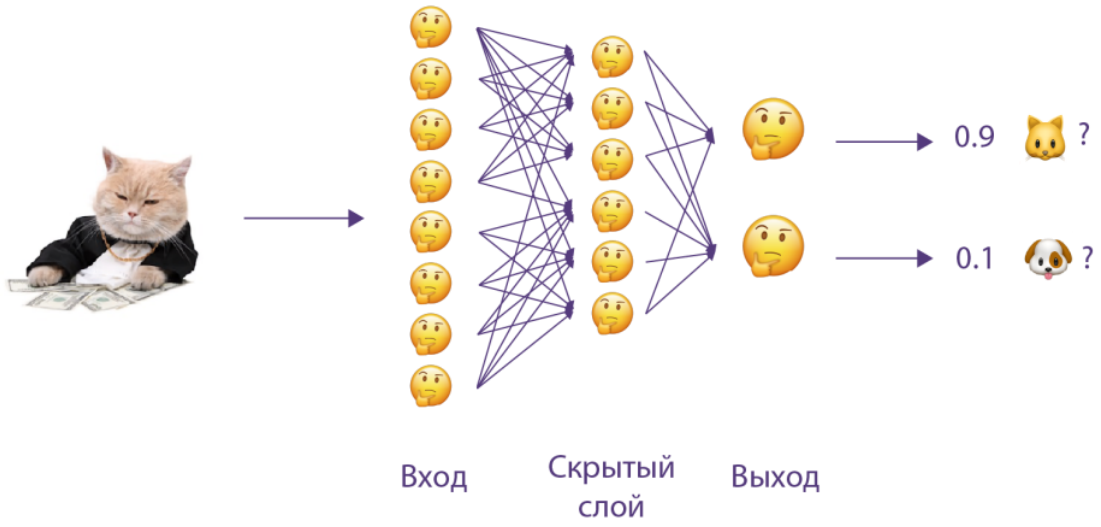
Единичная ступенька		$f(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$
Логистическая (сигмоида или Гладкая ступенька)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$
th		$f(x) = \mathrm{th}(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$
arctg		$f(x) = \mathrm{tg}^{-1}(x)$
Softsign		$f(x) = \frac{x}{1 + \lvert x \rvert}$
Обратный квадратный корень (англ. Inverse square root unit, ISRU)		$f(x) = \frac{x}{\sqrt{1 + ax^2}}$
Линейный выпрямитель (или Полулинейный элемент) (англ. Rectified linear unit, ReLU)		$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}$

Таблица 1: примеры функций активации (данные из Wikipedia).

Выход функции активации и будет выходом нашего нейрона. Можно перейти от рассмотрения работы одного нейрона к набору нейронов и получить простейшуюполносвязнуюнейросеть, состоящую из одного **скрытого слоя**, как на Рисунке 2 (**скрытым** называется слой, который содержит ненаблюдаемые обрабатывающие нейроны; так как нейроны на первом слое, принимающие исходные данные, и нейроны на третьем слое, выдающие результат, являются наблюдаемыми, то на рисунке ниже получаем всего 1 скрытый слой).

На Рисунке 2 представлена задача двухклассовой классификации картинок котов и собак. Мы подаем на вход нейросети изображение, в нашем случае кота, как числовой вектор размерности 8 (размер входного слоя). На выходе получаем вектор вероятности с компонентами 0.9 (вероятность, что на картинке кот) и 0.1 (не кот).



Основные понятия, которые надо знать при работе с нейросетями:

Вход нейронной сети

Данные / сигнал, которые нейросеть использует в качестве входного. Чаще всего на вход нейронной сети подают данные из обучающей выборки и хотят, чтобы выход нейронной сети был как можно ближе к меткам (целевым переменным, или, как их еще называют, таргетам) соответствующих данных. Входными данными может быть картинка, аудиосигнал или вектор признаков.

Нейрон

Примитивный элемент нейросети, задача которого состоит в получении входных сигналов **x**, суммированиих с весами **w** и пропуске через функцию активации.

Веса нейронной сети

Параметры нейронной сети, определяющие её работу.

Архитектура нейронной сети

Способ соединения нейронов в слои и слоев между собой. Обычно, чем больше слоев и нейронов, тем более сложные зависимости может выучить нейросеть, однако, тем сложнее её обучать.

Обучение нейронной сети

Подбор весов нейросети таким образом, чтобы минимизировать функцию потерь на обучающей выборке. Это самый тяжелый с точки зрения вычислительных ресурсов и временных затрат этап. Как правило, нейросети обучаются с помощью алгоритмов типа стохастического градиентного спуска. Во время обучения нейросети показывают группы из обучающей выборки, считают значение функции потерь и градиенты функции потерь по весам нейронной сети (с помощью алгоритма обратного распространения ошибки) и обновляют эти веса, чтобы уменьшить значение функции потерь.

Батч (мини-батч)

Набор объектов из обучающей выборки, на которые нейросеть «смотрит» за одну итерацию.

Эпоха

Полный цикл обновлений весов, после которых нейросеть «увидела» все данные обучающей выборки один раз.

Типы нейронных сетей и решаемые задачи

Сейчаснейронные сети используются практически везде. В Таблице 2 мы привели самые распространенные задачи в области компьютерного зрения (CV) и обработки естественного языка (NLP). Выше мы рассмотрели самые основы построения нейросетей (если точнее, мы познакомились с

полносвязнойархитектурой на Рисунке 2), и, конечно, архитектуры современных моделей продвинулись гораздо дальше, чем простое чередование слоев нейронов. Далее в этом курсе мы подробнее рассмотрим полносвязные сети (Рисунок 2), а также познакомимся с:

сверточными архитектурами (CNN, Рисунок 3)

реккурентными архитектурами (RNN, Рисунок 4)

генеративные нейронными сетями (GAN, Рисунок 5)

трансформерами (Transformers, Рисунок 6)

Тип данных / решаемая задача	Модели
Классификация изображений	LeNet, AlexNet, ResNet, VGG, GoogleNet, ViT
Генерация изображений	StyleGAN, Diffusionmodels
Моделирование естественного языка	GPT-3, Megatron-LM
Распознавание речи	Conformer, BERT, LSTM

Таблица 2. Примеры нейросетевыхмоделей

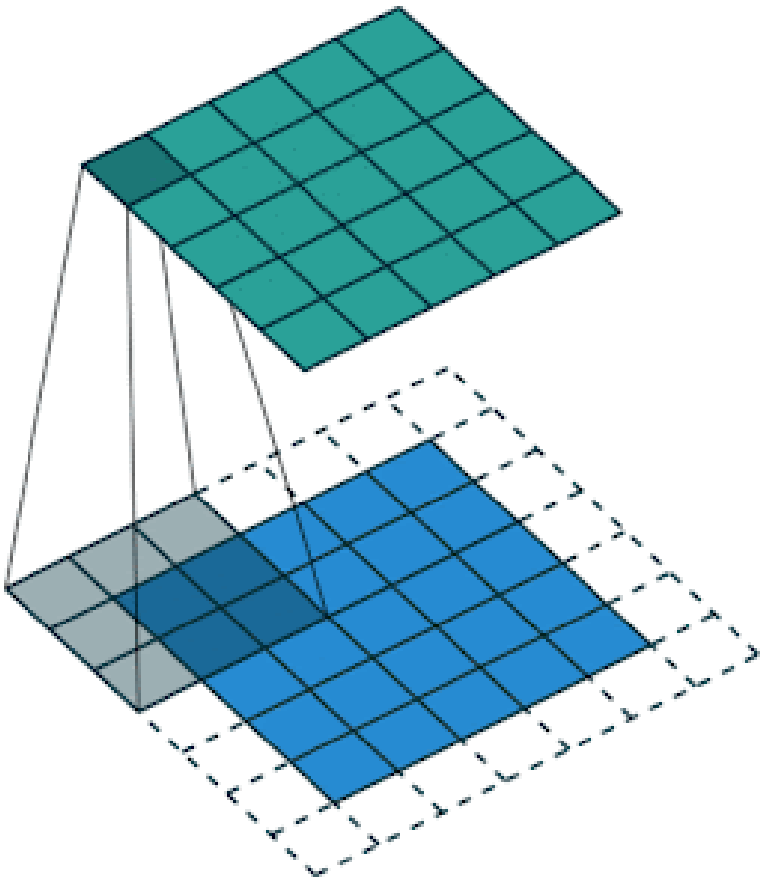


Рисунок 3: пример работы блока свёрточной нейронной сети. Ключевыемодели: AlexNet, LeNet, GoogleNet, ResNet.

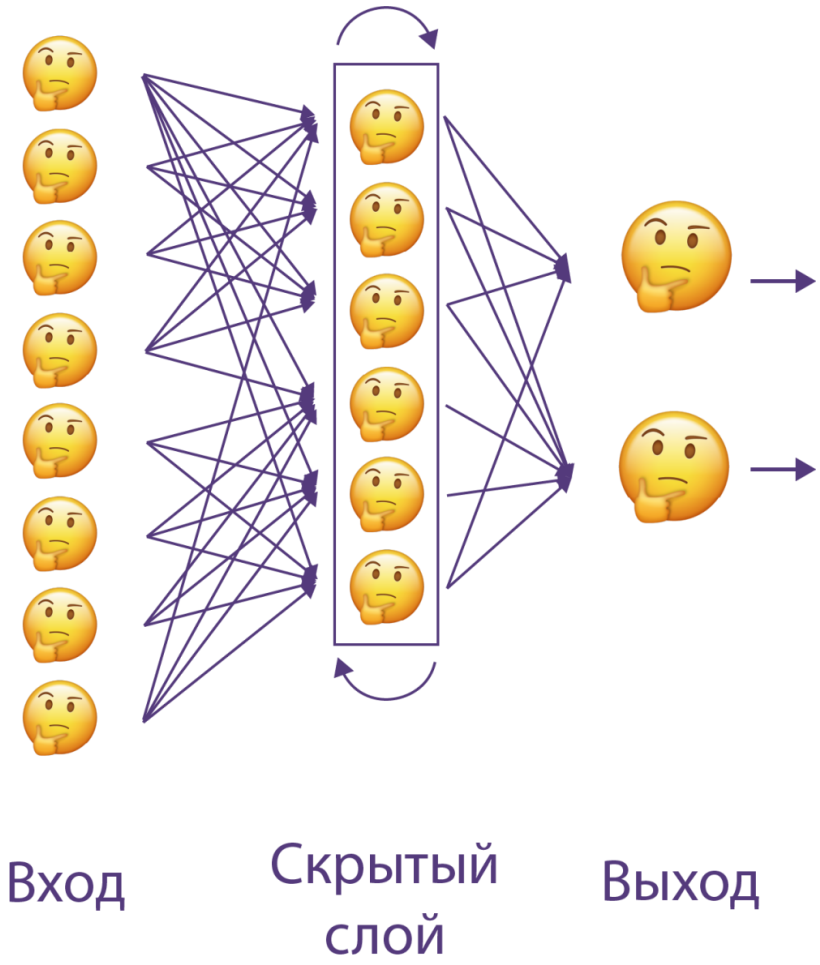


Рисунок 4: пример работы рекуррентной нейронной сети. Ключевые слова: LSTM-блок, GRU-блок.

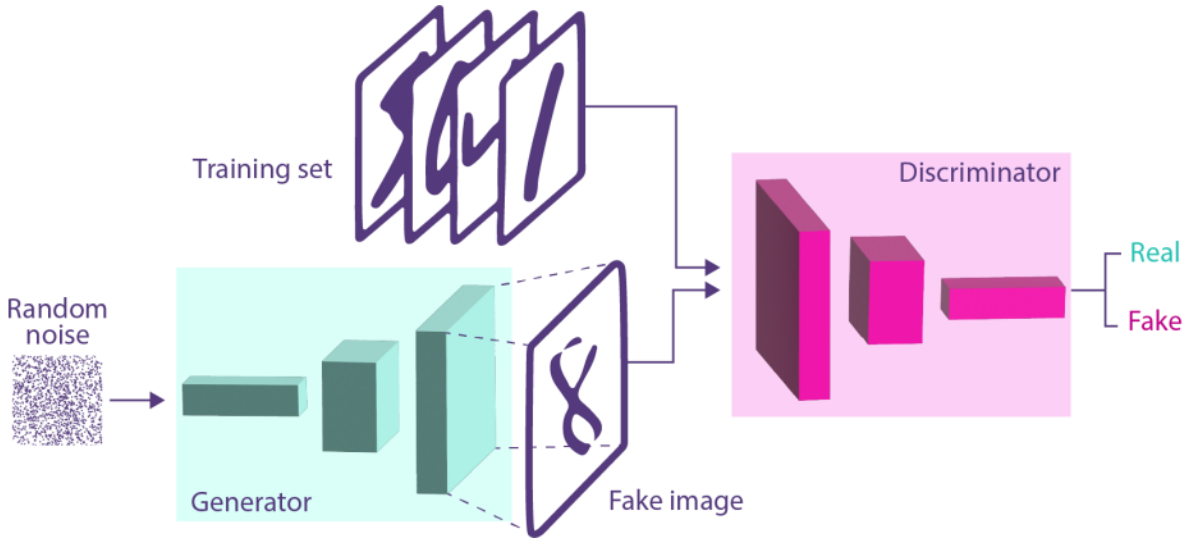


Рисунок 5: пример генеративной нейронной сети. Ключевыеслова: StyleGAN, CyclicGAN, DALL-E, Midjourney.

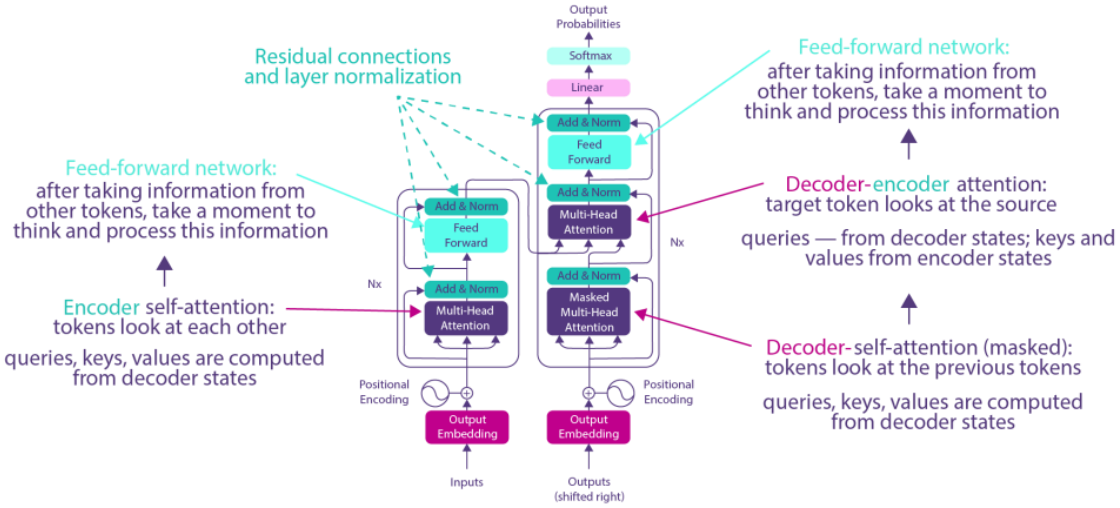


Рисунок 6: архитектуры трансформерного типа. Ключевые слова: GPT (GPT, GPT-2, GPT-3), OPT, YaLM, T5

На чем обучают нейронные сети

С развитием вычислительных мощностей значительно увеличилось количество самих слоев и, соответственно, нейронов. Так, нейросеть GPT-3 из области NLP состоит из 6 слоев с 175 миллиардами параметров. Для обучения нейронныхсетей обычно используют GPU, потому что они состоят из большого количества вычислительных ядер, что позволяет использовать параллелизм вычислений. Однако для обучения State-of-the-art архитектур одной-двух видеокарт не хватит, и вам может понадобится целый кластер.

Собственные кластеры

- NVIDIA DGX

Арендные мощности

- Sbercloud
- Amazon AWS
- Vast.ai
- Googlecloud

Полносвязные архитектуры для решения задач регрессии и классификации

Многослойная полносвязная сеть

В прошлый раз мы познакомились с моделью нейрона и провели ликбез по нейросетям. Сейчас мы более детально рассмотрим многослойную полносвязную модель и разберемся, почему же было создано такое разнообразие нейросетевых архитектур.

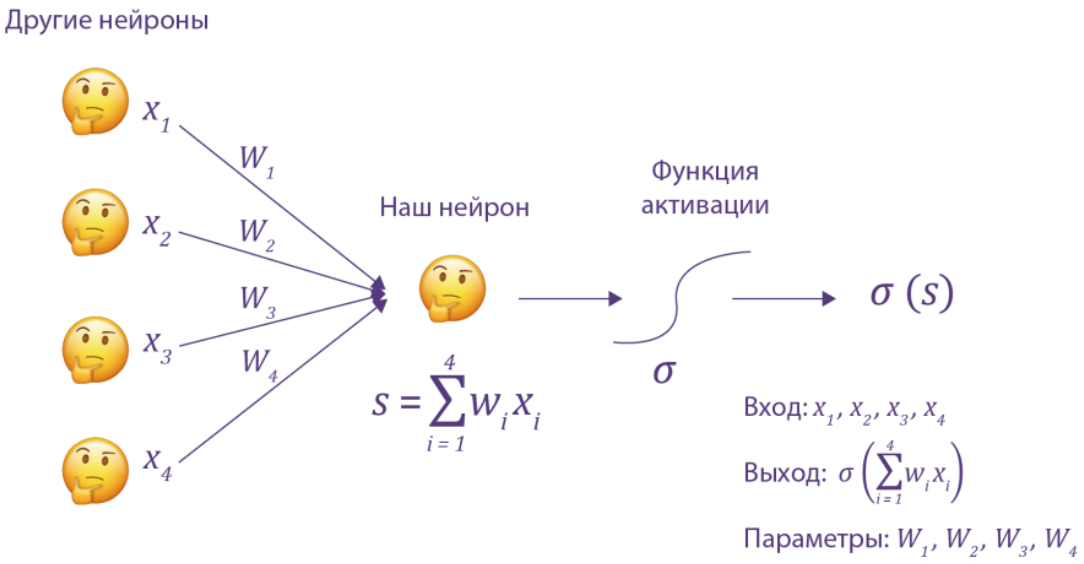


Рисунок 1. Принцип работы нейрона

Внимательные читатели могли заметить, что приведенная в пример модель нейрона (Рис. 1) соответствует *логистической регрессии*. То есть после применения функции активации, а именно сигмоиды, получится вероятность принадлежности к классу 1 в задаче бинарной классификации:

- Пусть дан датасет**D**:

$$D = \{X \times Y\}^n = \{(x^i, y^i) | x^i \in X, y^i \in Y = \{0, 1\}\}_{i=1}^n$$

Здесь $x^i = (x_1^i, x_2^i, x_3^i, x_4^i)$ — четырехмерный вектор признаков

- Тогда:

$$p(y = 1 | \omega, x) = \frac{1}{1 + e^{-w \cdot x}} = \sigma(\omega^T x) = \sigma(\sum_{i=1}^{4} \omega_i x_i)$$

Для большинства современных ML-задач обычной логистической регрессии недостаточно, например в задаче классификации картинок,

исходные признаки в которых малоинформативны. Поэтому для нашего примера мы возьмем суперпозицию логистических регрессий (Рис. 2):

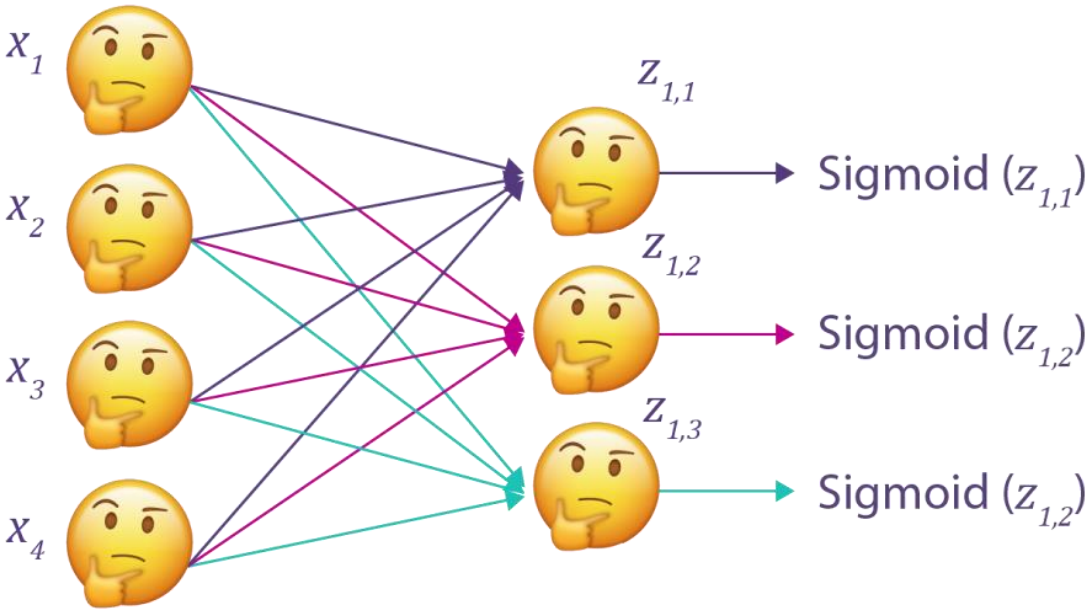


Рисунок 2. Переходим от нейрона к многослойной нейросети (здесь пока неполная сеть)

● Теперь мы используем 3 логистические регрессии

Каждый из нейронов на втором слое собирает входной сигнал (у каждого нейрона свой собственный вектор весов!) и отправляет в функцию активации. Теперь мы интерпретируем результаты каждой из регрессий как новые входные признаки для следующего слоя (Рис. 3):

$z_{i,j}$ — сигнал j-гонейрона на i-м слое

$$x_1^{new} = Sigmoid(z_{1,1}) = \sigma(\sum_{k=1}^{4} \omega_{1,k} x_k)$$

$$x_2^{new} = Sigmoid(z_{1,2}) = \sigma(\sum_{k=1}^{4} \omega_{2,k} x_k)$$

$$x_3^{new} = Sigmoid(z_{1,3}) = \sigma(\sum_{k=1}^{4} \omega_{3,k} x_k)$$

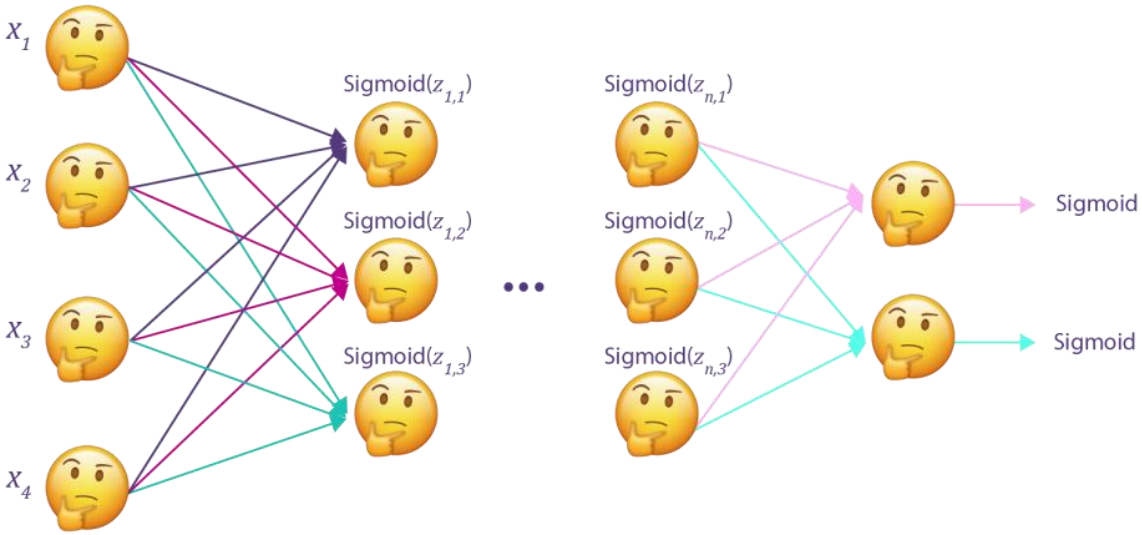


Рисунок 3. Полносвязнаянейронная сеть

Полносвязность заключается в том, что нейроны скрытых и выходного слоев принимают на вход сигналы от всех нейронов на предыдущем слое.

Вычислительная сложность

Рассмотренная выше полносвязнаянейронная сеть является *универсальным аппроксиматором*. Это значит, что мы можем решить конкретную задачу с заданной точностью ε (т. е. можем приблизить любую непрерывную функцию с точностью ε) при наличии в сети достаточного числа слоев N и числа нейронов □1, □2, ..., □□ на этих слоях (вспомните определение предела функции по Коши).

Давайте оценим количество параметров полносвязной нейросети для классификации изображений, например на MNIST:

- 28 × 28 — размер изображения; получаем 784 входных признака
- Пусть в сети 3 скрытых слоя
- Чтобы избежать потери информации, на каждом из слоев создадим 784 нейрона
- Количество нейронов в выходном слое 10, как и количество классов в датасете
- Итого:(28 × 28)1+3 × 10 ≈ 4 × 1012 параметров

Для сравнения: GPT-3, недавний лидер среди нейросетевыхмоделей в области NLP (и при этом самый многовесный), имеет 175 млрд параметров,

что в 20 раз меньше, чем у нашей модели. С другой стороны, сверточные архитектуры позволяют обойтись одним слоем низкой размерности (≈ 20 млн, чтобы получить accuracy более 99 %. Такую модель вы обучите на практике в этом модуле.

Демонстрация

https://colab.research.google.com/drive/1FBIgCMxA4S1PXRk_UZWEcsSyuyDN4e7M?usp=sharing

Оптимизация параметров нейронных сетей: алгоритм обратного распространения ошибки и метод стохастического градиентного спуска SGD

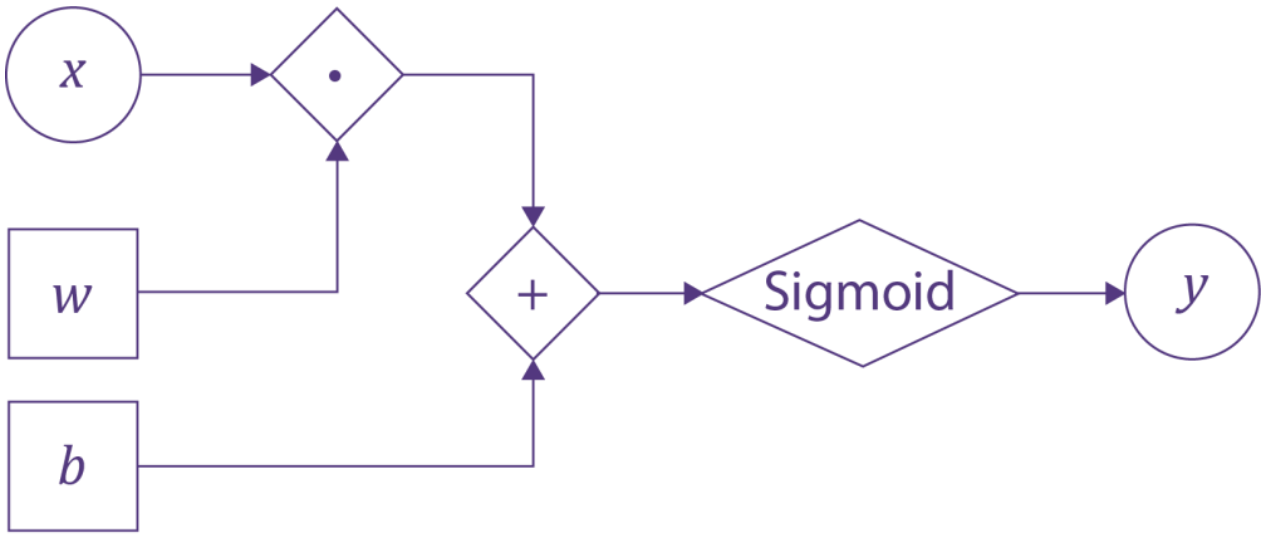
Алгоритм обратного распространения ошибки

В прошлый раз мы познакомились с тем, как работает один нейрон: как данные каждого входа умножаются на веса, суммируются, затем применяется функция активации. Так работает нейрон в режиме *Forwardpropagation*.

Есть совсем другая задача — обучить нейрон. Обучение заключается в том, чтобы найти наиболее подходящие для задачи веса. Обучение построено на простой идее, что если мы на выходе нейрона знаем предсказание нейрона и правильное выходное значение, то нам становится известна ошибка. Эту ошибку можно отправить обратно ко всем входам нейрона и понять, какой вход насколько сильно повлиял на эту ошибку, и, соответственно, подкорректировать вес на этом входе так, чтобы ошибку уменьшить.

Это основная идея алгоритма обратного распространения ошибки — *Backpropagation*. Этот процесс можно прогнать по всей сети и для каждого нейронанайти, как его веса можно модифицировать. Для этого нужно взять производные, но в последнее время делать вручную это не требуется. Все пакеты для работы с нейросетями автоматически дифференцируют. Однако, конечно же, принцип работы Backpropagation нужно знать, так что давайте рассмотрим его на следующем примере:

Задача. Дан граф вычислений:



Нужно найти производную **y** по параметру **b**.

Решение.

1. Запишем математическое выражение, которое задает этот граф:

$$y = \sigma(w \cdot x + b)$$

2. По правилу дифференцирования сложной функции рассчитаем производные по параметрам: $\frac{dy}{db} = \frac{dy}{d\sigma} \cdot \frac{d\sigma}{db}$

3. Получаем ответ: $\frac{dy}{db} = 1 \cdot \frac{d(\frac{1}{1+e^{-w \cdot x - b}})}{db} = \frac{e^{-w \cdot x - b}}{(1+e^{-w \cdot x - b})^2}$

Метод стохастического градиентного спуска SGD

Постановка задачи

Рассмотрим стандартную задачу оптимизации: $\min_w \sum_{i=1}^{n} L_i(w)$

- Целевая функция — сумма конечного числа функций потерь
- Число образцов в обучающей выборке n может быть очень большим

Особенности задачи оптимизации для нейронныхсетей

В силу сложности вычислений полной суммы функций в нейросетевой оптимизации руководствуются следующими принципами:

- Точное вычисление градиента занимает очень много времени
- Высокая точность решения обычно не требуется
- Допустимо введение случайности

Во многих случаях суммируемые функции имеют простую форму, что позволяет осуществить малозатратные вычисления для суммы функций и градиента суммы. Однако в случае нейронныхсетей вычисление градиента суммы может потребовать дорогих расчетов градиентов для всех суммируемых функций. На большом тренировочном множестве в отсутствие простых формул вычисление сумм градиентов становится очень дорогим, поскольку вычисление градиента суммы требует вычисления градиентов отдельных членов суммы. Поэтому для уменьшения объема вычислений используют **стохастический градиентный спуск SGD**. Для уменьшения объема вычислений стохастический градиентный спуск выбирает случайный индекс i на каждой итерации алгоритма и считает градиент только по L_i (w).

Для начала давайте вспомним реализацию обычного градиентного спуска на примере задачи классификации:

1. У нас есть исходное множество данных для обучения **D** и веса **w** нашего алгоритма: $D = \{X \times Y\}^n = \{(x_i, y_i) | x_i \in X, y_i \in Y\}_{i=1}^n$

2. Мы выбрали функцию потерь, для i-го объекта выборки обозначим ее как L_i (w)

3. Выберем начальную инициализацию весов $w^{(0)}$. Тогда шаг градиентного спуска будет вычисляться как $w^{(t+1)} = w^{(t)} - \lambda \sum_{i=1}^{n} \nabla L_i(w^{(t)})$, где λ определяет скорость сходимости

Таким образом, на каждой итерации градиентного спуска совершается проход по всей обучающей выборке. Такая реализация называется *пакетной (англ. batch)*. По сути, под пакетом понимается batch, и создается впечатление, что в градиентном спуске где-то должен быть batch. Но реально под пакетном в градиентном спуске понимается взятие градиента по всей обучающей выборке.

SGD

Приведем шаг стохастического градиентного спуска: $w^{(t+1)} = w^{(t)} - \lambda \nabla L_i \ (w^{(t)})$.

Здесь i — случайновыбранный индекс. Как видим, теперь направление **w** будет определяться за O(1) вместо O(n).

Mini-batch

На практике применяют подход*mini-batch* — среднее между пакетным градиентным спуском и SGD. На каждой итерации выбирается подмножество суммируемых функций, а не единственный случайный индекс. Датасет разбивается на пакеты — это то, что мы на практике понимаем под batch. Так что не путайте с пакетным градиентным спуском. Шаг оптимизации mini-batch можно записать следующим образом:

$$w^{(t+1)} = w^{(t)} - \lambda \sum_{i \in I} \nabla L_i \ (w^{(t)}).$$

Здесь I — случайно выбранное подмножество индексов. Как видим, теперь направление **w** будет определяться за O(I) вместо O(n).

Задача обучения нейронных сетей как задача обучения представления. Использование готовых представлений: transferlearning

https://colab.research.google.com/drive/17xTfqa9Z0IxmFNvvObuFKDqnOLQE4BYK

Как собирать данные: картинки, тексты, графы, их разметка

Чаще всего решение любой задачи машинного обучения начинается не с самого обучения, а с поиска и подготовки данных — того, на чем предстоит обучать. Даже в крупных компаниях для решения ML-задачи данных может не хватать или не иметься вовсе. ML-специалисту необходимо знать, где можно найти данные и с помощью каких инструментов выгрузить.

Давайте разберем основные способы сбора данных: использование готовых выборок, парсинг данных в вебе и разметка. Кроме того, мы

посмотрим, на что стоит обращать внимание при обогащении данных с помощью сторонних ресурсов.

Готовые наборы данных

Один из самых простых способов собрать данные для машинного обучения — использовать уже собранные кем-то данные.

Нередко бывает так, что с помощью одной и той же выборки можно решить не одну задачу машинного обучения. Понимая ценность собранных данных, многие организации и отдельные исследователи публикуют свои выборки в открытом доступе для свободного использования.

Единственная сложность — знать, где взять эти наборы данных. Посмотрим на несколько источников:

1. Поисковая система по наборам данных

DatasetSearch — специальная версия поиска от Google, которая по запросу пользователя находит подходящие наборы данных. Проиндексировано около 25 000 000 наборов данных с числовыми, текстовыми и файловыми данными.

2. Конкурсные платформы

Kaggle — веб-платформа, реализованная как социальная сеть для специалистов DataScience. Пользователи там организуют соревнования по анализу данных, а также публикуют собранные наборы данных в специальном разделе.

В поиске по наборам данных можно настроить фильтры, в каком формате необходимо получить данные. Среди доступных — JSON, CSV, SQLite, BigQuery. Можно отобрать только те выборки, которые обладают определенной лицензией. На выбор — около 180 000 выборок.

3. Агрегаторы данных

Если на конкурсных платформах не всегда понятно, каким образом были собраны данные и можно ли им доверять, то можно обратиться к платформам, которые собирают данные из первых рук.

Например:

- «Росстат»—официальныйсайт«Росстата»ежемесячно публикует данные о статистике общественных процессов в России. Скачать данные можно в формате XLSX на выбранный период
- Открытые базы данных ВЦИОМ—Всероссийский центр общественного мнения публикует на своем сайте результаты опросов населения по разным темам
- «Яндекс.Вордстат»—агрегаторстатистикипоисковыхзапросов от «Яндекса». Доступны как числовые, так и текстовые данные
- Unsplash—постоянно обновляющийсямеждународный сток фотографий на разные темы и для свободного использования

4. Открытые репозитории

Github — один из самых известных веб-сервисов для хостинга IT-проектов. По поиску открытых проектов на сервисе можно находить похожие исследования и наборы данных для них.

Веб-парсинг данных

Бывает так, что возможности скачать набор данных с веб-сайта за несколько кликов не предусмотрено, а копирование не запрещается, однако это долгий и неэффективный процесс.

Веб-парсинг — способ автоматического сбора данных с веб-сайтов. Рассмотрим несколько таких способов.

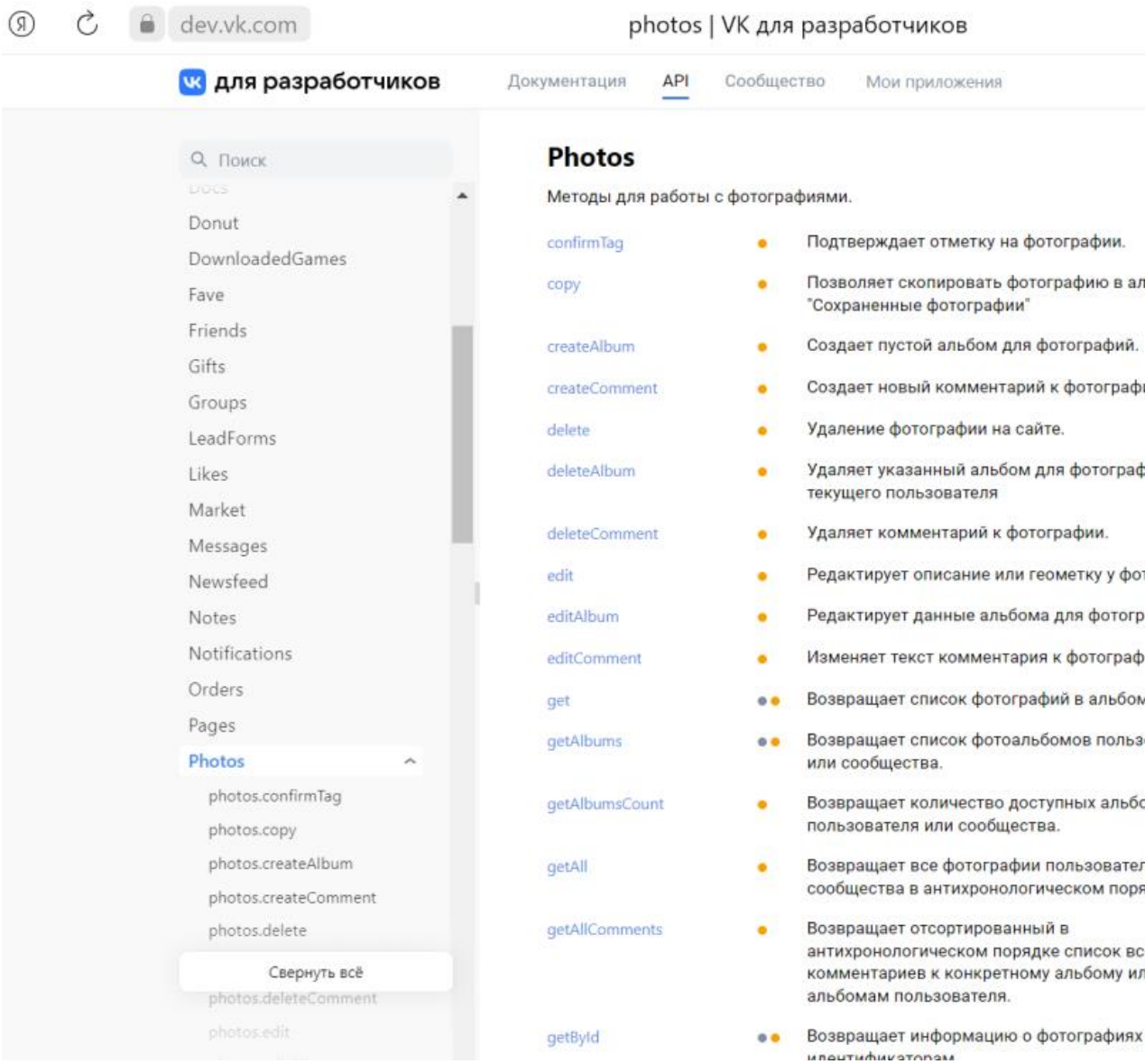
1. С использованием API

Проще всего скачать данные с веб-сайта, если у него есть свой публичный API (Applicationprogramminginterface). Это набор определенных методов взаимодействия с веб-сайтом (набор «ручек», по которым можно ходить и «дергать» необходимые данные).

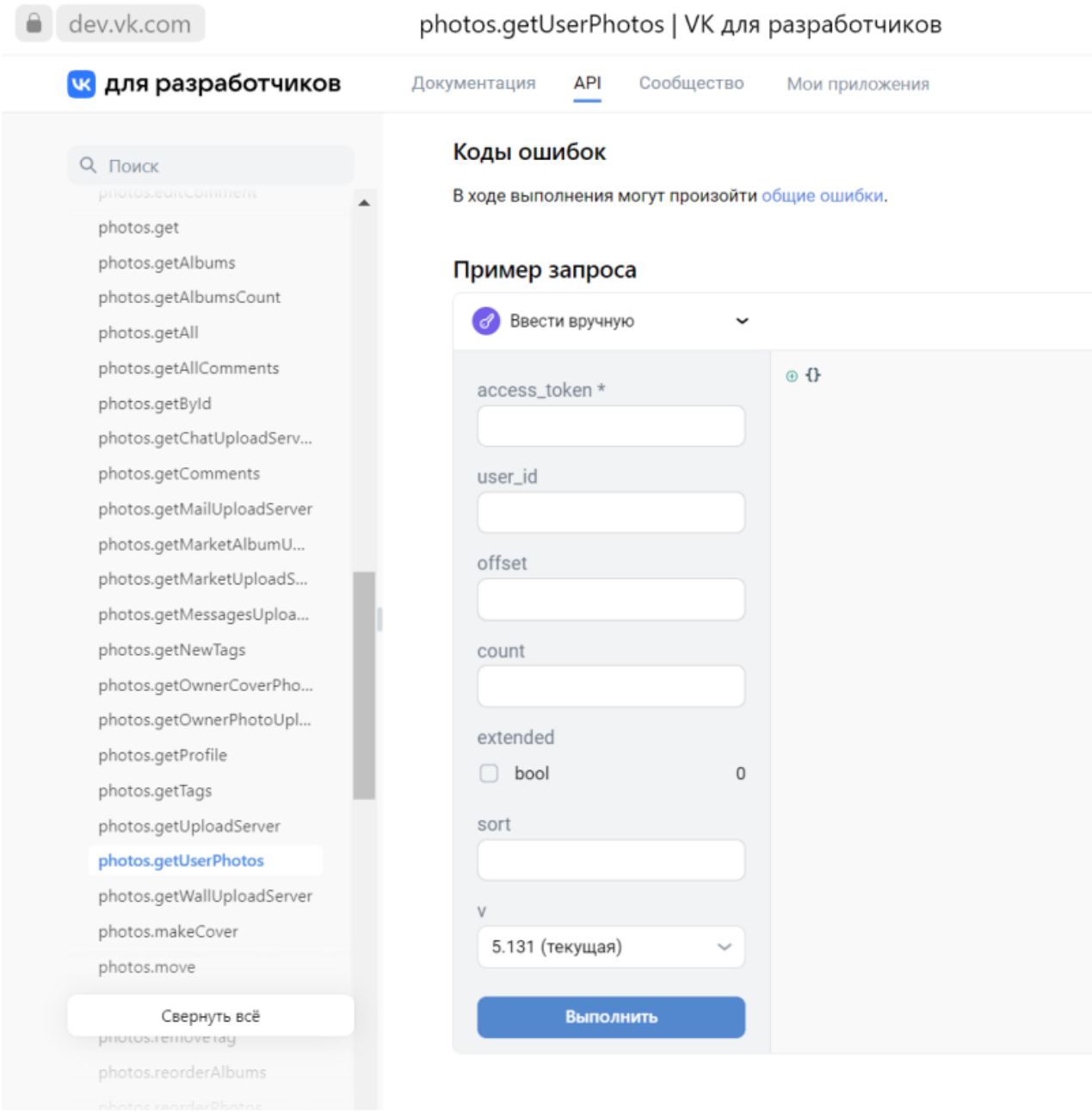
Рассмотрим на примере выгрузки данных из API «ВКонтакте».

Допустим, нам необходимо получить фотографии пользователя. Для этого нужно:

1. Зайти в документацию API «ВКонтакте» для разработчиков 2. В разделе методов API выбрать необходимый метод для выгрузки фотографий, напримерphotos.getUserPhotos



3. Перейти к примеру запроса, ввести необходимые данные: токен доступа, идентификатор пользователя и другие данные. После необходимо нажать «Выполнить запрос». Результат запроса можно скопировать



Как получить ключ доступа, рассматривается в этом разделе документации.

Если необходима регулярная выгрузка информации из «ВКонтакте», можно написать скрипт с помощью, например, библиотеки Requests для Python. Ее использование покажем в следующем примере.

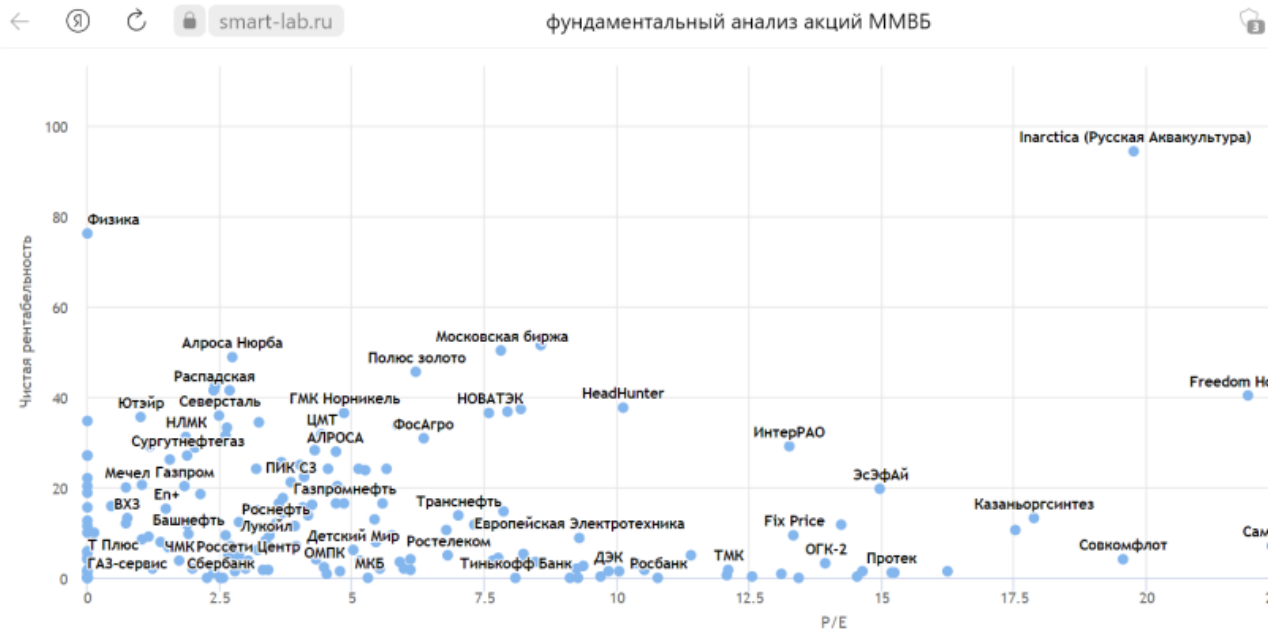
2. Без использования API

Если API у веб-сайта нет, однако политикой сайта не запрещено использование данных, то можно провести **синтаксический анализ HTML**: взять код HTML сайта и извлечь необходимую информацию из множества тегов.

Алгоритм будет выглядеть следующим образом:

1. Послать GET-запрос к веб-сайту
2. Получить HTML-документ необходимой веб-страницы
3. Очистить информацию от HTML-тегов
4. Сохранить данные в необходимом формате

Допустим, нам нужно скачать таблицу с сайта smart-lab.ru. Помимо необходимой таблицы, на сайте есть еще другая ненужная информация, поэтому нам предстоит найти, под какими html-тегами скрывается наша таблица, и достать именно ее.



№	Название	Тикер			Капит-я млрд руб ↓	EV млрд руб	Выручка	Чистая прибыль	ДД ао, %	ДД ап, %	ДД/ЧП, %	P/E	P/S	P/B	EV/EBI
1	Газпром	GAZP			3 844	6 714	10 241	2 093.0	0.0%		0%	1.8	0.4	0.2	1.8
2	Роснефть	ROSN			3 769	9 185	8 761	1 057.0	11.7%		42%	3.6	0.4	0.7	3.9
3	НОВАТЭК	NVTK			3 199	3 333	1 157	421.3	6.8%		51%	7.6	2.8	1.7	4.5
4	Лукойл	LKOH			2 620	2 700	9 431	773.4	21.8%		74%	3.4	0.3	0.6	1.9
5	Газпромнефть	SIBN			2 441	2 601	3 068	503.4	10.9%		53%	4.8	0.8	1.1	2.9

Чтобы написать скрипт данных на Python, который позволяет вытащить такую таблицу с веб-сайта, можно импортировать следующие библиотеки:

```
1 import requests
2 from bs4 import BeautifulSoup
3 import pandas as pd
```

- Requests—библиотека для отправки get-запроса к веб-сайту
- Bs4—библиотека для синтаксического разбора html
- Pandas—библиотека для сохранения данных в датафрейм для последующей обработки

Для начала с помощью Requests отправляем get-запрос к веб-сайту и в качестве ответа получаем строку, что является HTML запрашиваемой страницы.

```
1 link = "https://smart-lab.ru/q/shares_fundamental/"
2 response = requests.get(link)
3 response.text
```

```
'<!DOCTYPE html>\n<html lang="ru">\n<head>\n\n\t\n\t<!-- Global Site Tag (gtag.js) - Google Analytics -->\n\t<
etagmanager.com/gtag/js?id=UA-16537214-3"></script>\n\t<script>\n\twindow.dataLayer = window.dataLayer || [];\
rguments);}\n\tgtag(\'js\', new Date());\n\tgtag(\'config\', \'UA-16537214-3\', {\n\t\t\t\'custom_map\': {\n\t
tred\',\n\t\t\t\t\'dimension2\' : \'content_owner\'\n\n\t\t\t},\n\n\t\t\t\'user_registred\': \'No\',\n\t\t\t\'
cript>\n\t\n\n\t<meta name="push-subscribes" content="no">\n\t<meta http-equiv="content-type" content="text/ht
a property="vk:image" content="https://smart-lab.ru/templates/skin/smart-lab-x3/images/smart-logo-vk.png">\n\t
акций ММВБ</title>\n\t<meta name="DESCRIPTION" content="Фундаментальный анализ акций компаний, обращающихся на
```

Найти необходимую таблицу в такой строке достаточно сложно, поэтому используем специальную библиотеку для синтаксического разбора HTML.

```
1 soup = BeautifulSoup(response.text,'lxml')
2 soup
```

```
<!DOCTYPE html>
<html lang="ru">
<head>
<!-- Global Site Tag (gtag.js) - Google Analytics -->
<script async="" src="https://www.googletagmanager.com/gtag/js?id=UA-16537214-3"></script>
<script>
        window.dataLayer = window.dataLayer || [];
        function gtag(){dataLayer.push(arguments);}
        gtag('js', new Date());
        gtag('config', 'UA-16537214-3', {
                        'custom_map': {
                                'dimension1' : 'user_registred',
                                'dimension2' : 'content_owner'

                        },
```

Находим, что наша таблица заключается в теге table, и с помощью bs4 забираем информацию только по таблице. Далее используем pandas и специальный метод read_html, который позволяет сохранить найденную таблицу в датафрейме для последующей обработки данных.

```
1 tables = soup.find_all('table')
2 dfs = pd.read_html(str(tables), index_col=0)
3 dfs
```

```
[           Название Тикер  Unnamed: 3  Unnamed: 4 Капит-я млрд руб  \
№
1            Газпром  GAZP         NaN         NaN             3 802
2           Роснефть  ROSN         NaN         NaN             3 518
3            НОВАТЭК  NVTK         NaN         NaN             3 087
4             Лукойл  LKOH         NaN         NaN             2 982
5     ГМК Норникель  GMKN         NaN         NaN             2 212
..              ...   ...         ...         ...               ...
239           ЦИАН  CIAN         NaN         NaN               NaN
240         Физика  NPOF         NaN         NaN               NaN
241         Т Плюс  VTGK         NaN         NaN               NaN
242  Косогорский МЗ  KMTZ         NaN         NaN               NaN
243      Уралкалий  URKA         NaN         NaN               NaN
```

Разметка данных

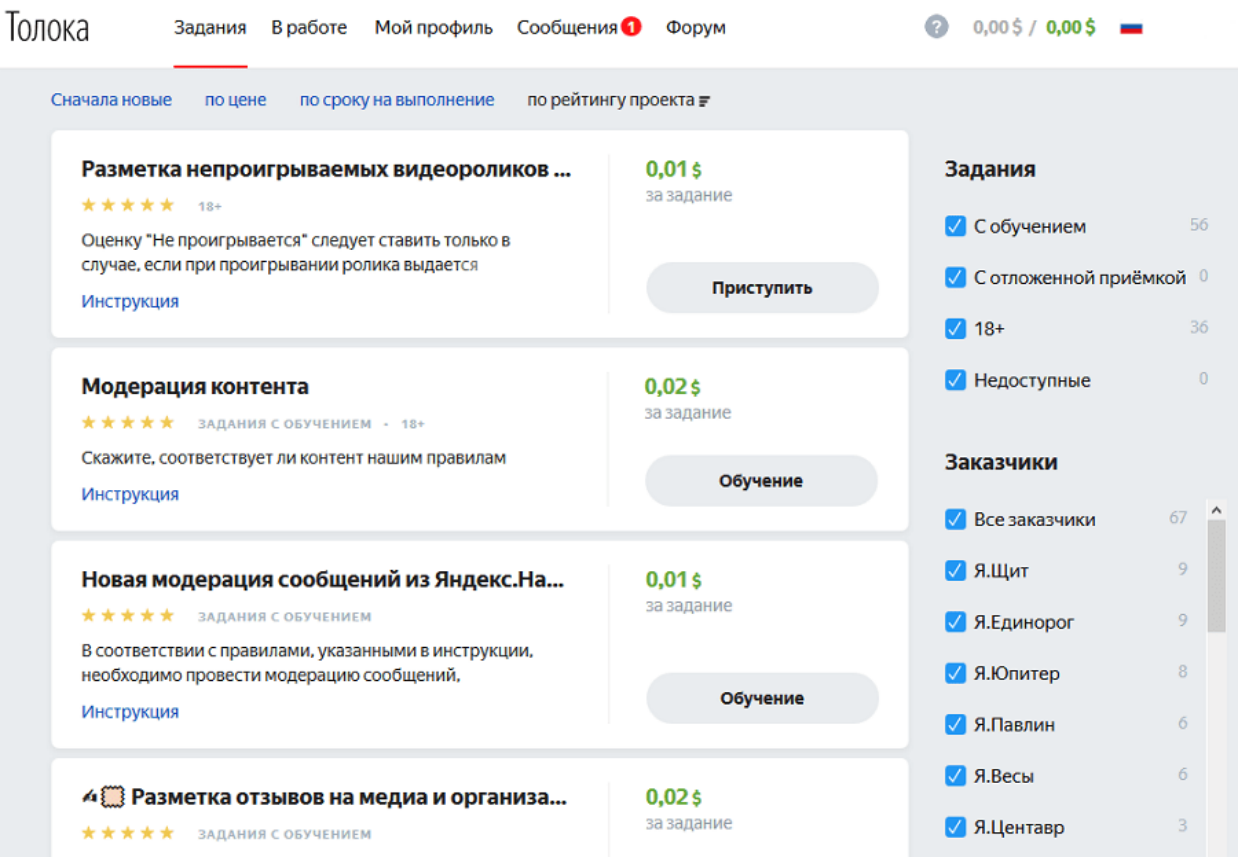
Бывают случаи, когда какие-то данные уже собраны, однако для тренировки их недостаточно, потому что не хватает целевых меток.

Например, нам необходимо обучить модель, которая должна определять, есть ли на фотографиях дорожные знаки. Допустим, фотографии мы нашли, но нет информации о том, есть ли на них дорожные знаки.

Можно разметить эти данные самостоятельно, однако если объем фотографий большой, то это не быстро, а еще можно ошибиться. В таком случае можно воспользоваться краудсорсинговыми платформами.

«Яндекс.Толока» — одна из платформ для разметки данных. На ней зарегистрировано свыше десятка тысяч пользователей, готовых размечать данные за деньги.

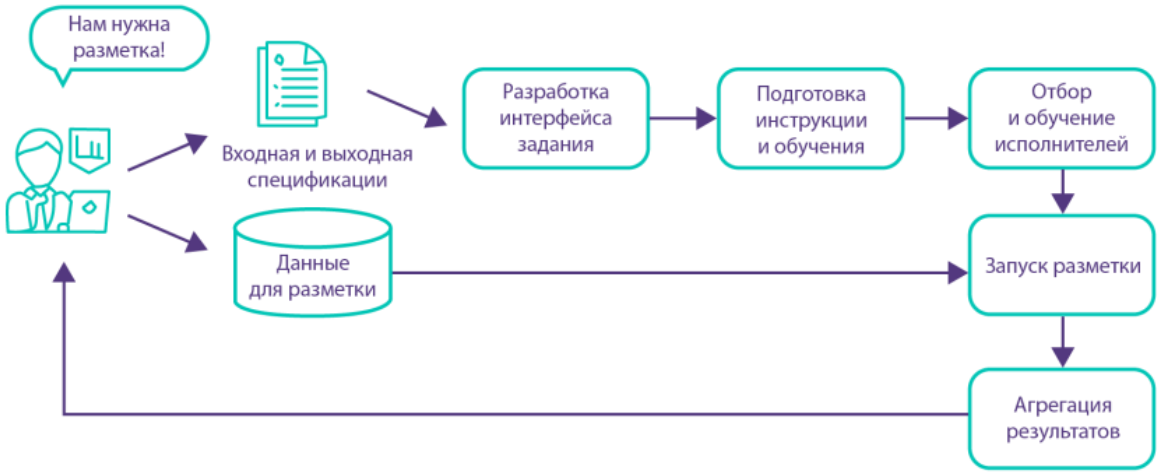
Вот так выглядит интерфейс для разметчиков.



Можно повышать качество разметки, отбирая только самых умелых исполнителей, а также регулировать скорость, делая задание более привлекательным с помощью цены и повышения удобства интерфейса.

Общий процесс подготовки разметки данных состоит из шагов с картинки ниже. Самое главное — понять, в каком формате вы хотите получить разметку, а также объяснить исполнителям, что вы требуете от них. Последнее можно сделать с помощью инструкции или отдельных обучающих заданий

.



Подробнее о настройке разметки в «Толоке» можно почитать в документации платформы.

Особенности сбора данных

При использовании любого способа сбора данных стоит помнить о следующих вещах:

Данные могут быть недостоверными

Стоит всегда проверять, откуда вы скачиваете данные. Если вам необходима статистика, используйте официальные сайты и проверенные источники. Еще один способ проверки — сравнение одних и тех же данных с разных источников.

Данные могут быть неразнообразными

Вы можете скачать фотографии дорог с дорожными знаками, однако там будут встречаться одни и те же дорожные знаки, что может сильно ограничить возможности вашей модели. Данные нужно проверять на разнообразие и в случае необходимости обогащать данные с помощью

других источников. Не бойтесь использовать несколько источников для создания одного набора данных.

Данные могут быть неактуальными

Всегда следует проверять, когда был выложен датасет, и соответствуют ли данные действительности. Бывает так, что со временем в наборах данных в интернете обнаруживаются ошибки; об этом пишут на соответствующих форумах.

Универсального способа сбора данных не существует. Зачастую итоговый набор данных для обучения модели состоит из нескольких источников, и вероятность качества данных в таком наборе данных выше.