Современное состояние NLP. Введение в нейронные сети

Елена Кантонистова

План рассказа

- О курсе
- Современное состояние NLP
- Введение в нейронные сети

О курсе

О курсе

- **Первая неделя**
  - Обзор сегодняшнего состояния NLP. Что такое нейросеть и как она работает
  - Библиотека NumPy
- **Вторая неделя**
  - GD и Backpropagation
  - Работа с текстом в Python
- **Третья неделя**
  - Фреймворк PyTorch
  - Векторные представления слов - 1
- **Четвертая неделя**
  - Скрапинг
  - Векторные представления слов - 2
- **Пятая неделя**
  - Написание телеграм бота
  - Подведение итогов, защита проектов (по желанию)

Проект курса

Обучение:

- Собираем данные с РИА Новостей (за последний год)
- Решаем на них задачу классификации (теги с сайта - ответы)
- Обучаем на них свой w2v и/или берем веса из предобученной модели

Проект курса

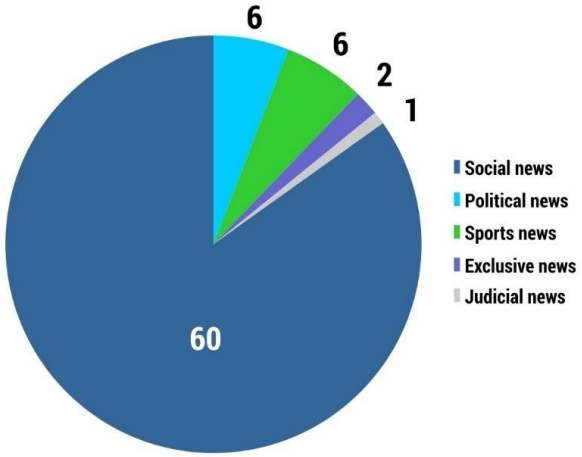
Применение (тг-бот):

Опция-1: пользователь вводит дату, парсим все новости за эту дату и применяем обученный классификатор.

- Как результат - для каждой новости показываем ее тему
- Строим визуализацию распределения новостей по темам

за выбранную дату

Опция-2: пользователь вручную задает список тем ("красивые животные", "российская анимация 90-х"), и для новостей за введенную пользователем дату получаем, какие из этих новостей на сколько % соответствуют введенным пользователем темам



Современное состояние NLP

Задачи обработки языка (NLP)

- Классификация текстов/документов
- Распознавание именных сущностей (NER)
- Машинный перевод
- Вопрос-ответные системы (QA)
- Извлечение информации (IR)
- Суммаризация текстов/документов
- Генерация текста

и множество других

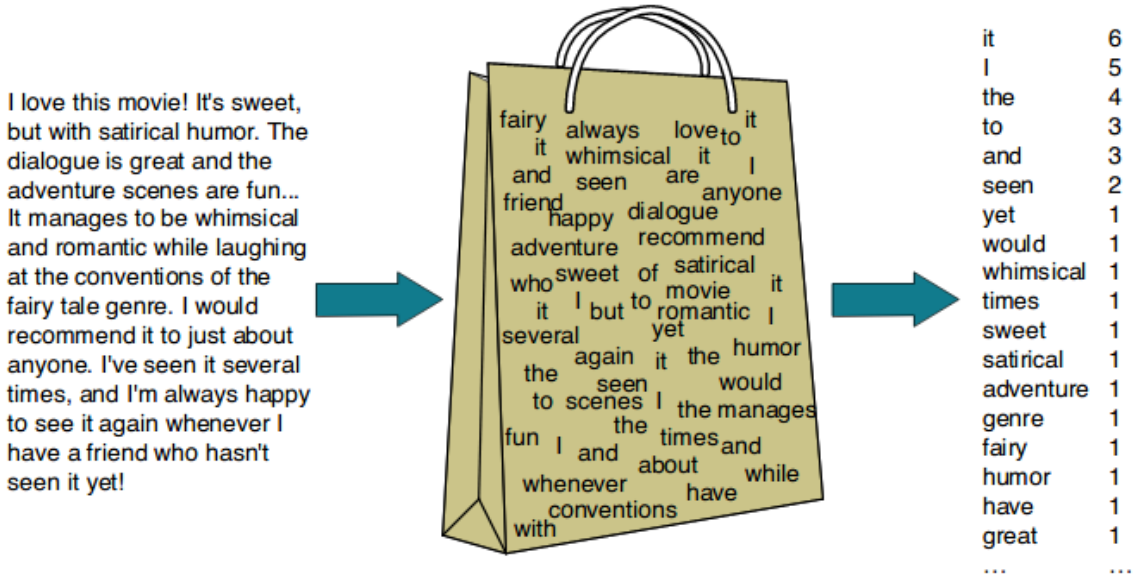
Реальные кейсы применения NLP

- Поисковые системы (Google, Yandex, etc.)
- Машинный перевод (Google Translate, Abbyy Lingvo, Linguee, etc.)
- Виртуальные ассистенты, голосовые помощники etc.
- Фильтр спама (e-mail / телефонный / etc.)
- Дополнение текста, автокоррекция
- Авторазметка отзывов пользователей
- Чат-боты
- Автосуммаризация текста

Подходы к решению NLP-задач

Классическое машинное обучение:

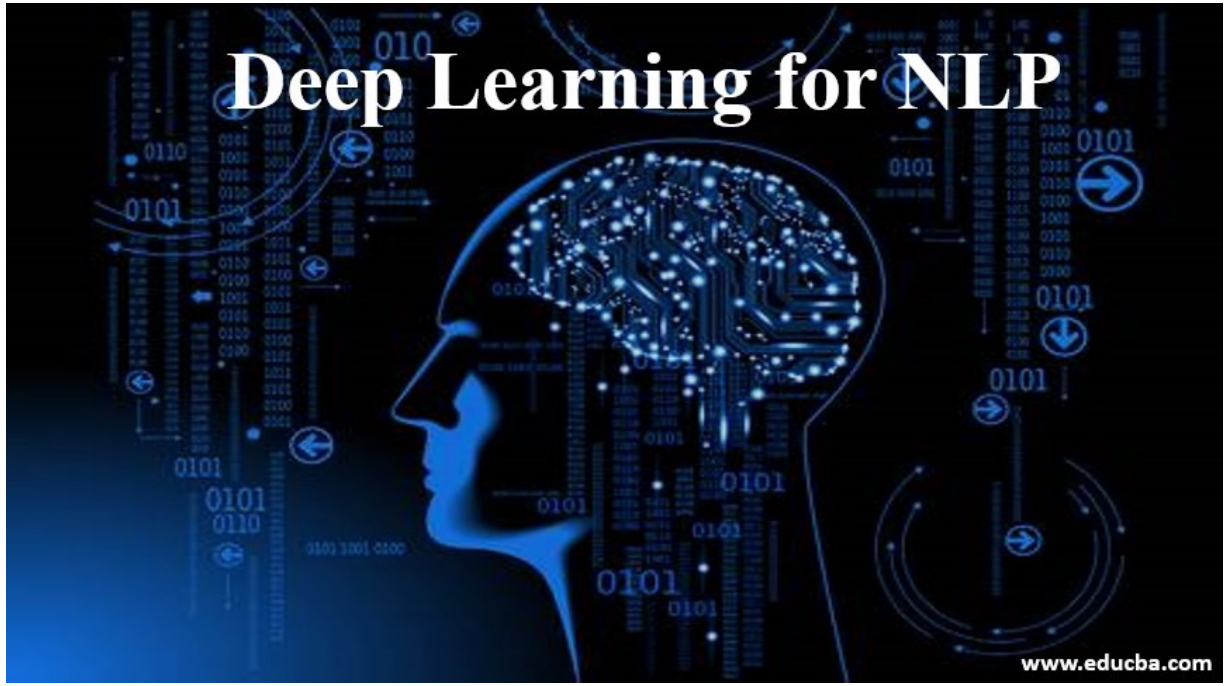
- Извлекаем признаки из текстов (bag of words, tf-idf)
- На этих признаках обучаем ML-модель



Подходы к решению NLP-задач

Глубинное обучение:

- Нейронные сети самостоятельно извлекают необходимую информацию из текстов



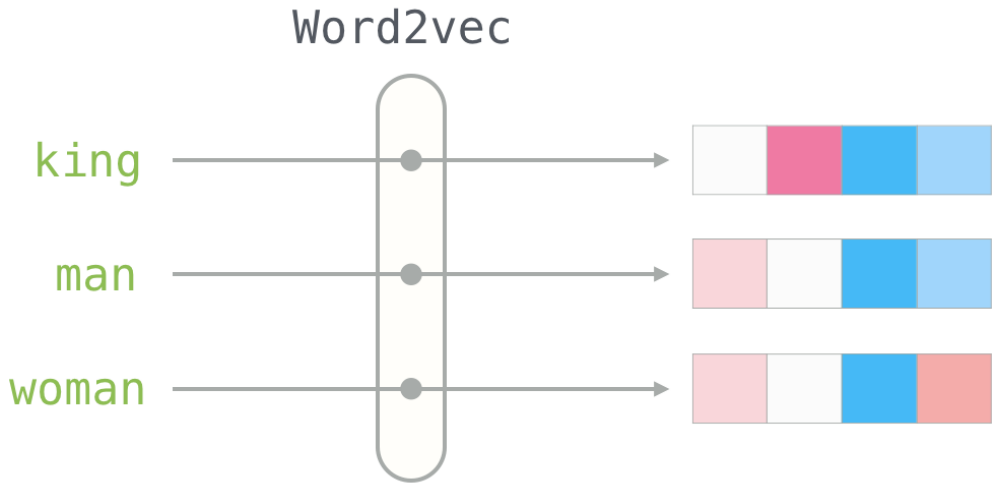
Глубинное обучение в NLP

Виды нейронных сетей:

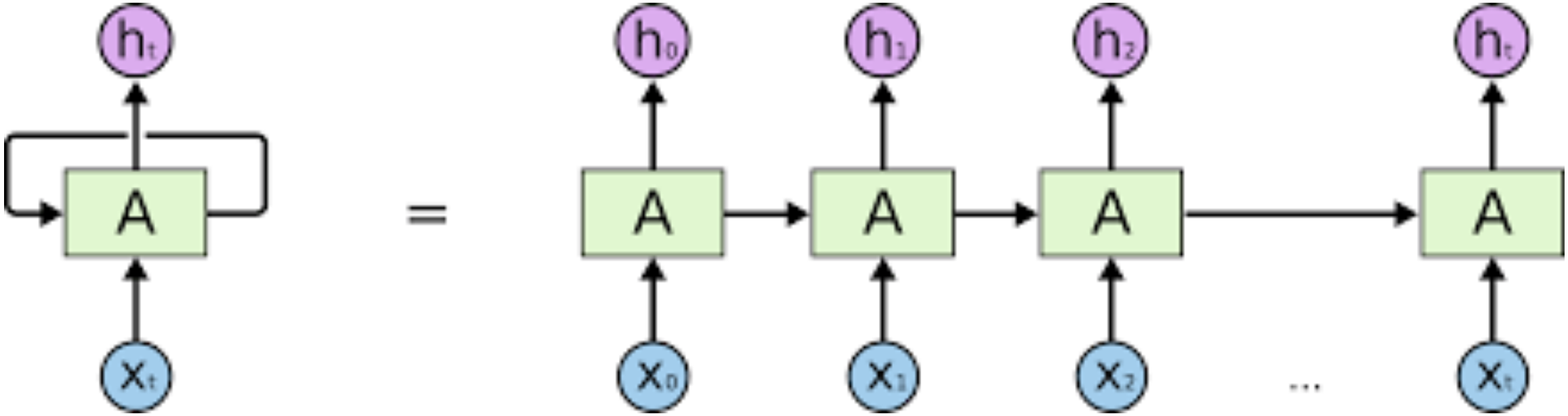
1. Полносвязные нейронные сети (основа основ) – word2vec, fasttext, GloVe

2. Рекуррентные нейронные сети

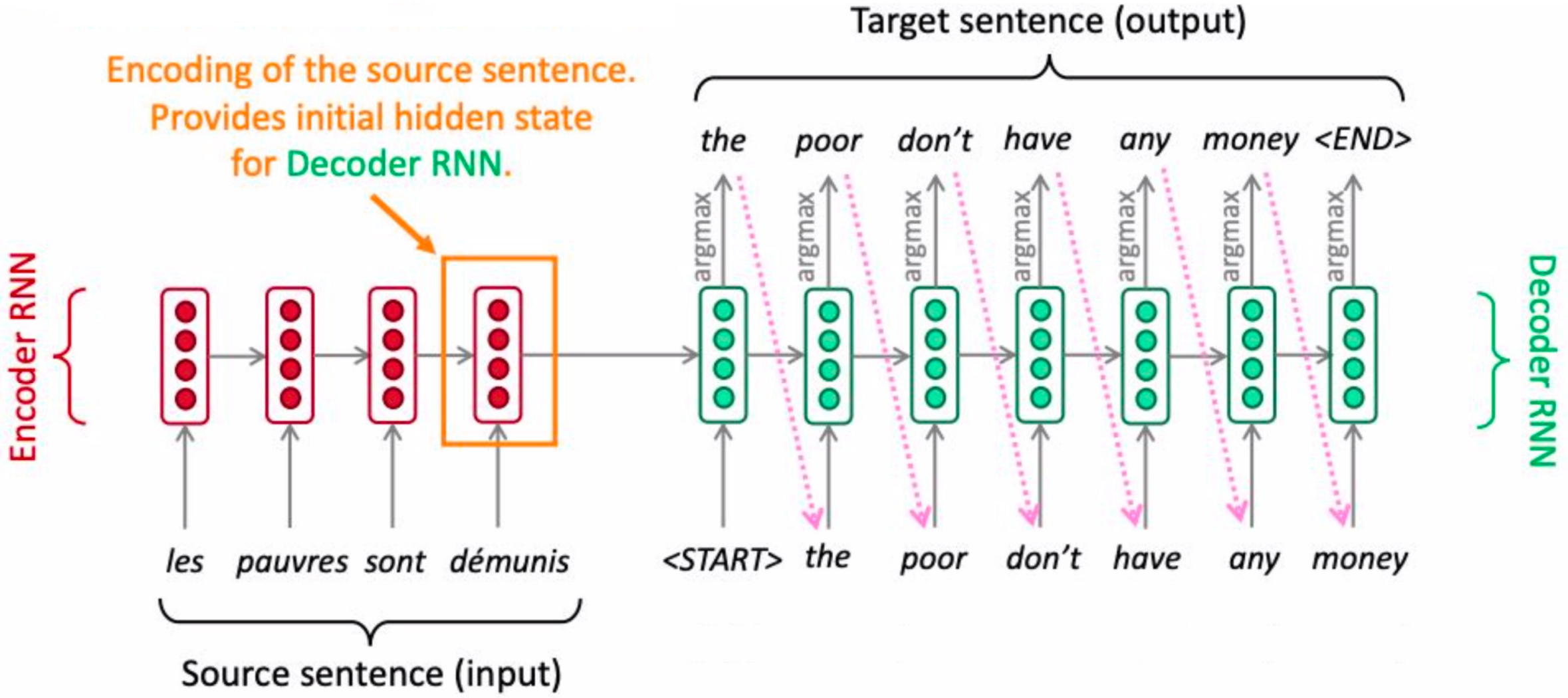
3. Attention и трансформеры



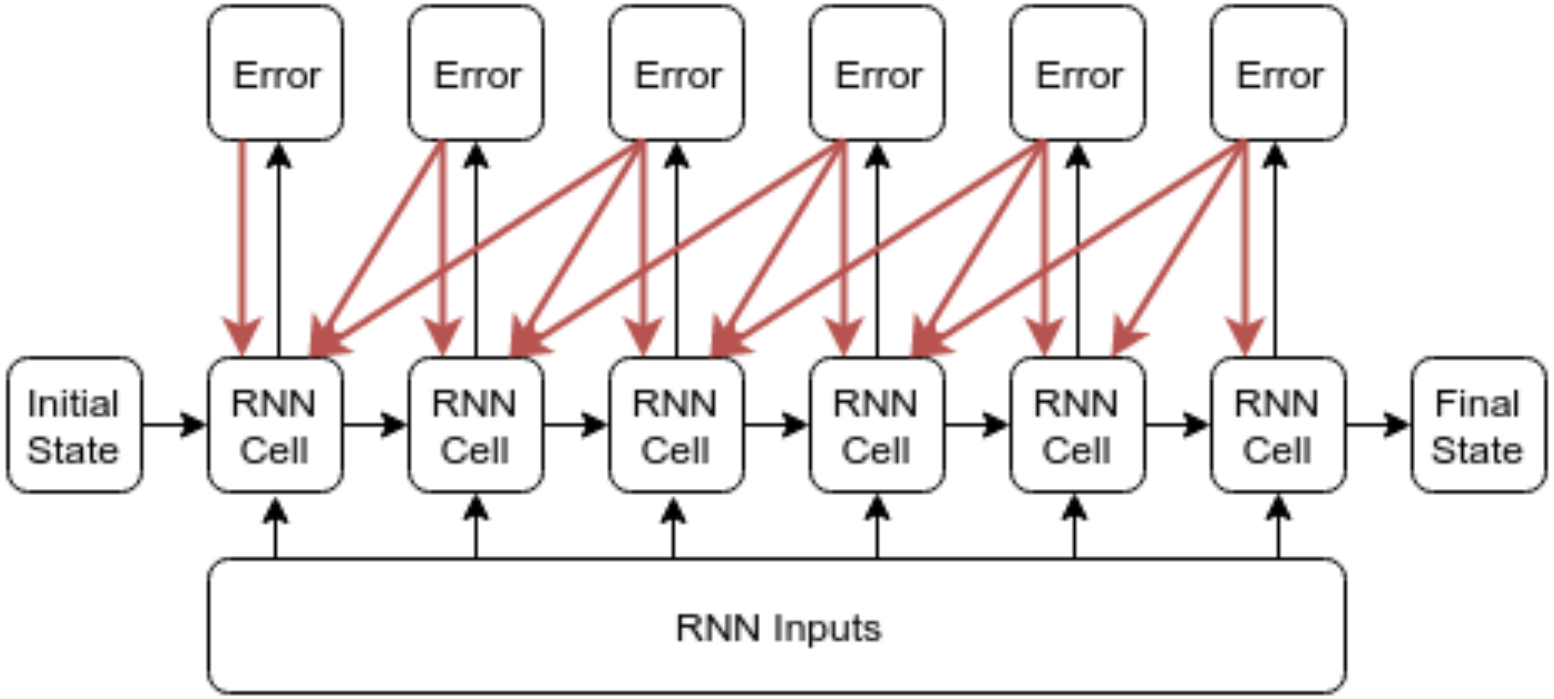
2. Рекуррентная нейронная сеть

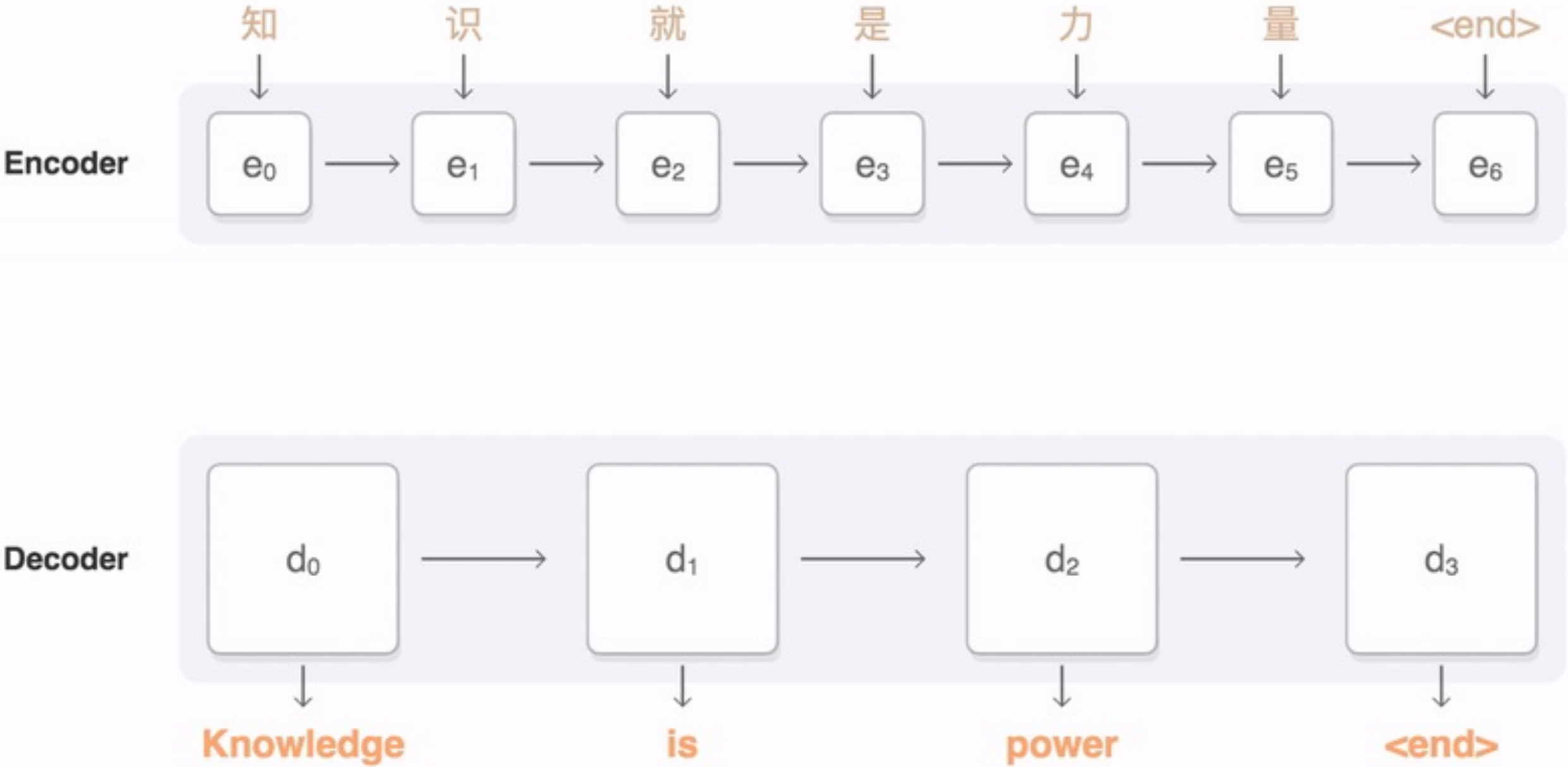


Seq2Seq-задачи



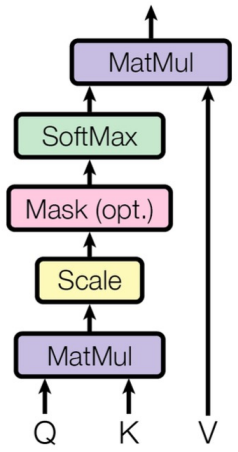
Проблема забывания RNN



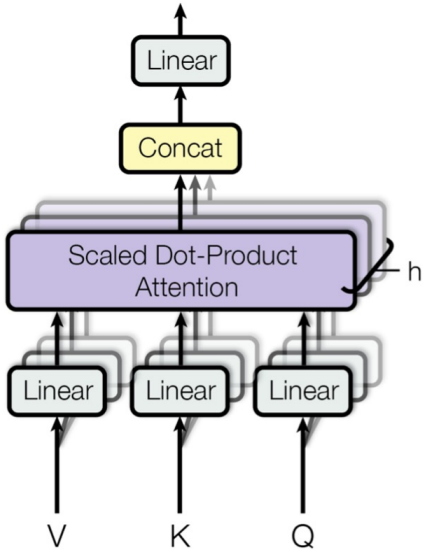

知
识
就
是
力
量
<end>
Encoder
e0
e1
e2
e3
e4
e5
e6
Decoder
d0
d1
d2
d3
Knowledge
is
power
<end>

3. Transformer и Attention

Scaled Dot-Product Attention

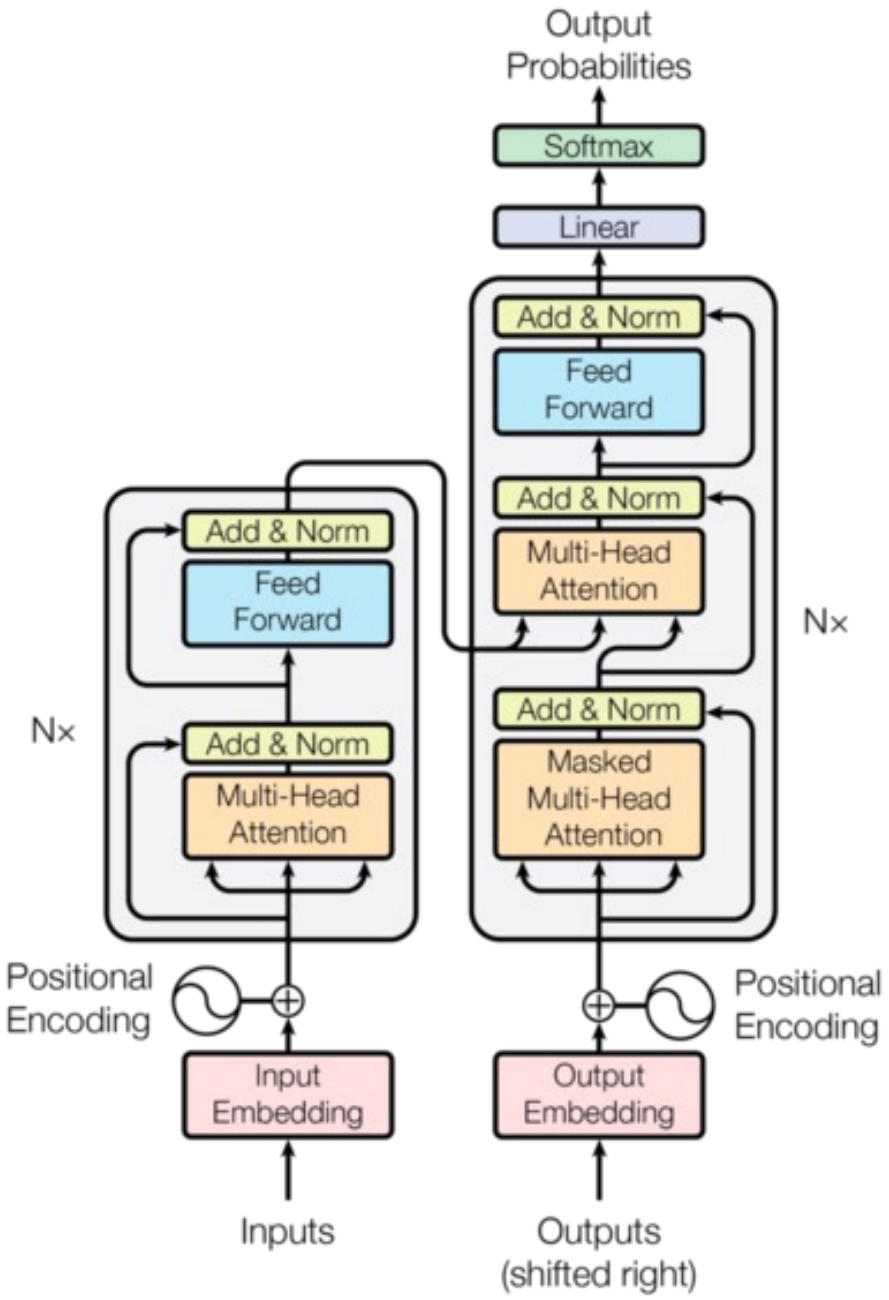


Multi-Head Attention



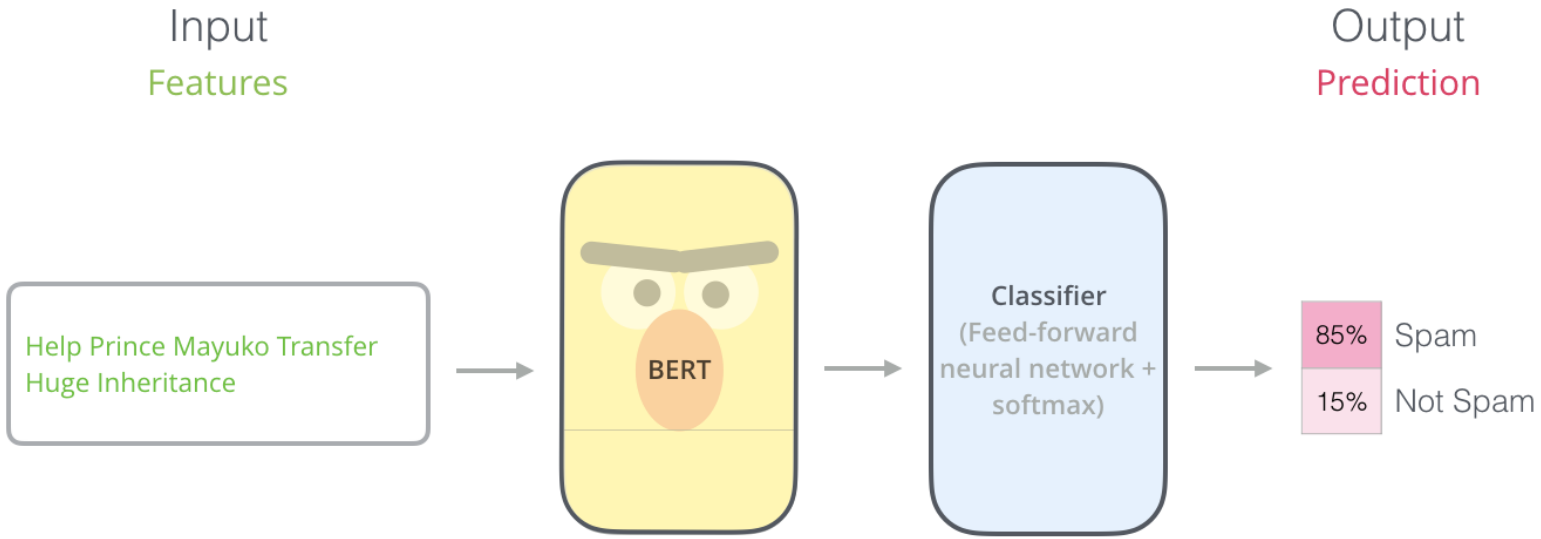
$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

TRANSFORMER



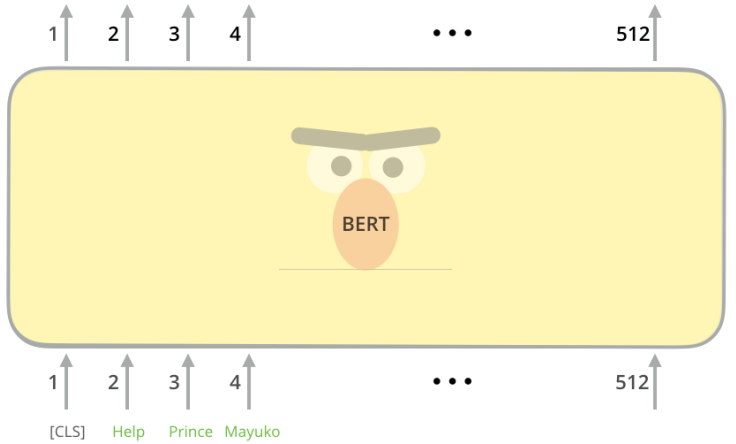
BERT (Bidirectional Encoder Representations from Transformers)

BERT разработан для решения задач, связанных с пониманием контекста и семантики в тексте. Он способен создавать контекстные векторные представления слов и фраз, что помогает в решении различных NLP-задач.

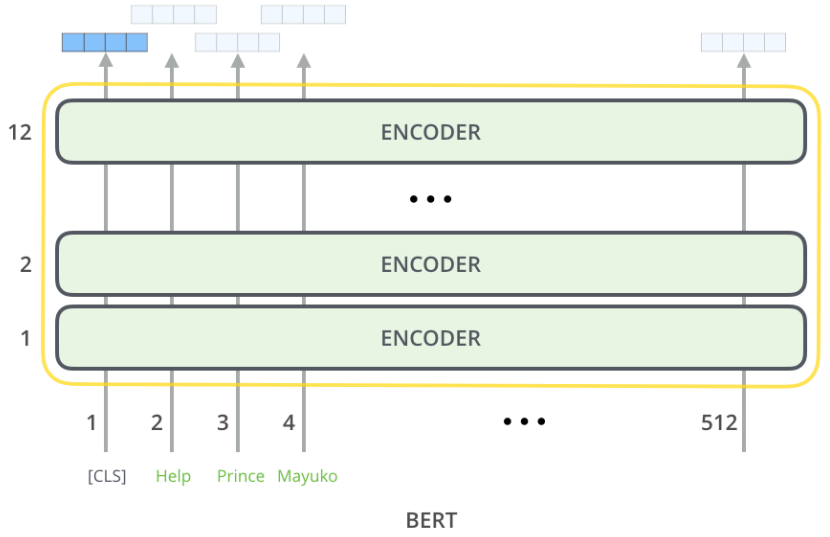


BERT: Архитектура

Высокоуровнево:

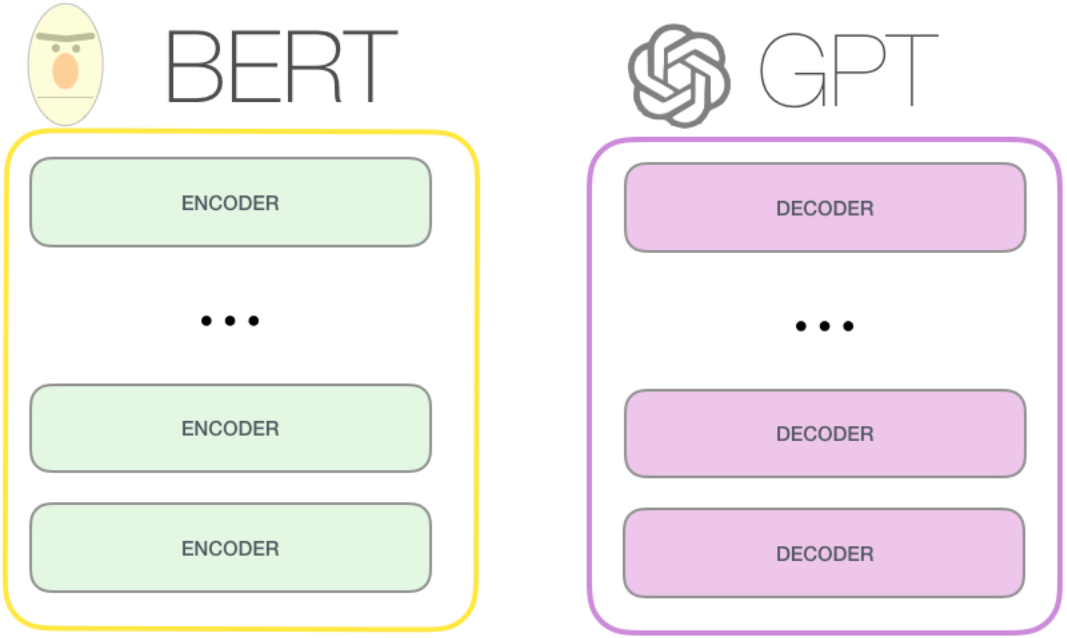


Чуть подробнее:

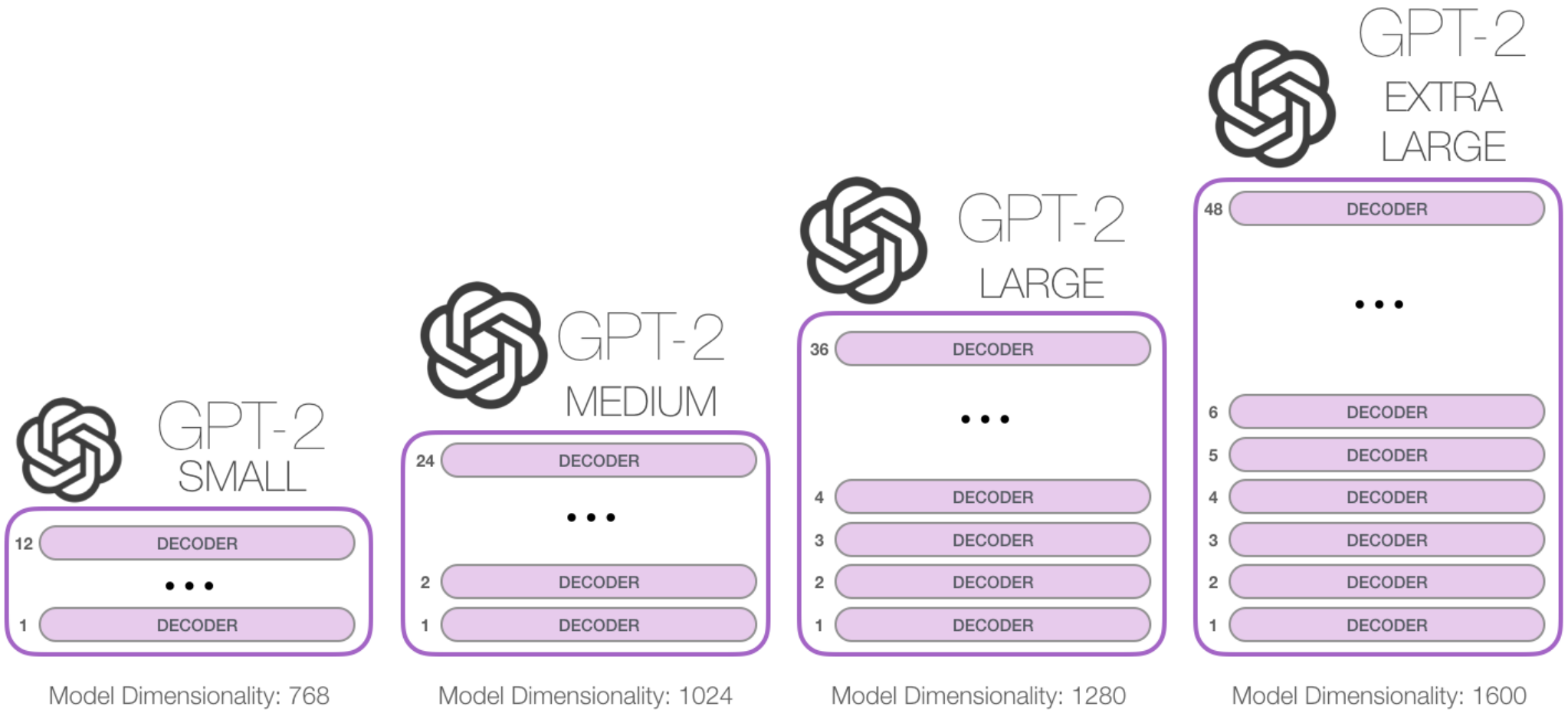


GPT (Generative Pre-trained Transformer)

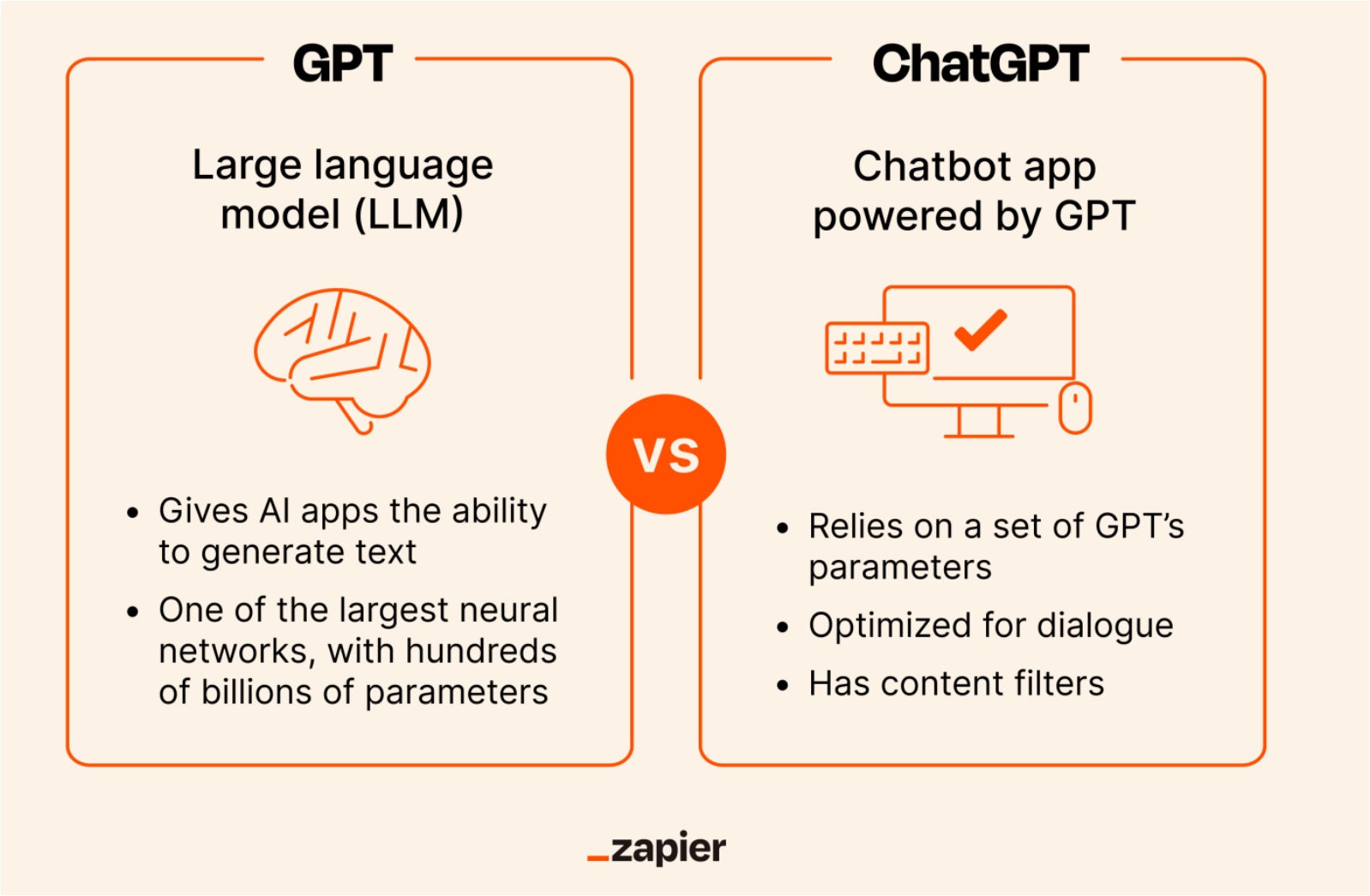
GPT разрабатывался для решения задачи генерации текста и понимания контекста на естественном языке. Он способен генерировать текст, продолжая начатый пользователем или предоставленный контекст, а также отвечать на вопросы и выполнять задачи, требующие понимания текста.



GPT: эволюция



GPT и ChatGPT



Введение в нейронные сети

Мотивация использования нейросетей

Существует несколько причин для использования нейронных сетей. Давайте разберемся со всеми по порядку.

Как мы действуем в классическом машинном обучении:

- У нас есть данные - информация об объектах. Для обучения моделей ***мы (на этапе сбора, а также обработки данных) конструируем и вычисляем признаки объектов***
- Затем на собранном датасете обучаем алгоритм

Мотивация использования нейросетей

Существует несколько причин для использования нейронных сетей. Давайте разберемся со всеми по порядку.

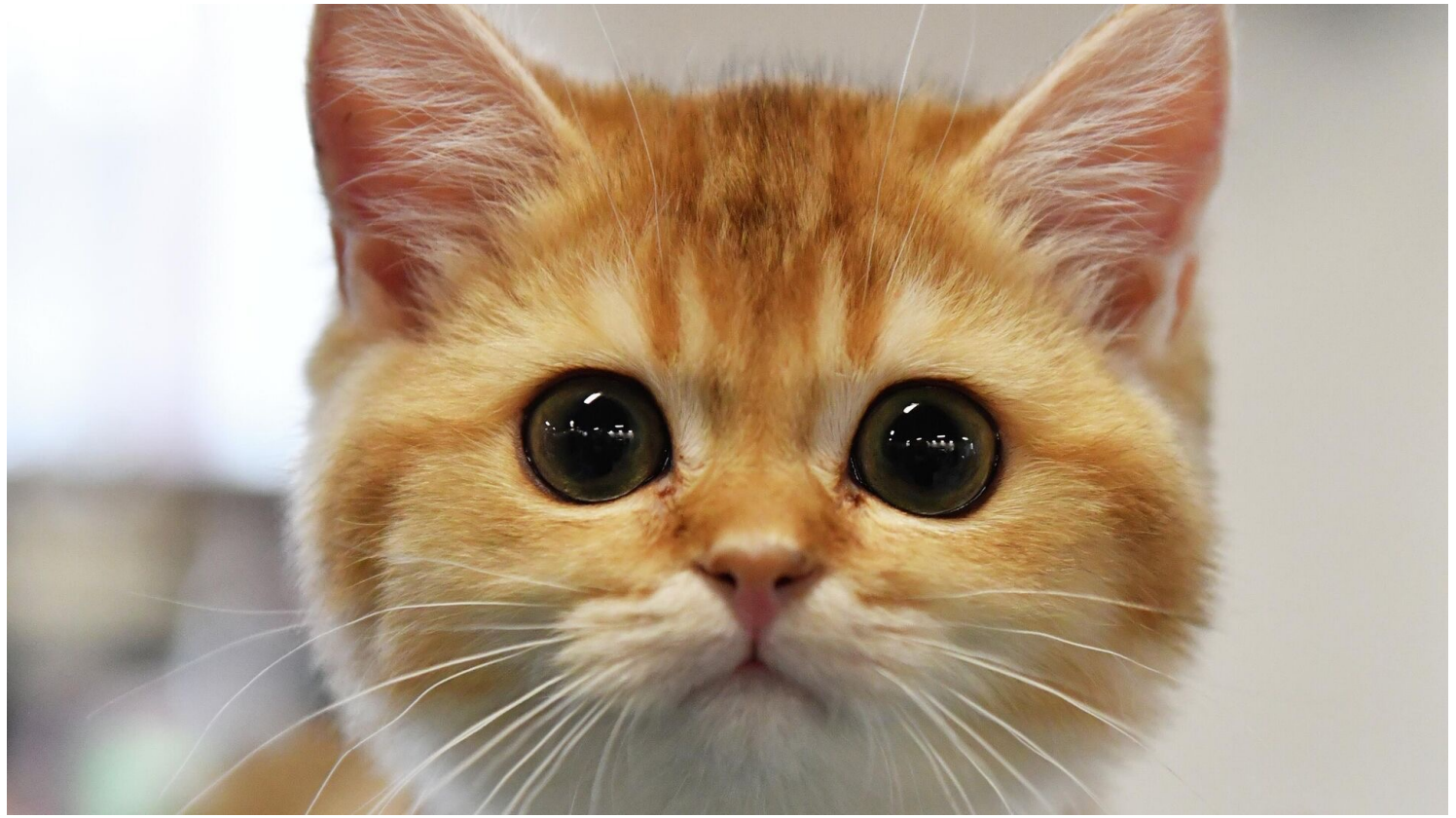
Как мы действуем в классическом машинном обучении:

- У нас есть данные - информация об объектах. Для обучения моделей ***мы (на этапе сбора, а также обработки данных) конструируем и вычисляем признаки объектов***
- Затем на собранном датасете обучаем алгоритм

В этом подходе работа с данными, а именно, извлечение необходимых признаков - лежит на человеке (на data scientist). Но не всегда легко понять, какие признаки нужны для того, чтобы хорошо решить задачу. А также далеко не всегда легко извлечь нужные признаки.

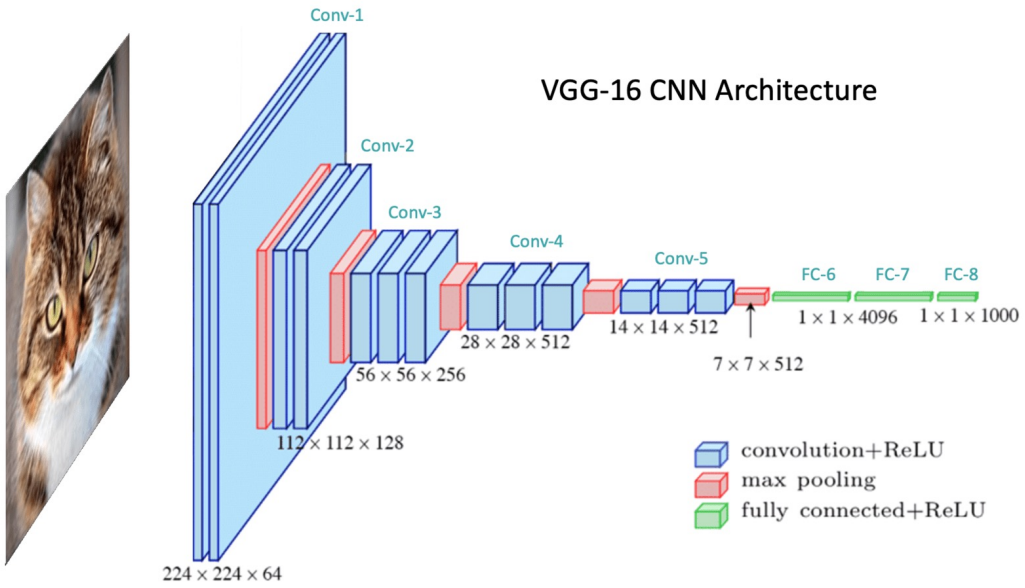
Мотивация использования нейросетей

Какие признаки извлечь из изображения для задачи классификации?

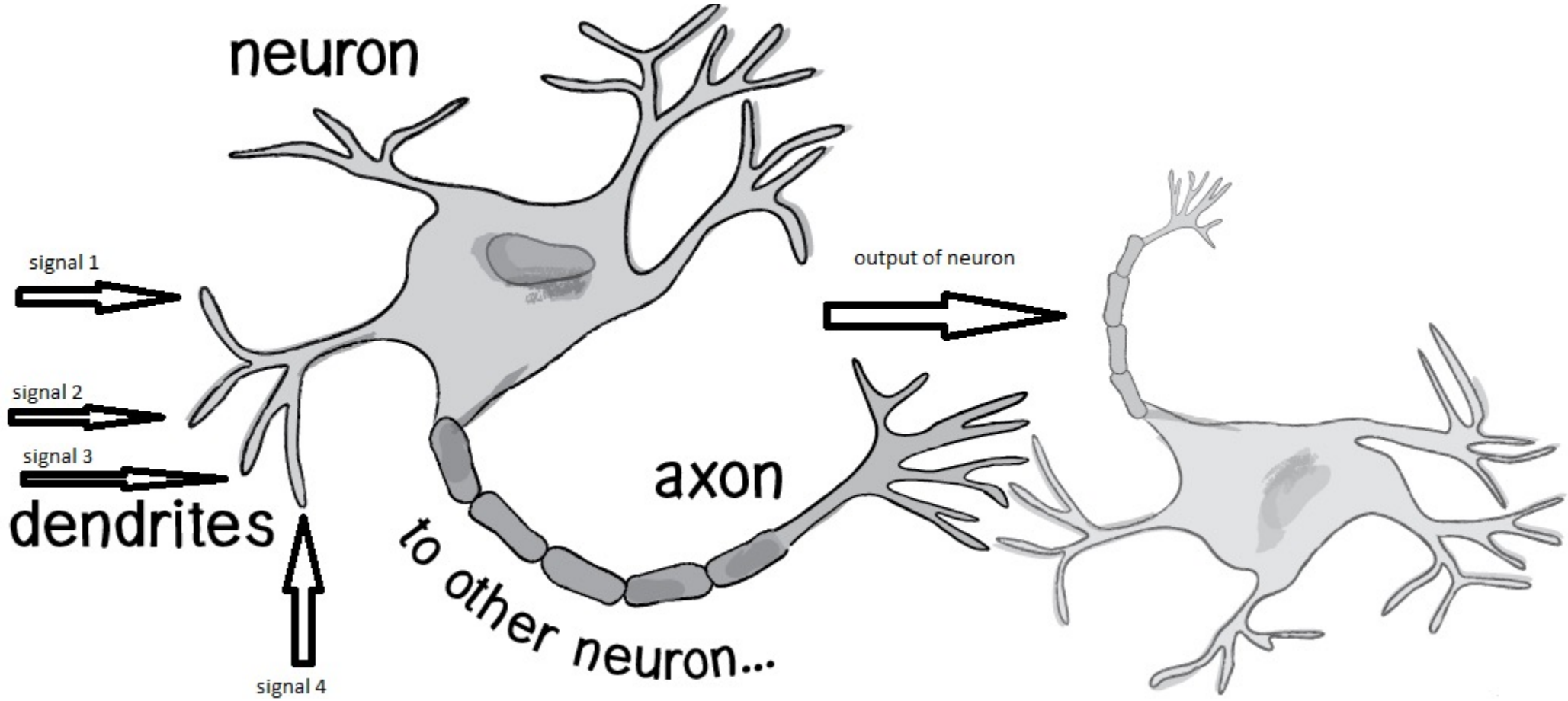


Мотивация использования нейросетей

На вход сети подается изображение (как матрица пикселей), а затем нейронная сеть последовательно извлекает из него все более сложные и содержательные признаки. Наконец, когда признаки получены, сеть обучается на них предсказывать ответ (в нашем примере - вид животного на изображении).

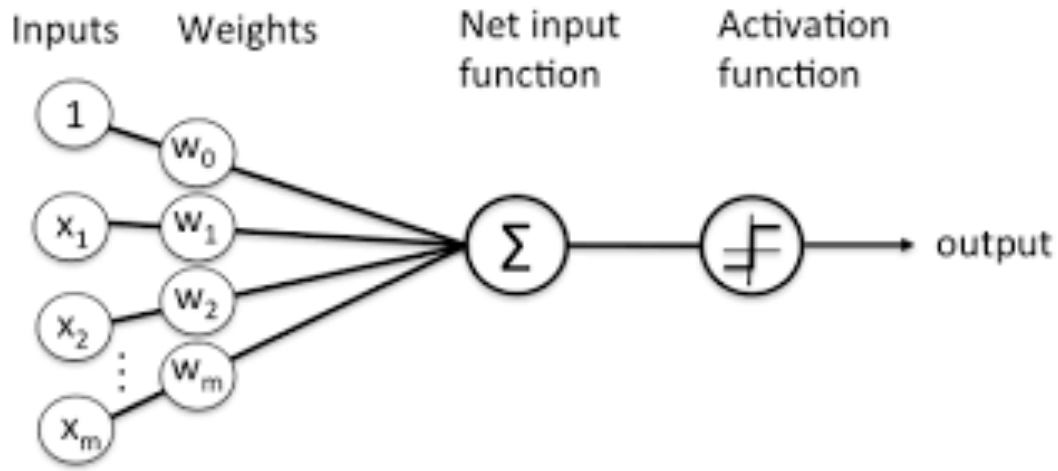


Биологический нейрон



Искусственный нейрон

$$a(x, w) = \sigma(w_0 + \sum_{j=1}^{n} w_j x_j)$$



Функции активации:

$$\sigma(z) = sign(z) \Rightarrow a(x, w) = sign[w_0 + \sum_{j=1}^{n} w_j x_j]$$

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \Rightarrow a(x, w) = \frac{1}{1 + \exp(-(w_0 + \sum_{j=1}^{n} w_j x_j))}$$

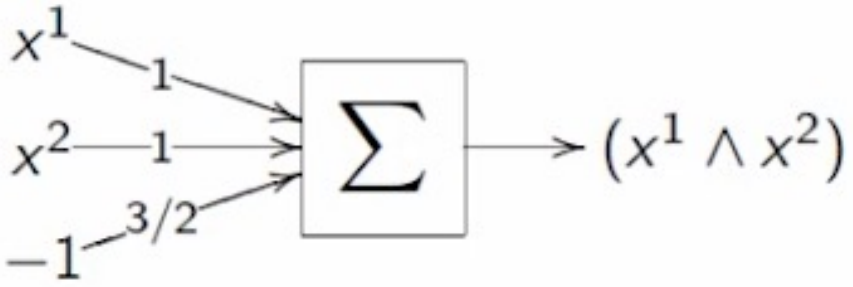
Какие функции реализуются нейроном?

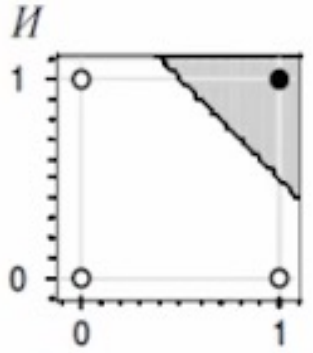
Функции И, ИЛИ, НЕ от бинарных переменных x^1 и x^2:

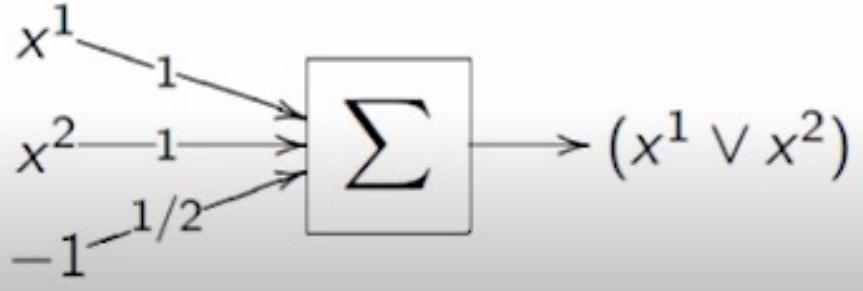
$$x^1 \wedge x^2 = \left[x^1 + x^2 - \tfrac{3}{2} > 0\right];$$

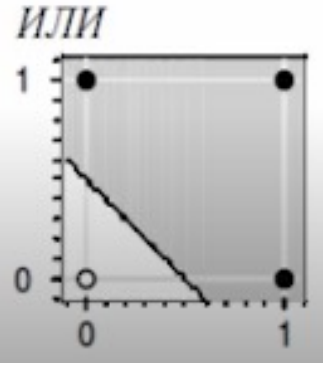
$$x^1 \vee x^2 = \left[x^1 + x^2 - \tfrac{1}{2} > 0\right];$$

$$\neg x^1 = \left[-x^1 + \tfrac{1}{2} > 0\right];$$



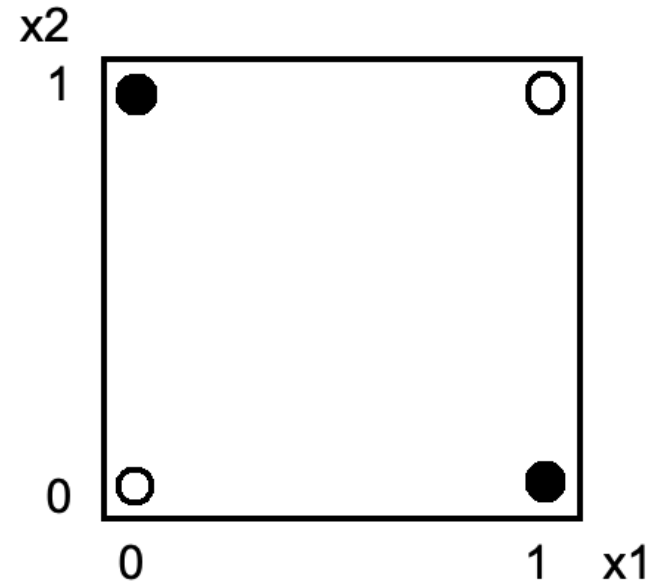






Пример функции, не реализуемой нейроном

XOR – сумма по модулю 2

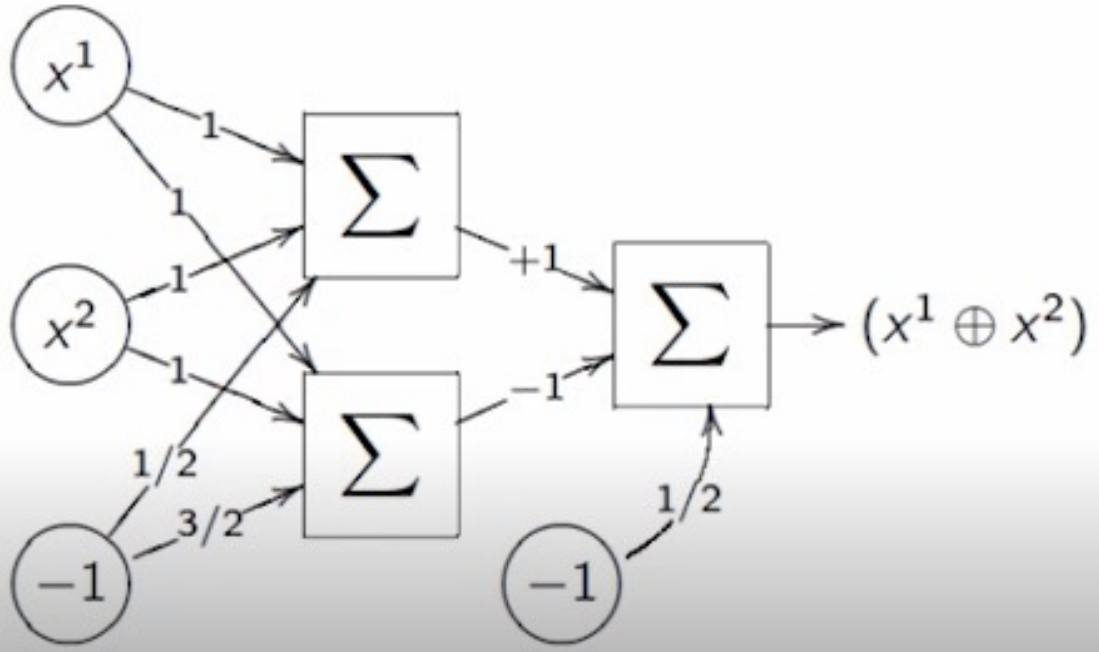


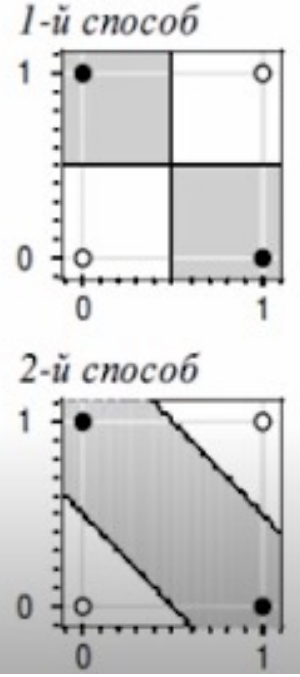
Пример функции, не реализуемой нейроном

Функция $x^1 \oplus x^2 = [x^1 \neq x^2]$ не реализуема одним нейроном.

Два способа реализации:

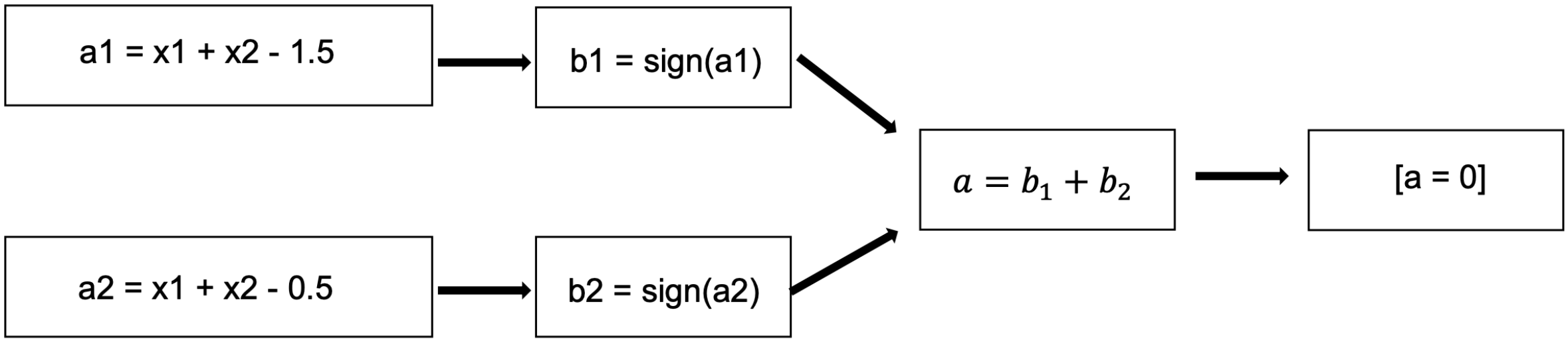
- Добавлением нелинейного признака:
  $x^1 \oplus x^2 = \left[x^1 + x^2 - 2x^1x^2 - \frac{1}{2} > 0\right]$;
- Сетью (двухслойной суперпозицией) функций И, ИЛИ, НЕ:
  $x^1 \oplus x^2 = \left[(x^1 \vee x^2) - (x^1 \wedge x^2) - \frac{1}{2} > 0\right]$.





Пример функции, не реализуемой нейроном

Решение задачи XOR – двухслойная нейронная сеть:

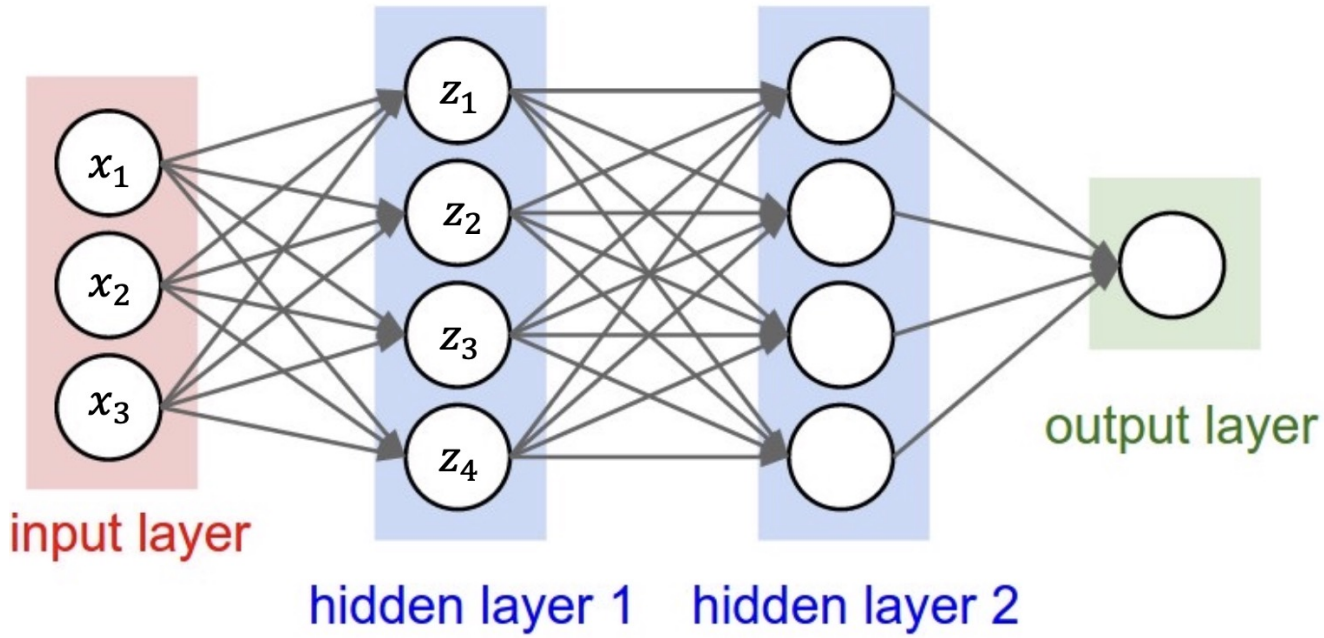


Полносвязная нейронная сеть

Шаг 0: На вход полносвязной сети подаются признаки объекта (они могут быть не очень осмысленными - например, яркости пикселей на изображении)

Шаг 1: Затем на основе входных признаков строится несколько *вспомогательных признаков* - то есть вычисляются суммы

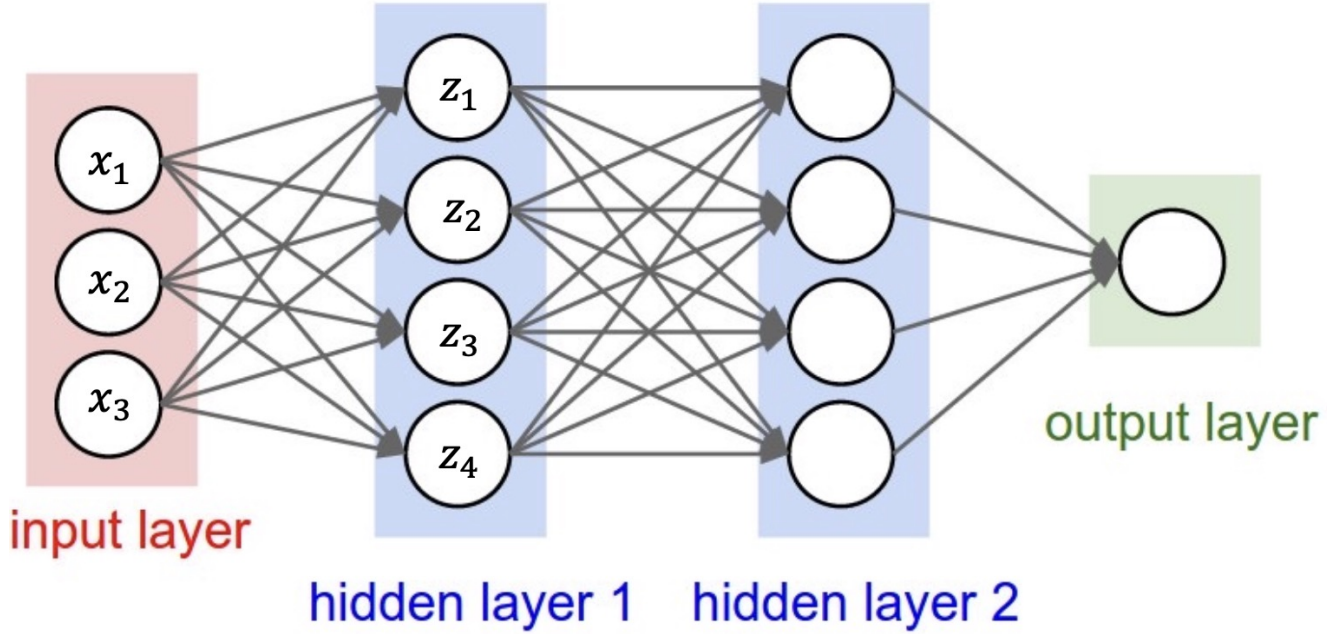
$$z_i = \sum_{i=1}^{d} w_{ij} x_j$$



Полносвязная нейронная сеть

Шаг 2: После вычисления вспомогательных признаков от них берутся некоторые нелинейные функции (в примере с XOR это были *sign* и индикатор). Они называются ***функциями активации***.

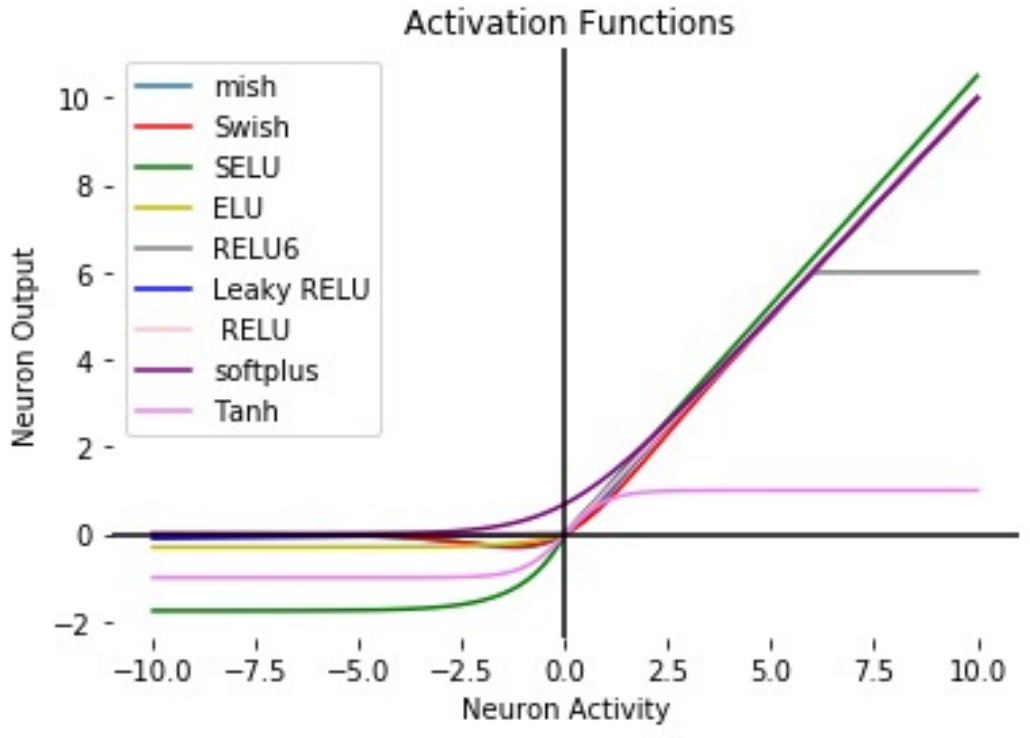
Затем шаги 1-2 можно несколько раз повторить. Вспомогательные признаки с идущей за ними функцией активации образуют ***полносвязный слой*** нейронной сети.



Функции активации

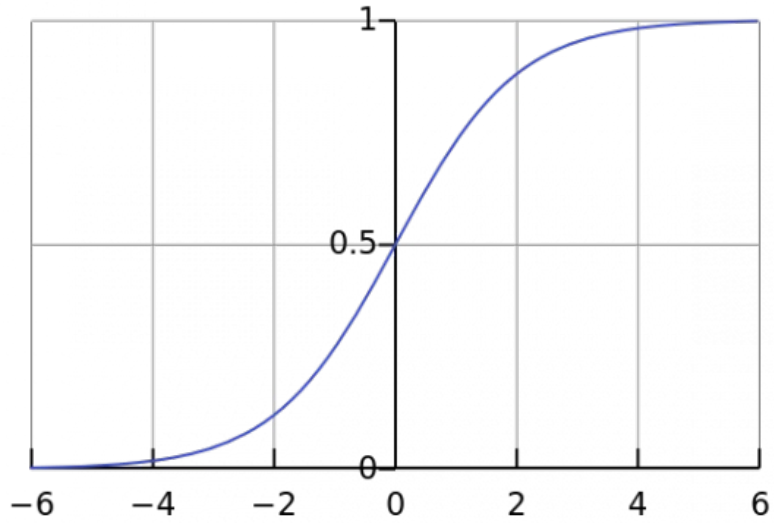
В каждом полносвязном слое после вычисления линейной комбинации признаков с предыдущего слоя мы применяем к результату некоторую нелинейную функцию - *функцию активации*.

Существует список чаще всего используемых функций активации, некоторые из них изображены на рисунке ниже:



Функции активации: сигмоида

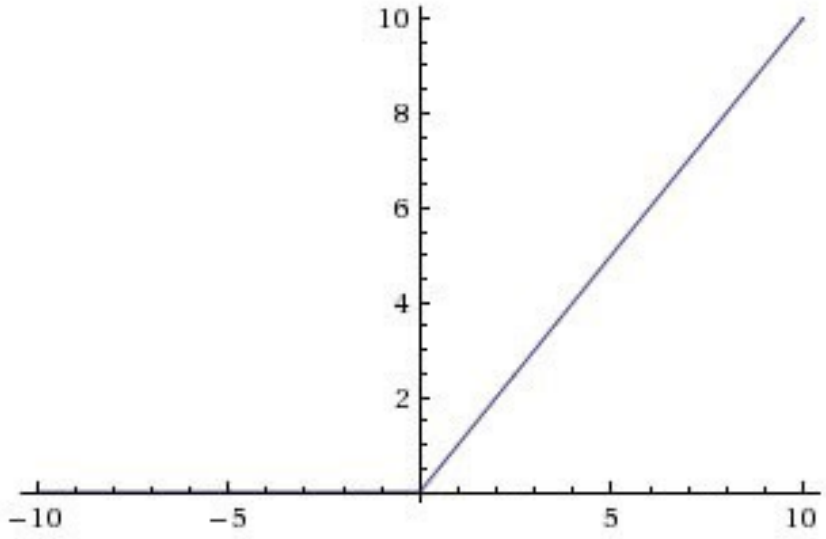
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



- Сигмоида хорошо подходит как функция активации на последнем слое нейронной сети в задачах бинарной классификации, так как в совокупности с подходящей функцией потерь для обучения сети (log-loss) на выходе мы получим вероятности классов.
- При этом у сигмоиды есть и недостаток, из-за которого ее стараются не использовать в промежуточных слоях сети: при увеличении $|x|$ значения функции $\sigma(x)$ слабо изменяются - это означает, что $\sigma'(x) \approx 0$ в этих областях. Близкая к нулю производная может приводить к занулению градиента функции потерь, а следовательно, обучение сети застопорится.

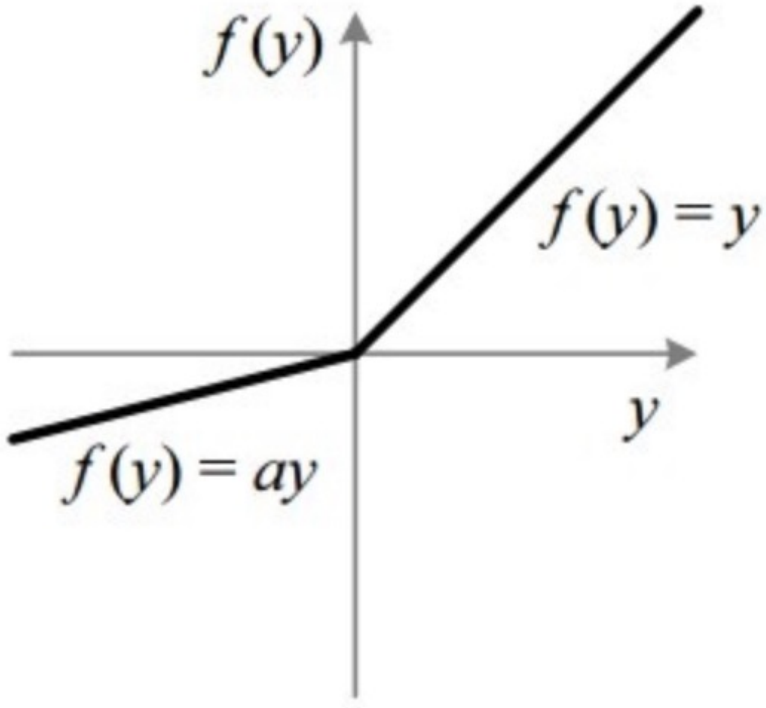
Функции активации: ReLU

$$ReLU(x) = \max\{0, x\}$$



- На первый взгляд кажется, что ReLU линейная функция, но это не так - она линейна лишь в каждой полуплоскости, а в совокупности нелинейна, поэтому вполне подходит в качестве функции активации.
- ReLU имеет свойство регуляризации - она зануляет нейроны, принимающие отрицательные значения. Это означает, что в каждый момент времени не все нейроны активны, что снижает переобучение

Функции активации: Leaky ReLU



- Предыдущий пункт также иногда можно отнести и к минусам ReLU - может произойти так, что некоторые нейроны большую часть времени зануляются, поэтому по ним не происходит обучения (зануление градиента). Для решения этой проблемы существуют вариации ReLU, такие как, например, Leaky ReLU:

Мощь полносвязных нейронных сетей

Теорема (Цыбенко): если у нас есть $y=g(x)$ - непрерывная функция от признаков x, то ***всегда можно построить двухслойную нейронную сеть, приближающую эту функцию сколь угодно точно***

Другими словами теорема утверждает, что любую задачу регрессии можно очень хорошо решить даже двухслойной нейронной сетью! То есть нейронные сети ОЧЕНЬ МОЩНЫЕ!

Демонстрация работы нейронных сетей

https://playground.tensorflow.org