

Основы глубинного обучения

Лекция 1

Введение в глубинное обучение

Евгений Соколов

esokolov@hse.ru

НИУ ВШЭ, 2025

Чем будем заниматься?

Dogs vs. Cats

Create an algorithm to distinguish dogs from cats



Kaggle · 213 teams · 7 years ago

[Overview](#) [Data](#) [Notebooks](#) [Discussion](#) [Leaderboard](#) [Rules](#)

Overview

Description

Prizes

Evaluation

Winners

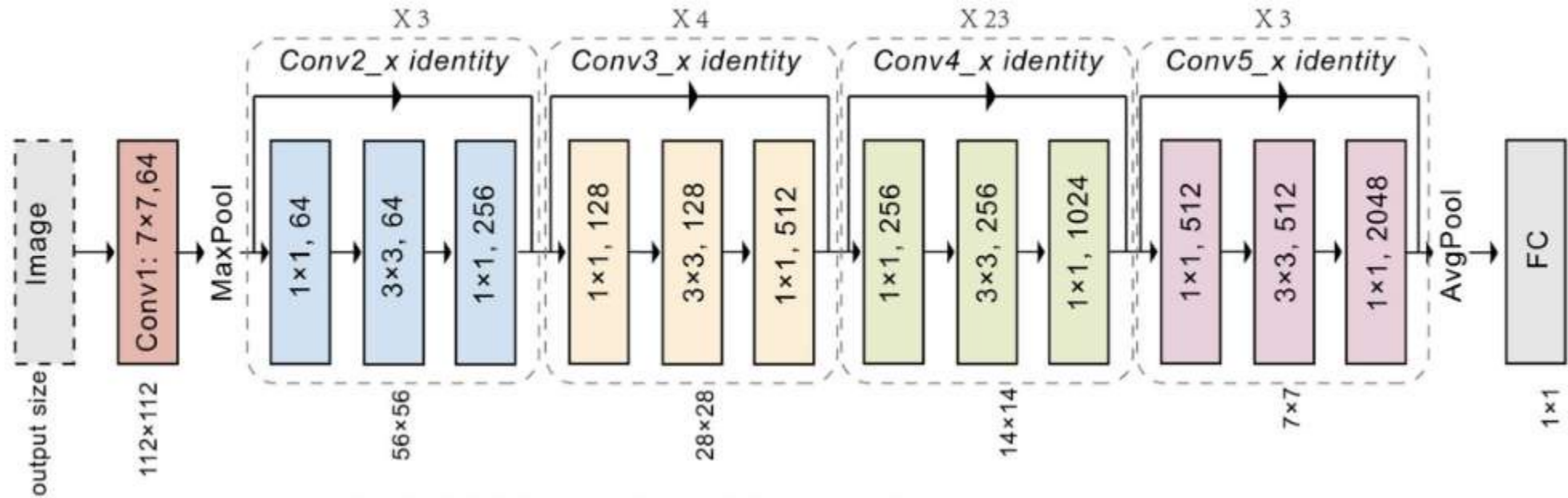
In this competition, you'll write an algorithm to classify whether images contain either a dog or a cat. This is easy for humans, dogs, and cats. Your computer will find it a bit more difficult.



Классическое компьютерное зрение

1. Считаем признаки (есть ли усы, какой формы уши, какой длины хвост, ...)
 2. Обучаем на них градиентный бустинг
- Посчитать признаки — целая история

Современное компьютерное зрение



Классическое NLP

1. Подсчитываем статистику, как часто то или иное слово встречается после данного
2. Генерируем следующее слово из этого распределения

"Manure, almond gelato and frozen pies, you are also had it was in one but it will post office buildings s
ucks). their chinese food. comfort food while they liked their lids ripped off. it an early morning of jon
still a spade so maybe too much. the same. but, at the baked rigatoni, and not in other options and it see
ms odd taste). our visit). i go to nfl kickoff arrived with \$. that's about when you come down hoyt street
is actually higher than impressed with a regular theater! so at it, halfway through their pork and though
i've"

Современное NLP

Create a lively and interactive visual experience that reacts to music, user interaction or real-time data. Your animation should include smooth transitions and colorful and lively visuals with an engaging flow in the experience. The animation should feel organic and responsive to the music, user interaction or live data and facilitate an experience that is immersive and captivating. Complete this project using JavaScript or React, and allow for easy customization to set the mood for other experiences.



jsx

Copy

```
speedX: Math.random() * 2 - 1,
speedY: Math.random() * 2 - 1,
color: interpolateColor(
  primaryColor,
  secondaryColor,
  Math.random()
),
originalSize: 2 + Math.random() * 4,
angle: Math.random() * Math.PI * 2,
frequency: 0.1 + Math.random() * 0.3,
});
}
particlesRef.current = particles;
};

// Color interpolation helper
const interpolateColor = (color1, color2, factor) => {
  const result = color1.slice(1).match(/.{2}/g).map((hex, index) => {
    const value1 = parseInt(hex, 16);
    const value2 = parseInt(color2.slice(1).match(/.{2}/g)[index], 16);
    const value = Math.round(value1 + (value2 - value1) * factor)
      .toString(16)
      .padStart(2, '0');
    return value;
  });
  return `#${result.join('')}`;
};

// Handle window resize
useEffect(() => {
  const handleResize = () => {
    const canvas = canvasRef.current;
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;
```

Успехи в глубинном обучении

- Изображения и видео
- Трёхмерное компьютерное зрение
- Тексты (+код, математика...)
- Звук и речь
- Рекомендательные системы
- Агентные системы
- ...

Организационное

Про курс

- wiki: http://wiki.cs.hse.ru/Основы_глубинного_обучения
- Канал: @iad_2025
- Домашние задания
- Проверочные работы (неоцениваемые)
- Контрольная работа
- Письменный экзамен
- Автоматы — решим позже

Про оценку

$$O_{\text{итоговая}} = 0.4 * ДЗ + 0.3 * КР + 0.3 * Э$$

Примерный план курса

- Метод обратного распространения ошибки
- Полносвязные сети
- Сверточные сети
- Методы оптимизации для глубинного обучения
- Работа с последовательностями

численность

лекция

backprop

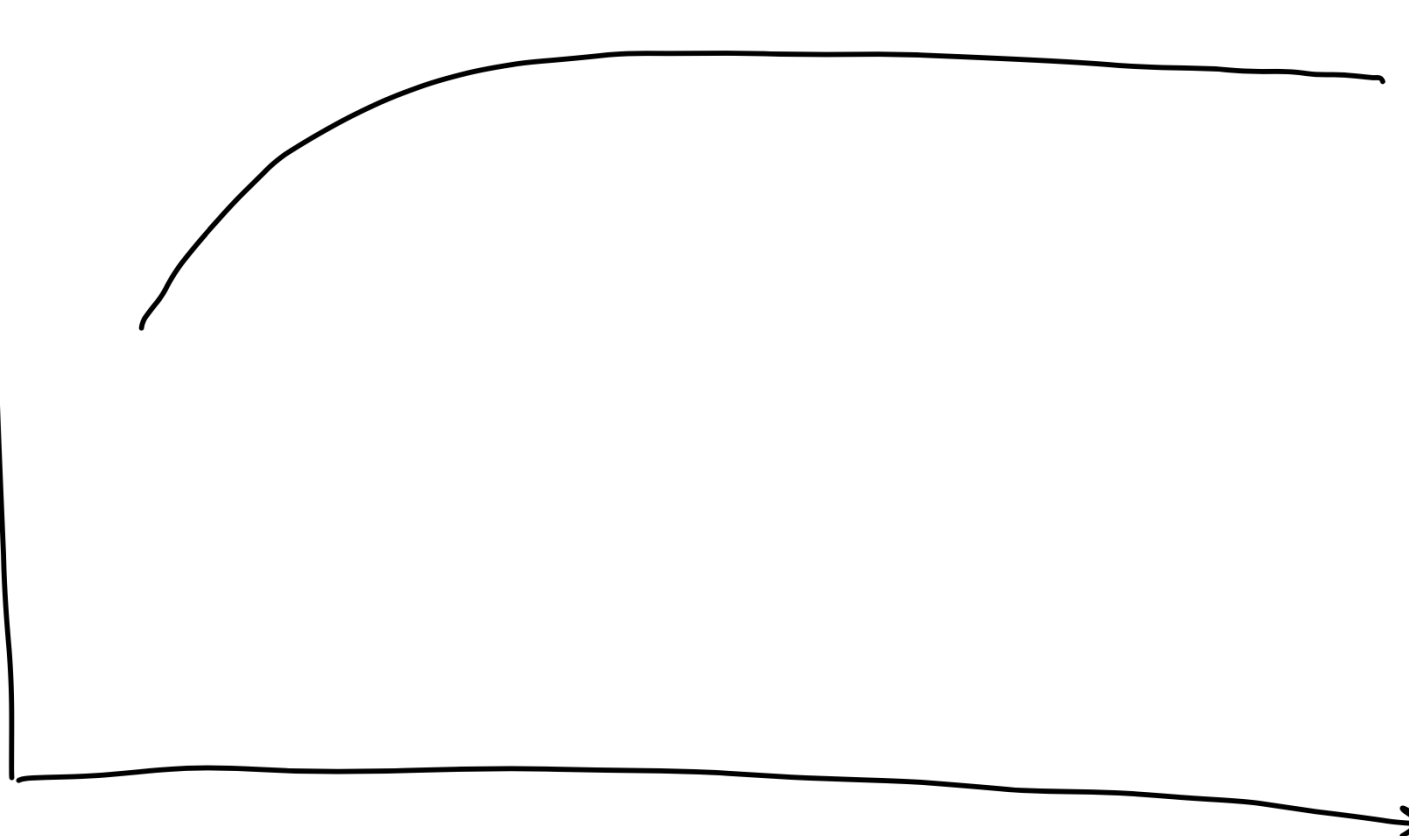
генерация

трансформер

цели

Скорость

до сих пор



время

Зачем нужны нейронные сети?

Предсказание стоимости квартиры

- Линейная модель:

$$a(x) = w_0 + w_1 * (\text{площадь}) + w_2 * (\text{этаж}) \\ + w_3 * (\text{расстояние до метро}) + \dots$$

- Вряд ли признаки не связаны между собой

Предсказание стоимости квартиры

- Линейная модель с полиномиальными признаками:

$$\begin{aligned} a(x) = & w_0 + w_1 * (\text{площадь}) + w_2 * (\text{этаж}) \\ & + w_3 * (\text{расстояние до метро}) + w_4 * (\text{площадь})^2 \\ & + w_5 * (\text{этаж})^2 + w_6 * (\text{расстояние до метро})^2 \\ & + w_7 * (\text{площадь}) * (\text{этаж}) + \dots \end{aligned}$$

- Может быть сложно интерпретировать модель
- Что такое $(\text{расстояние до метро}) * (\text{этаж})^2$?

Градиентный бустинг

$$a_N(x) = \sum_{n=1}^N b_n(x)$$

- Обучение N -й модели:

$$\frac{1}{\ell} \sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + b_N(x_i)) \rightarrow \min_{b_N(x)}$$

Градиентный бустинг

- Обучение N -й модели:

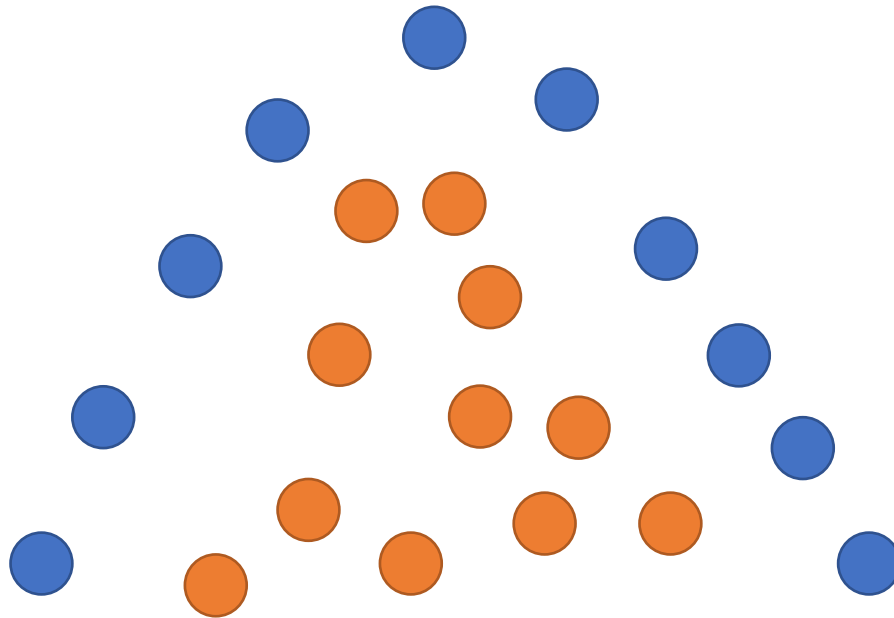
$$\frac{1}{\ell} \sum_{i=1}^{\ell} \left(b_N(x_i) - s_i^{(N)} \right)^2 \rightarrow \min_{b_N(x)}$$

$$s_i^{(N)} = - \frac{\partial}{\partial z} L(y_i, z) \Big|_{z=a_{N-1}(x_i)} \text{ — сдвиги}$$

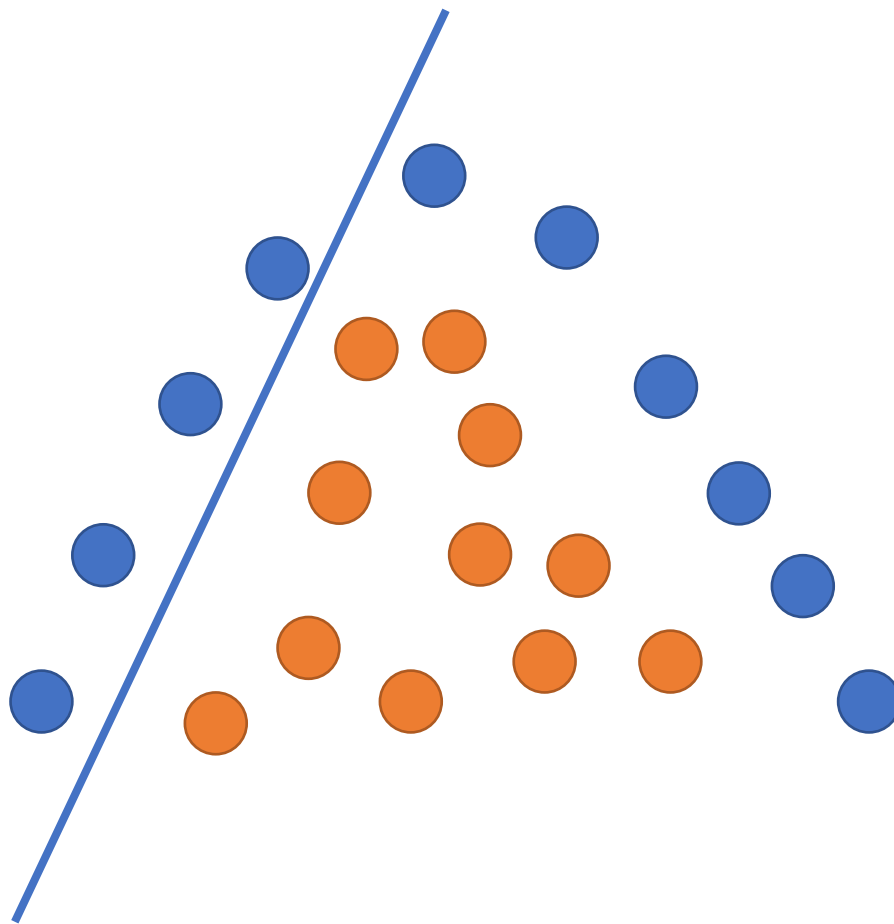
Кратко о предыдущем курсе

- Линейные модели обучаются градиентным спуском, но плохо подходят для поиска сложных закономерностей
- Решающие деревья и их композиции дают отличные результаты, но обучать их трудно

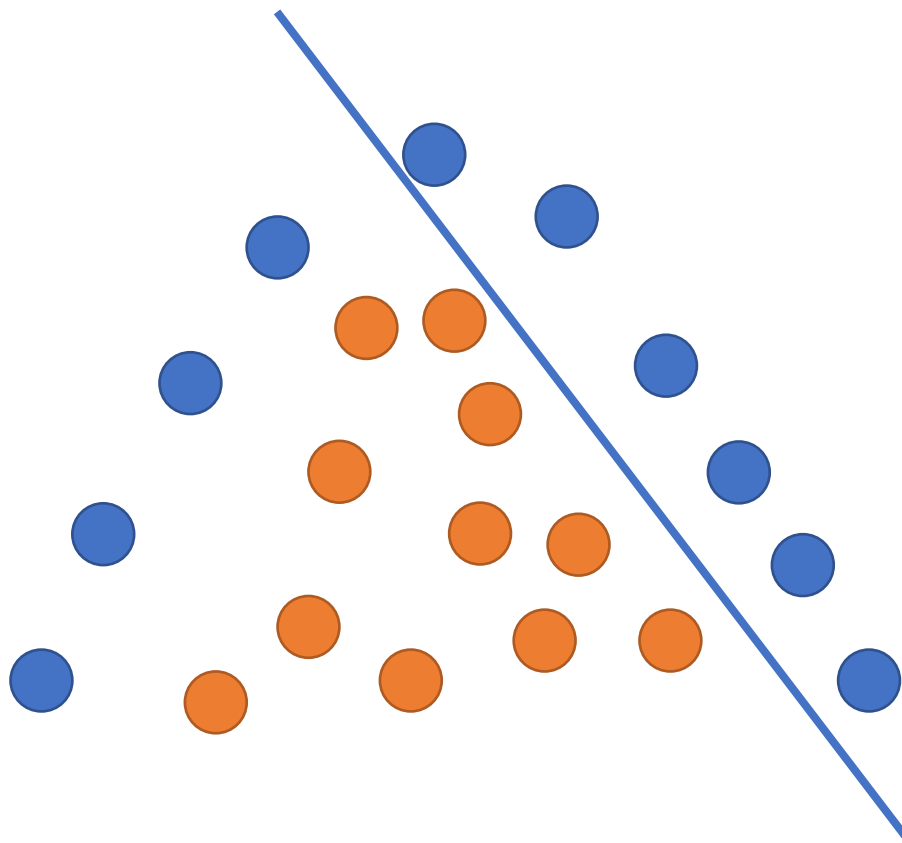
Нелинейные закономерности



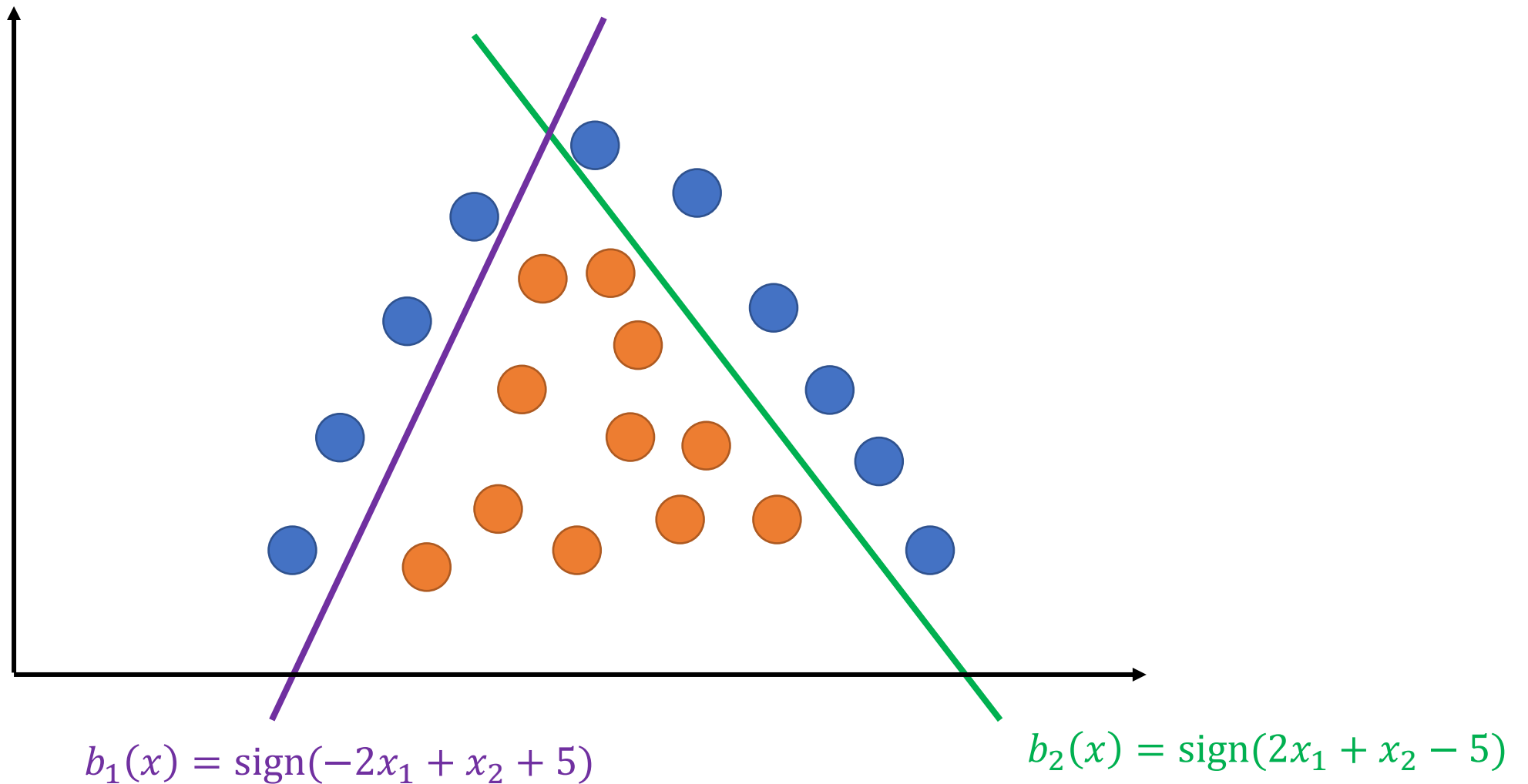
Нелинейные закономерности



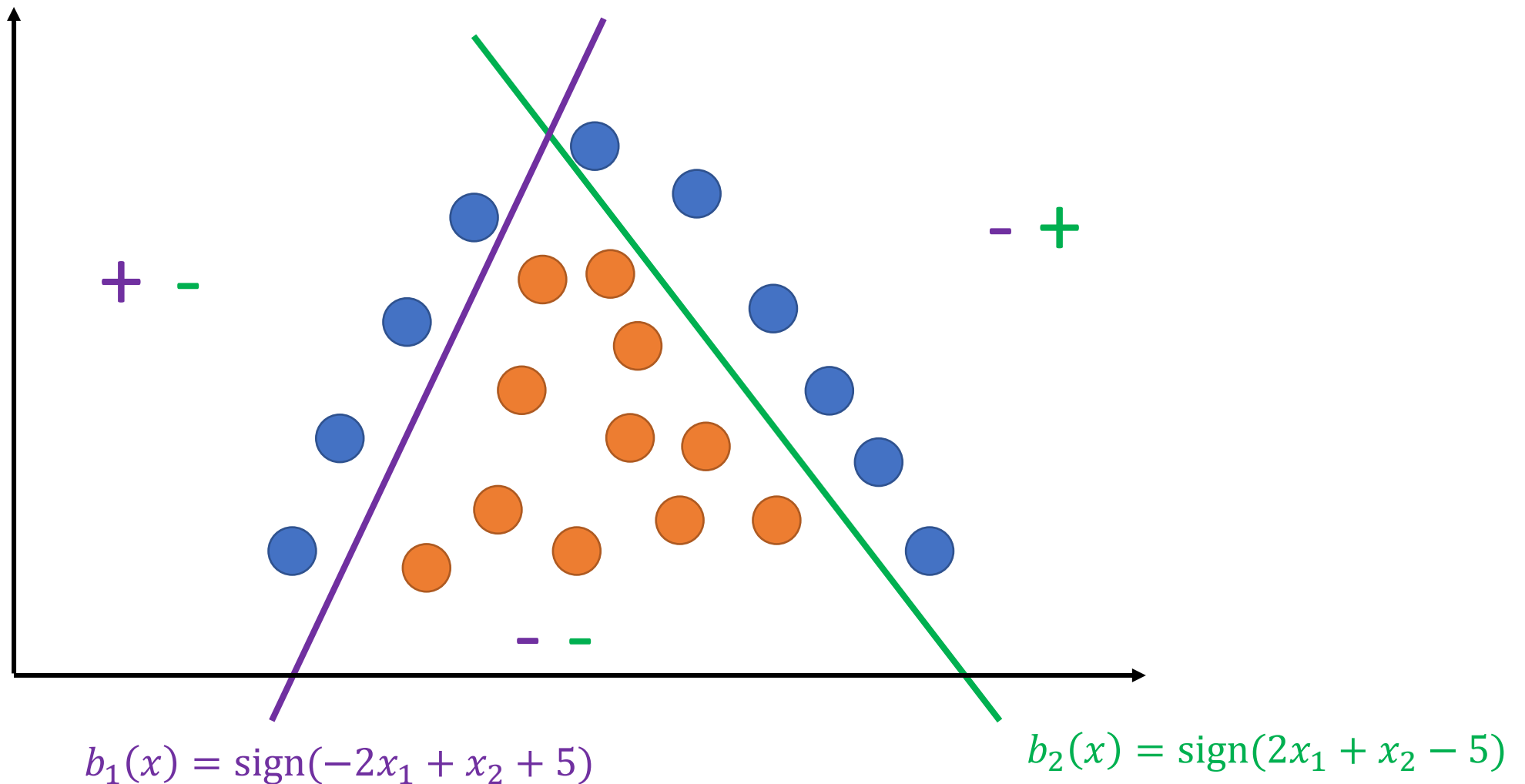
Нелинейные закономерности



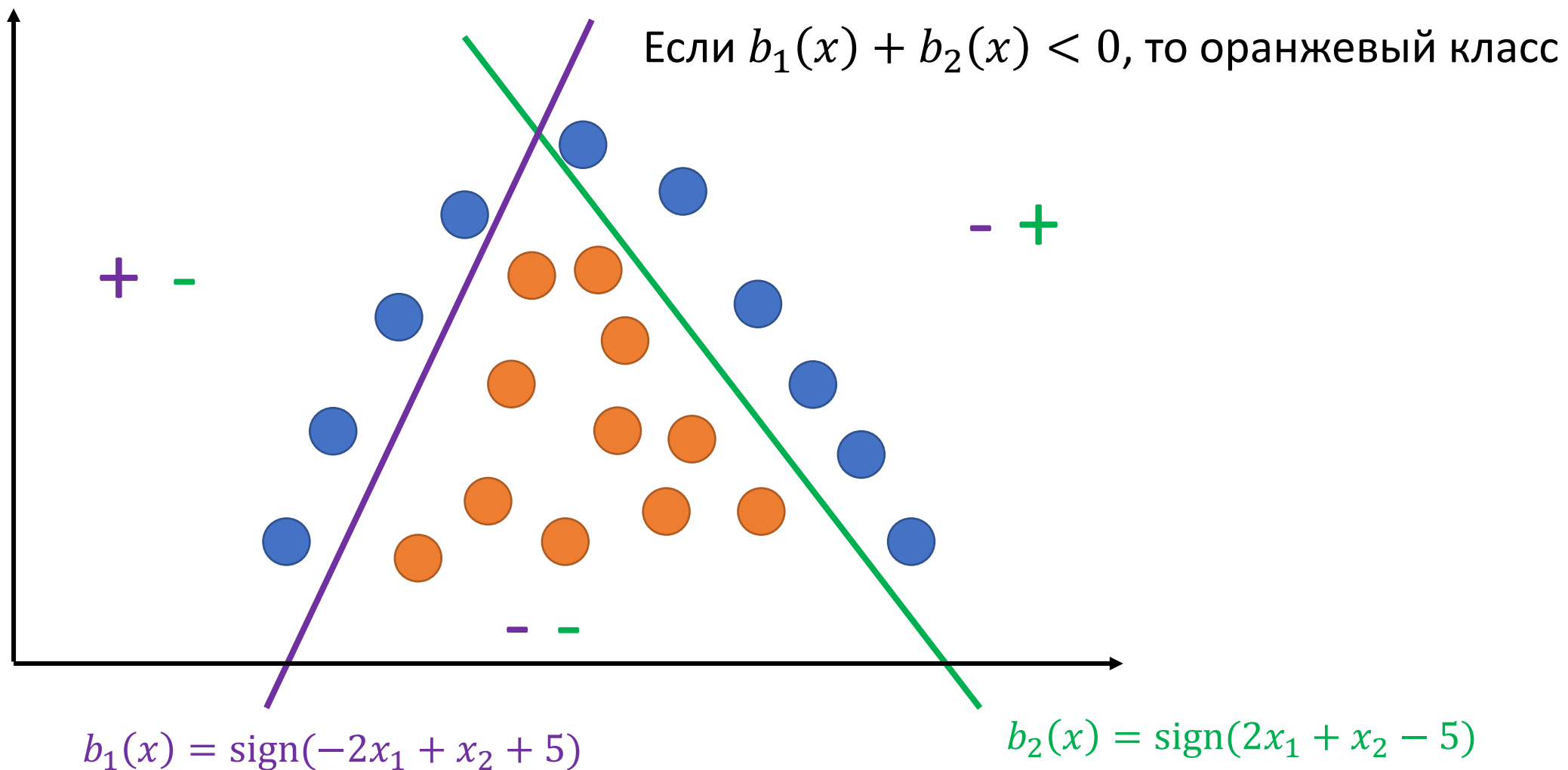
Нелинейные закономерности



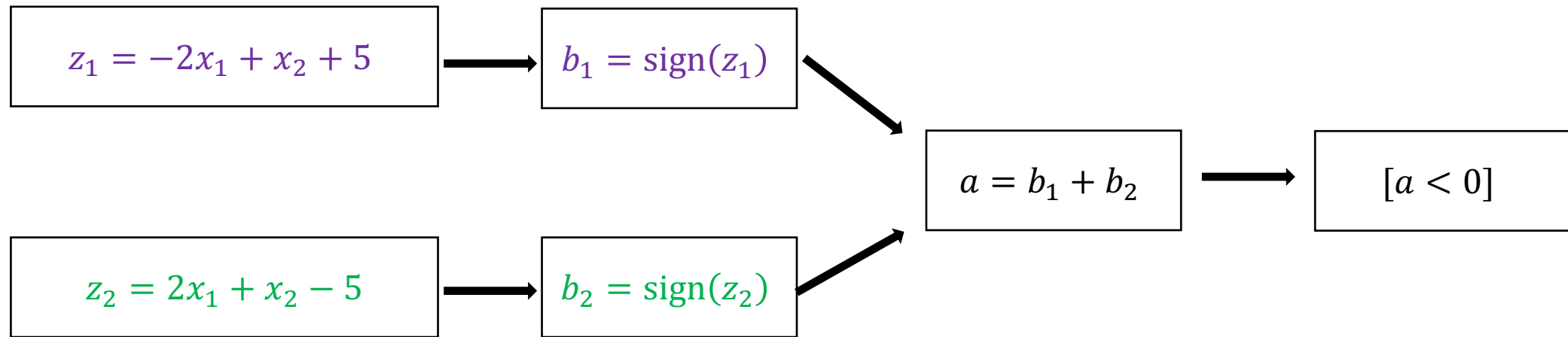
Нелинейные закономерности



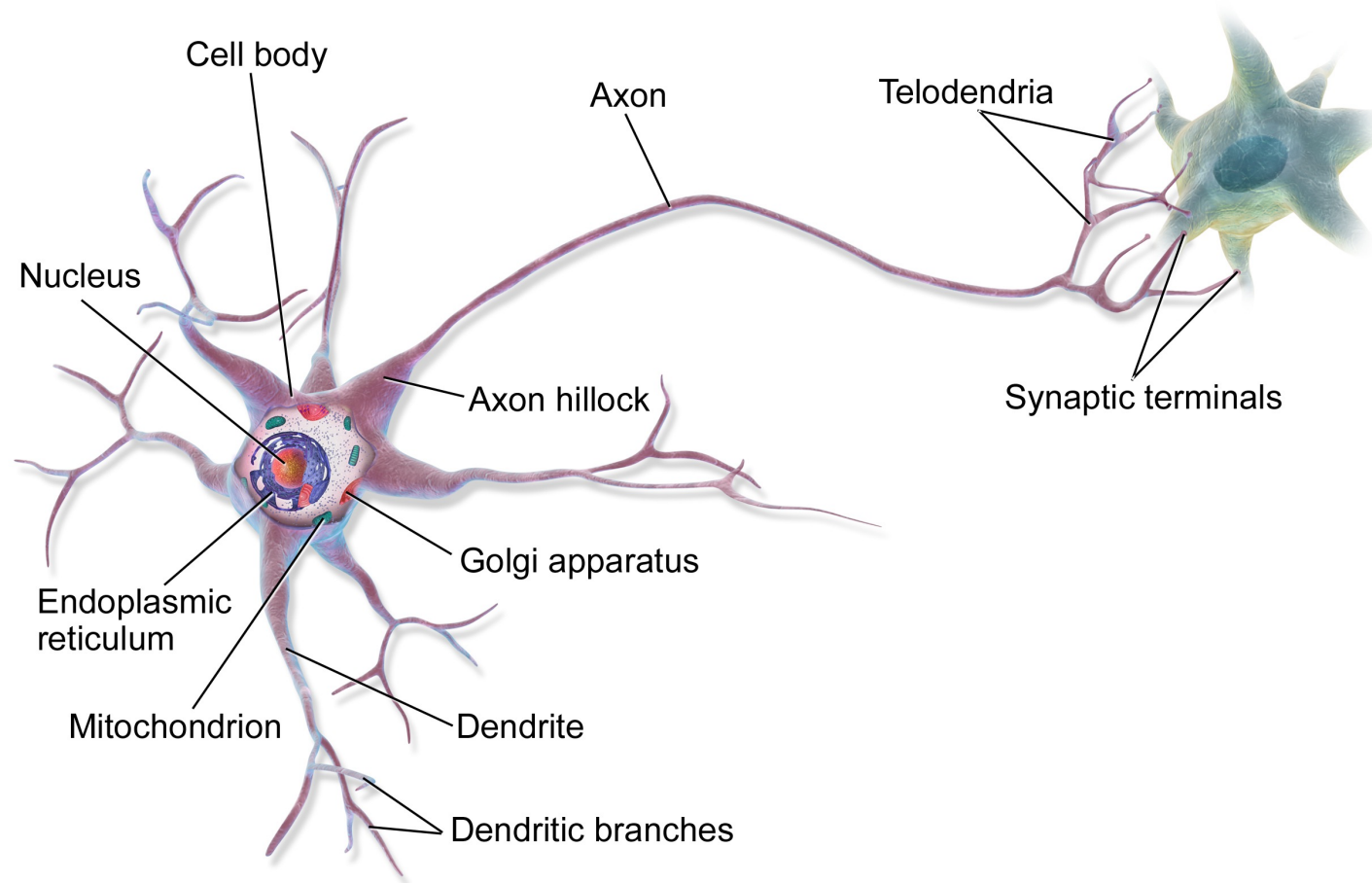
Нелинейные закономерности



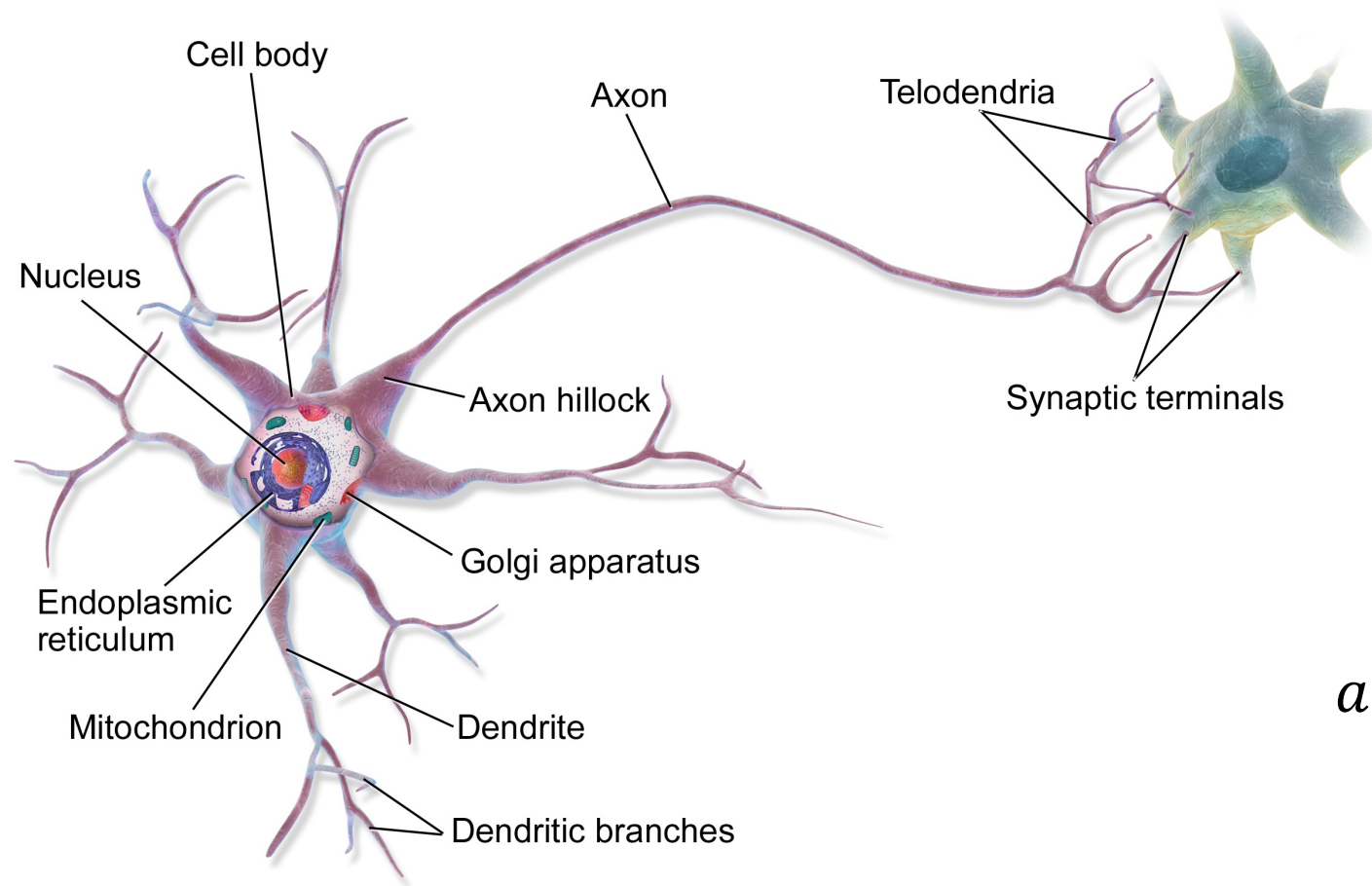
Нелинейные закономерности



Нейрон

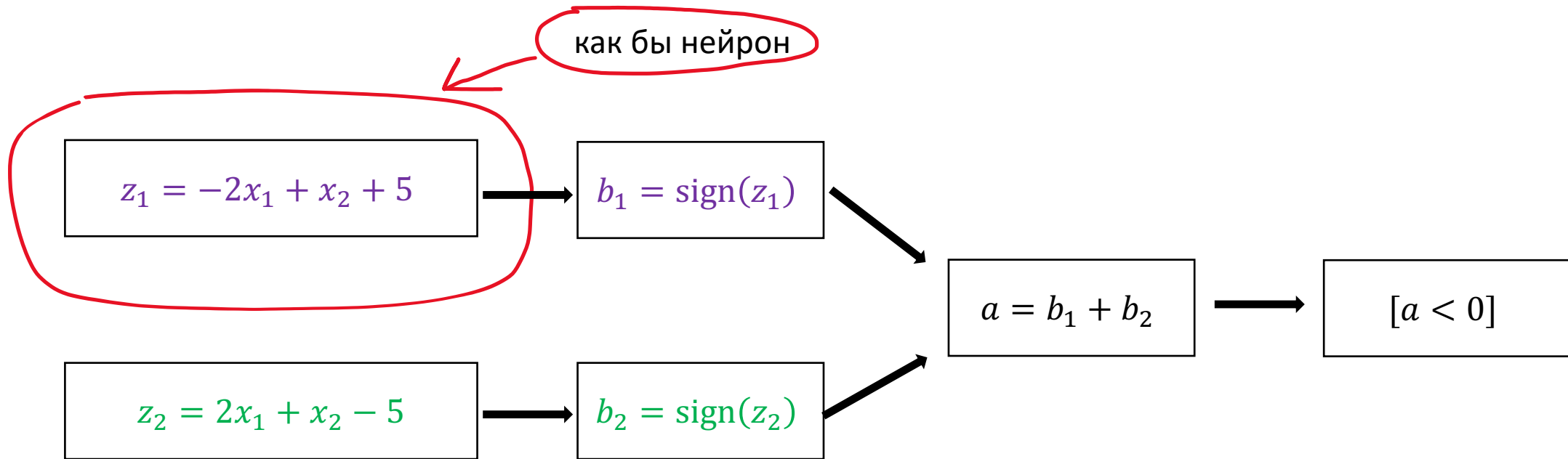


Нейрон



$$a(x) = \sum_{j=1}^d w_j x_j$$

Нелинейные закономерности



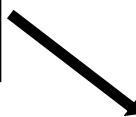
Нелинейные закономерности

как бы нейронная сеть

$$z_1 = -2x_1 + x_2 + 5$$



$$b_1 = \text{sign}(z_1)$$



$$z_2 = 2x_1 + x_2 - 5$$



$$b_2 = \text{sign}(z_2)$$



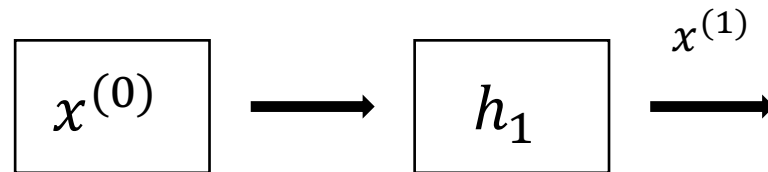
$$a = b_1 + b_2$$



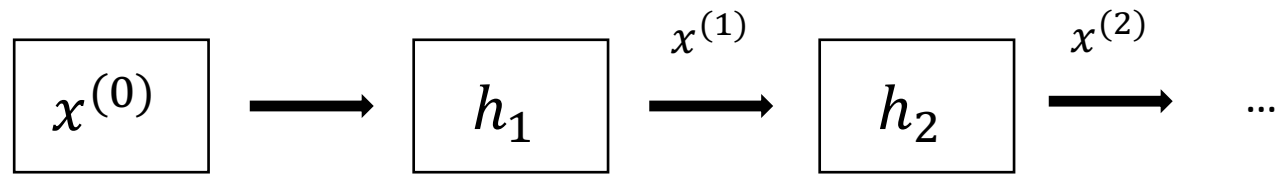
$$[a < 0]$$

Граф вычислений (или нейронная сеть)

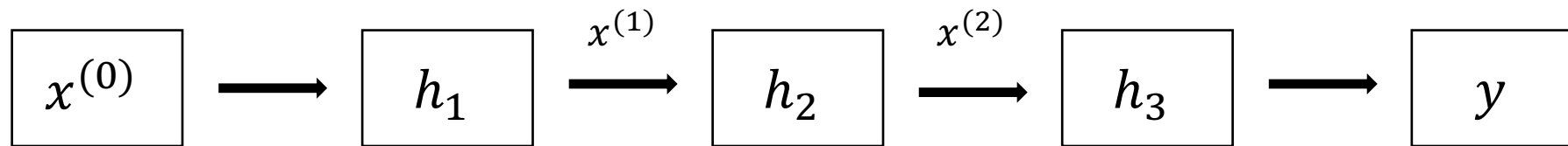
- $x^{(0)}$ — признаки объекта
- $h_1(x)$ — преобразование («слой»)
- $x^{(1)}$ — результат



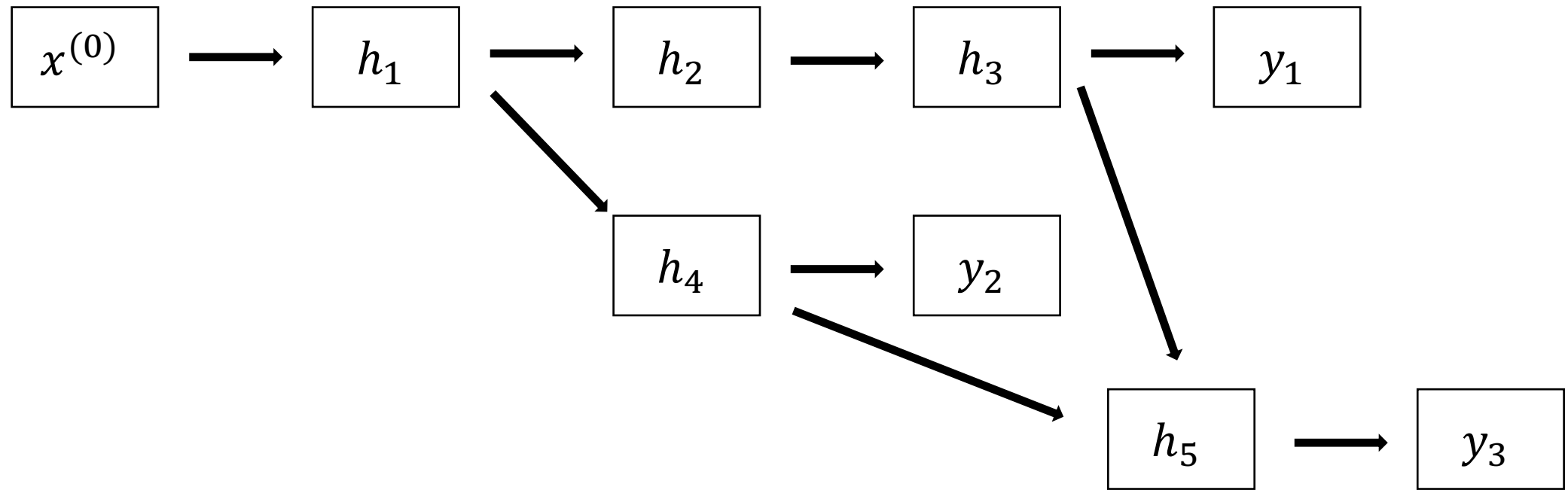
Граф вычислений (или нейронная сеть)



Граф вычислений (или нейронная сеть)



Граф вычислений (или нейронная сеть)



Полносвязные слои

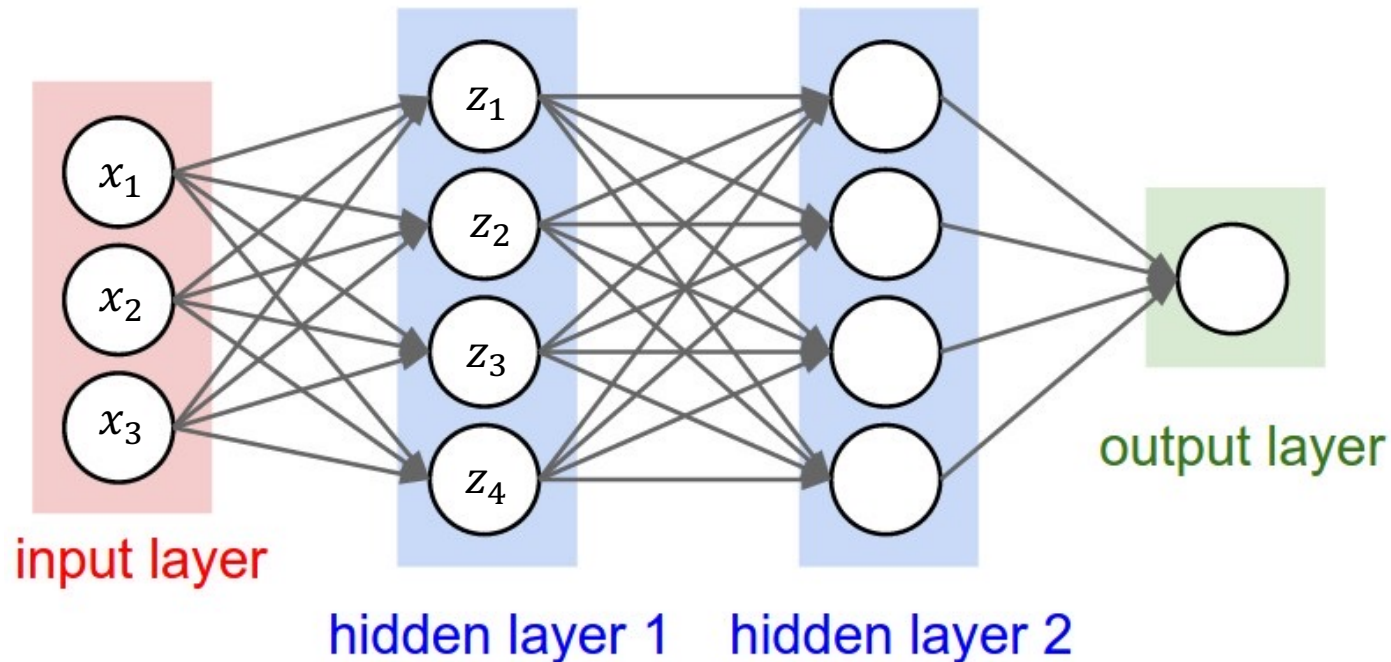
Полносвязный слой (fully connected, FC)

- На входе n чисел, на выходе m чисел
- x_1, \dots, x_n — ВХОДЫ
- z_1, \dots, z_m — ВЫХОДЫ
- Каждый выход — линейная модель над входами

$$z_j = \sum_{i=1}^n w_{ji} x_i + b_j$$

Полносвязный слой (fully connected, FC)

$$z_j = \sum_{i=1}^n w_{ji} x_i + b_j$$



Полносвязный слой (fully connected, FC)

$$z_j = \sum_{i=1}^n w_{ji} x_i + b_j$$

- t линейных моделей, в каждой $(n + 1)$ параметров
- Всего примерно tn параметров в полносвязном слое

Полносвязный слой (fully connected, FC)

$$z_j = \sum_{i=1}^n w_{ji} x_i + b_j$$

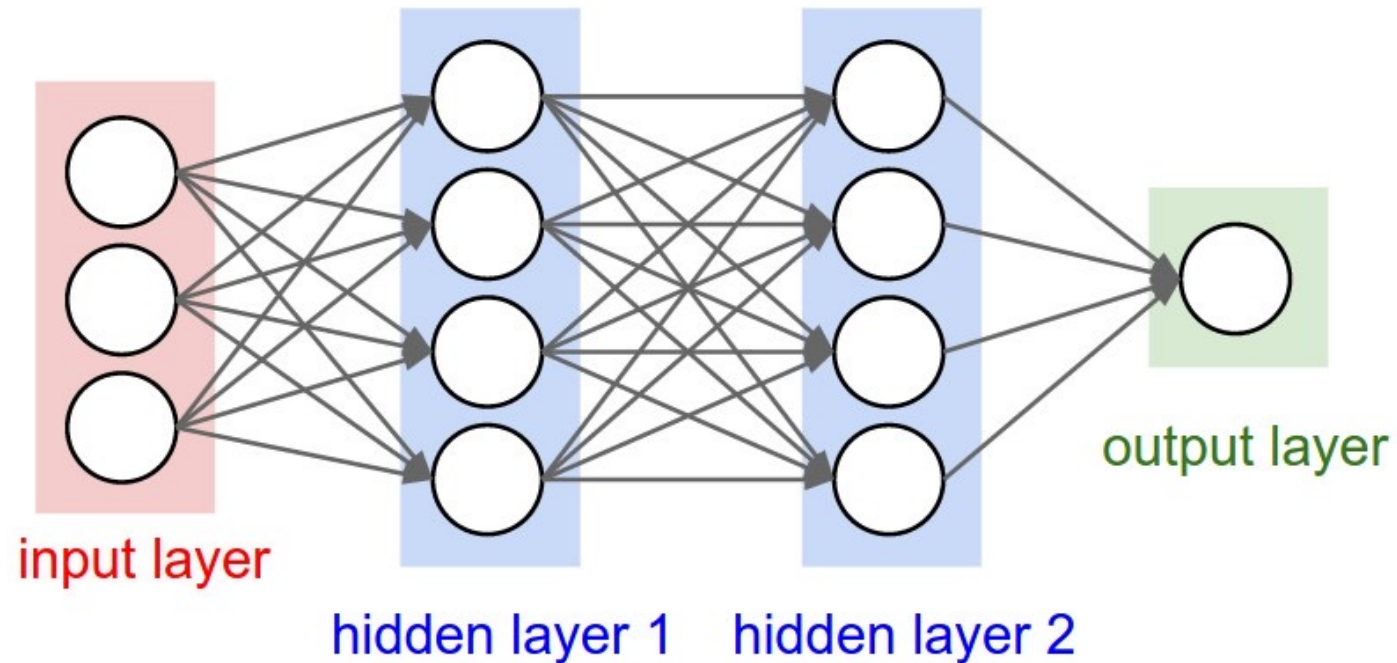
- t линейных моделей, в каждой $(n + 1)$ параметров
- Всего примерно tn параметров в полносвязном слое
- Это очень много: если у нас 1.000.000 входных признаков и 1000 выходов, то это 1.000.000.000 параметров
- Надо много данных для обучения

Важный вопрос в DL

Как объединить слои в мощную модель?

Нелинейность

- Рассмотрим два полносвязных слоя



Нелинейность

- Рассмотрим два полносвязных слоя

$$\begin{aligned} S_k &= \sum_{j=1}^m v_{kj} z_j + c_k = \sum_{j=1}^m v_{kj} \sum_{i=1}^n w_{ji} x_i + \sum_{j=1}^m v_{kj} b_j + c_k = \\ &= \sum_{j=1}^m \left(\sum_{i=1}^n v_{kj} w_{ji} x_i + v_{kj} b_j + \frac{1}{m} c_k \right) \end{aligned}$$

- То есть это ничем не лучше одного полносвязного слоя

Нелинейность

- Нужно добавлять нелинейную функцию после полносвязного слоя

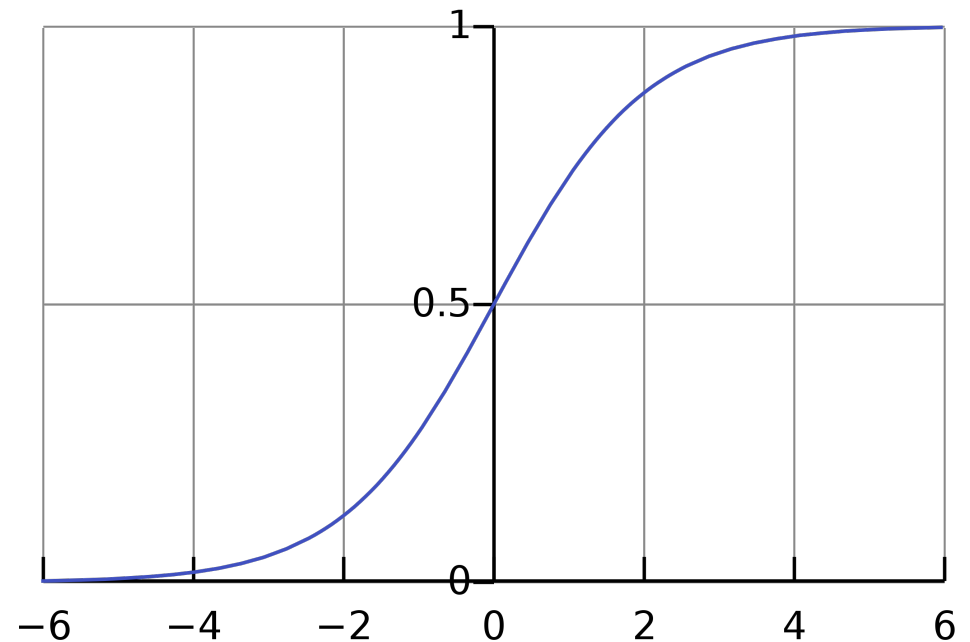
$$z_j = f \left(\sum_{i=1}^n w_{ji} x_i + b_j \right)$$

Нелинейность

$$z_j = f \left(\sum_{i=1}^n w_{ji} x_i + b_j \right)$$

Вариант 1: $f(x) = \frac{1}{1 + \exp(-x)}$

(сигмоида)

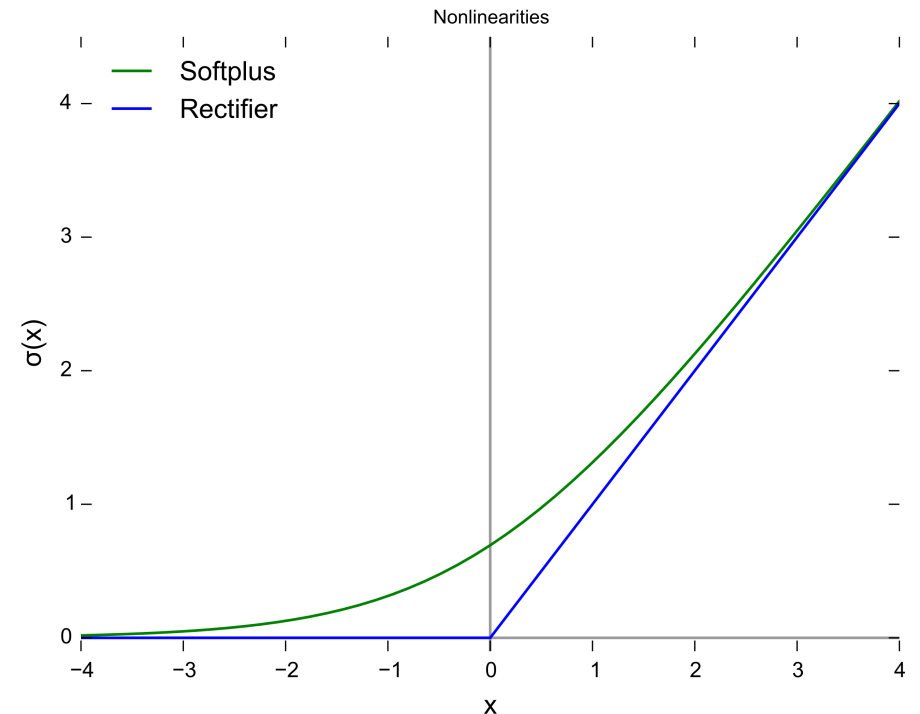


Нелинейность

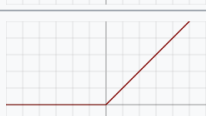

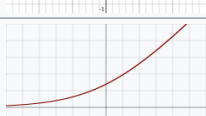
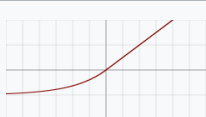
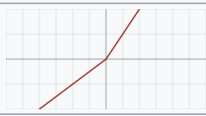
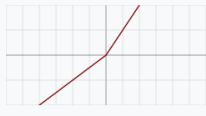
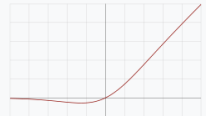
$$z_j = f \left(\sum_{i=1}^n w_{ji} x_i + b_j \right)$$

Вариант 2: $f(x) = \max(0, x)$

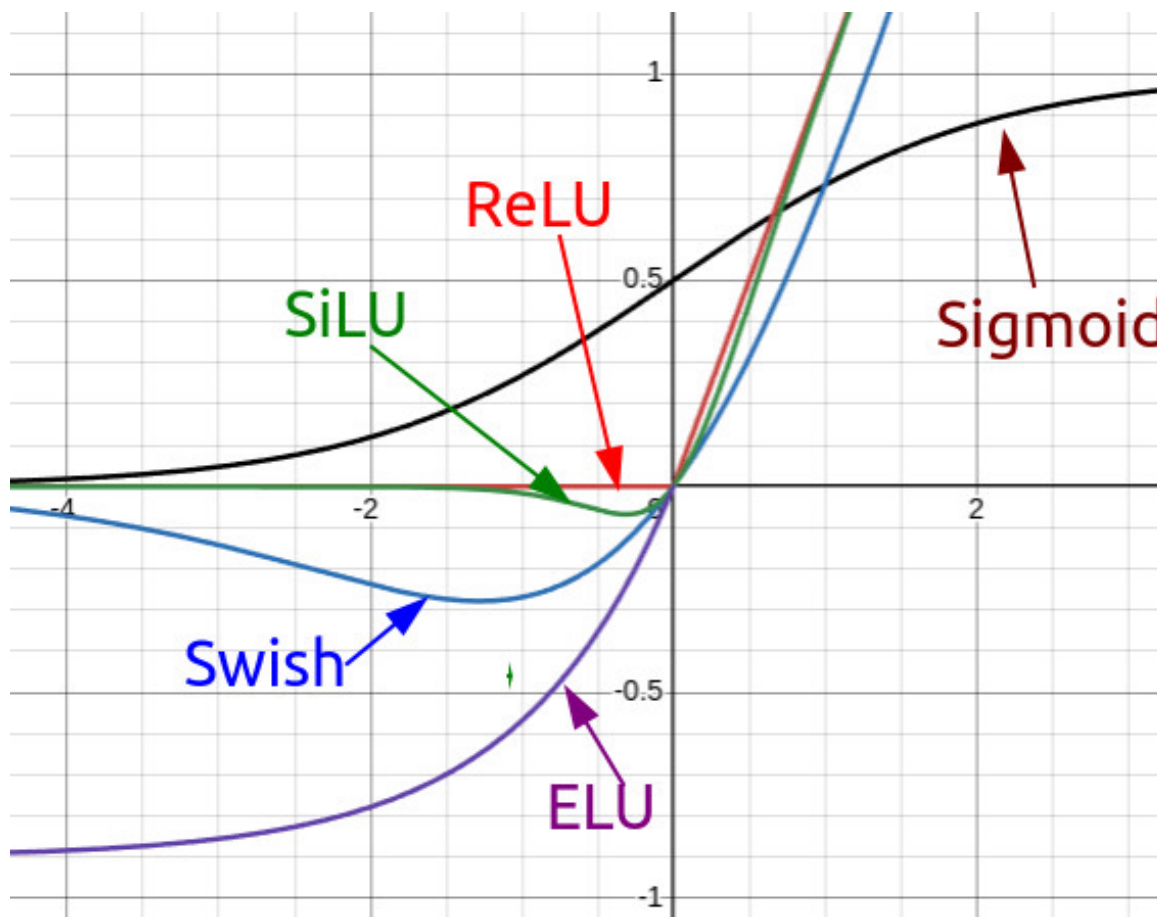
(ReLU, REctified Linear Unit)



Нелинейность

Rectified linear unit (ReLU) ^[9]		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ $= \max\{0, x\} = x \mathbf{1}_{x>0}$
Gaussian Error Linear Unit (GELU) ^[4]		$\frac{1}{2}x \left(1 + \operatorname{erf} \left(\frac{x}{\sqrt{2}} \right) \right)$ $= x\Phi(x)$
Softplus ^[10]		$\ln(1 + e^x)$
Exponential linear unit (ELU) ^[11]		$\begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ <p>with parameter α</p>
Scaled exponential linear unit (SELU) ^[12]		$\lambda \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ <p>with parameters $\lambda = 1.0507$ and $\alpha = 1.67326$</p>
Leaky rectified linear unit (Leaky ReLU) ^[13]		$\begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$
Parameteric rectified linear unit (PReLU) ^[14]		$\begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ <p>with parameter α</p>
Sigmoid linear unit (SiLU, ^[4] Sigmoid shrinkage, ^[15] SiL, ^[16] or Swish-1 ^[17])		$\frac{x}{1 + e^{-x}}$

Нелинейность



Нелинейность

$$\text{ReGLU}(x, W, V, b, c) = \max(0, xW + b) \otimes (xV + c)$$

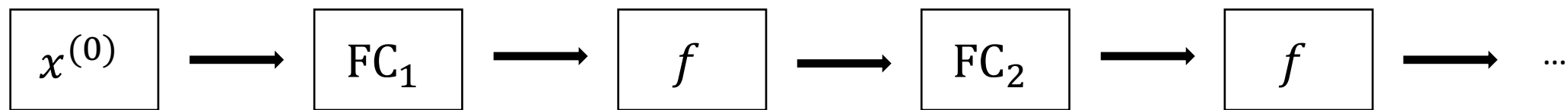
$$\text{GEGLU}(x, W, V, b, c) = \text{GELU}(xW + b) \otimes (xV + c)$$

$$\text{SwiGLU}(x, W, V, b, c, \beta) = \text{Swish}_{\beta}(xW + b) \otimes (xV + c)$$

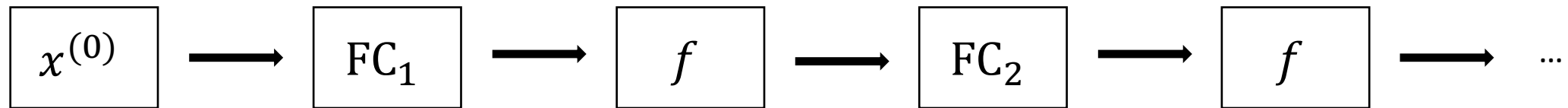
4 Conclusions

We have extended the GLU family of layers and proposed their use in Transformer. In a transfer-learning setup, the new variants seem to produce better perplexities for the de-noising objective used in pre-training, as well as better results on many downstream language-understanding tasks. These architectures are simple to implement, and have no apparent computational drawbacks. We offer no explanation as to why these architectures seem to work; we attribute their success, as all else, to divine benevolence.

Типичная полносвязная сеть



Типичная полносвязная сеть



- На входе признаки
- В последнем слое выходов столько, сколько целевых переменных мы предсказываем

Теорема Цыбенко

Вольное изложение:

- Пусть $g(x)$ — непрерывная функция
- Тогда можно построить двуслойную нейронную сеть, приближающую $g(x)$ с любой заранее заданной точностью

То есть двуслойные нейронные сети **ОЧЕНЬ** мощные!

Теорема Цыбенко

Вольное изложение:

- Пусть $g(x)$ — непрерывная функция
- Тогда можно построить двуслойную нейронную сеть, приближающую $g(x)$ с любой заранее заданной точностью

То есть двуслойные нейронные сети **ОЧЕНЬ** мощные!

Но очень много параметров и очень сложно обучать

Резюме

- Идея глубинного обучения — совмещение большого количества дифференцируемых слоёв
- Слои извлекают сложные признаки из данных
- Полносвязные слои — самый простой (и при этом мощный) вариант
- Важны нелинейности