

Введение в глубинное обучение

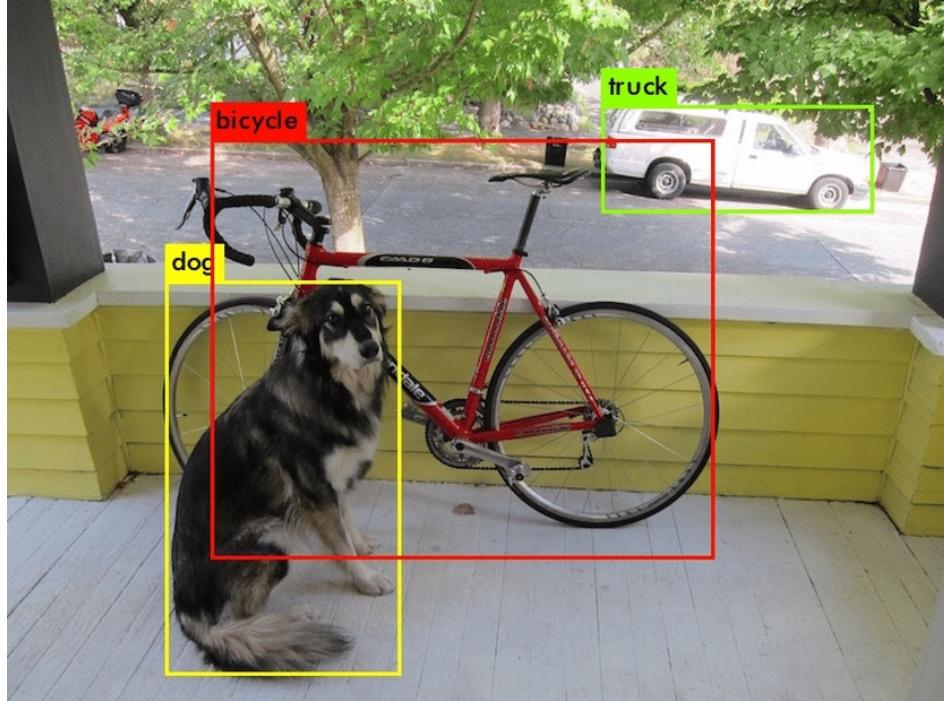
Евгений Соколов

esokolov@hse.ru

Сириус, декабрь 2025

Пример задачи: детекция объектов

- Данна фотография
- Найти человека на фотографии
- Гарантируется, что человек точно есть и точно один



Обозначения

- x — объект — что анализируем
 - Фотография
 - Массив $n * m * 3$
- y — ответ, целевая переменная — что на выходе
 - Прямоугольник, содержащий человека
 - Четыре числа: координаты левого верхнего угла прямоугольника (a, b), ширина и высота (c, d)
 - $y = (a, b, c, d)$

Обучающая выборка

- Нам нужны примеры, из которых будем выводить правила
- $X = (x_i, y_i)_{i=1}^{\ell}$ — обучающая выборка
- ℓ — размер выборки
- Много фотографий с прямоугольниками

Признаки

- Объекты — абстрактные сущности
- Компьютеры работают только с числами
- Признаки, факторы — характеристики объектов
- d — количество признаков
- $x = (x_1, \dots, x_d)$ — признаковое описание

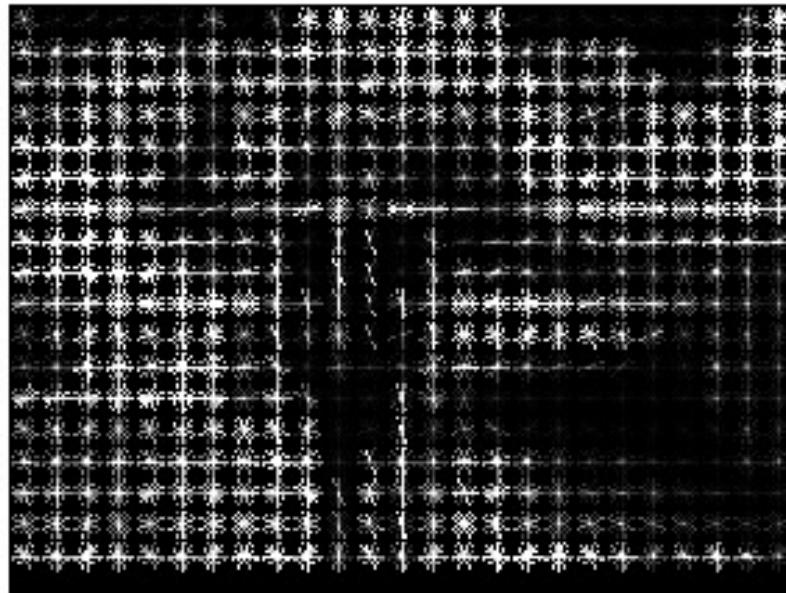
Признаки

Признаки

Input image



Histogram of Oriented Gradients



Алгоритм

- $a(x)$ — алгоритм, модель — функция, оценивающая ответ для любого объекта
- Линейная модель: $a(x) = w_0 + w_1x_1 + \dots + w_dx_d$

ФУНКЦИЯ ПОТЕРЬ

- Как понять, что наш алгоритм работает?
- Сравнить его прогнозы с правильными ответами на обучающей выборке!

$$L(y, z) = L\left((a_y, b_y, c_y, d_y), (a_z, b_z, c_z, d_z)\right) = \\ (a_y - a_z)^2 + (b_y - b_z)^2 + (c_y - c_z)^2 + (d_y - d_z)^2$$

Функционал ошибки

- Функционал ошибки — мера качества работы алгоритма на выборке
- Обычно вычисляется как среднее значение функции потерь:

$$\frac{1}{\ell} \sum_{i=1}^{\ell} L(y_i, a(x_i))$$

- Чем меньше, тем лучше

Функционал ошибки

- Должен соответствовать бизнес-требованиям
- Одна из самых важных составляющих анализа данных

Обучение алгоритма

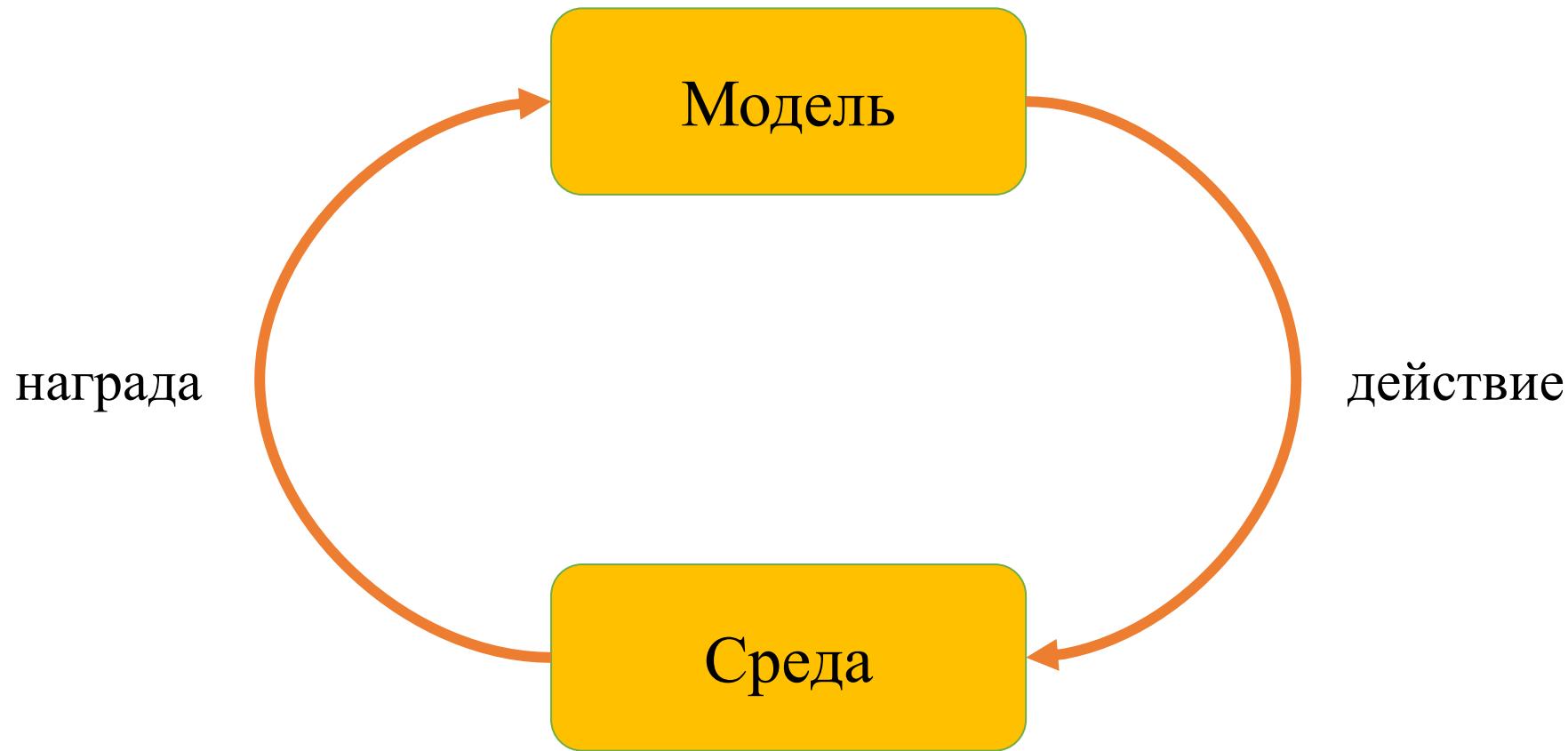
- Есть обучающая выборка и функционал ошибки
- Семейство алгоритмов \mathcal{A}
 - Из чего выбираем алгоритм
 - Пример: все линейные модели
 - $\mathcal{A} = \{w_0 + w_1x_1 + \dots + w_dx_d \mid w_0, w_1, \dots, w_d \in \mathbb{R}\}$
- Обучение: поиск оптимального алгоритма с точки зрения функционала ошибки

$$a(x) = \arg \min_{a \in \mathcal{A}} Q(a, X)$$

Машинное обучение

- Не все задачи имеют такую формулировку!
- Обучение без учителя
- Обучение с подкреплением
- И т.д.

Обучение с подкреплением



Резюме

- Машинное обучение — построение алгоритмов на основе примеров
- Постановка задачи: объект и ответ
- Обучающая выборка
- Модель/алгоритм
- Функция потерь

Обобщающая способность

Обобщающая способность

Как готовиться к экзамену?

Заучить все примеры с
занятий

Разобраться в предмете и
усвоить алгоритмы решения
задач

Обобщающая способность

Как готовиться к экзамену?

Заучить все примеры с
занятий

Переобучение (overfitting)

Разобраться в предмете и
усвоить алгоритмы решения
задач

Обобщение (generalization)

Обобщающая способность

Как готовиться к экзамену?

Заучить все примеры с занятий

Переобучение (overfitting)

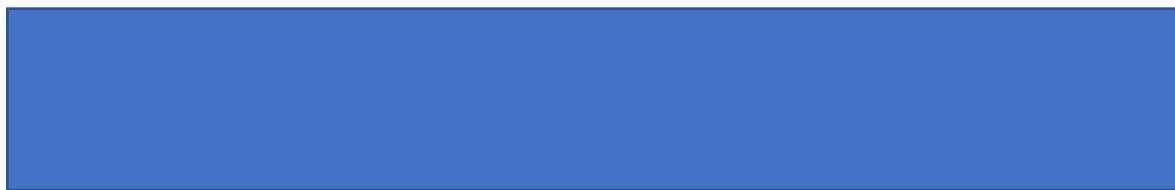
Хорошее качество на обучении
Низкое качество на новых данных

Разобраться в предмете и усвоить алгоритмы решения задач

Обобщение (generalization)

Хорошее качество на обучении
Хорошее качество на новых данных

Отложенная выборка



Обучение



Тест

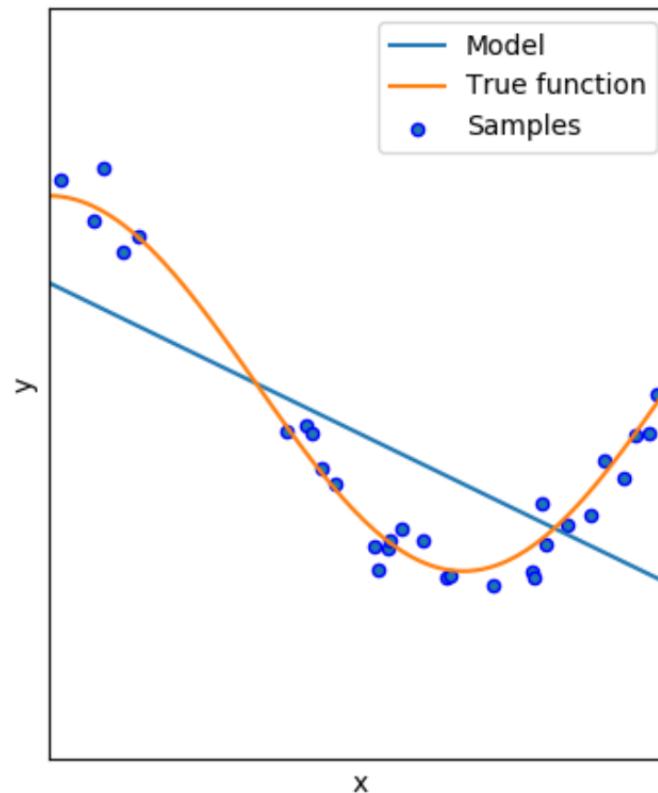
Отложенная выборка



- Слишком большое обучение — тестовая выборка нерепрезентативна
- Слишком большой тест — модель не сможет обучиться
- Обычно: 70/30, 80/20

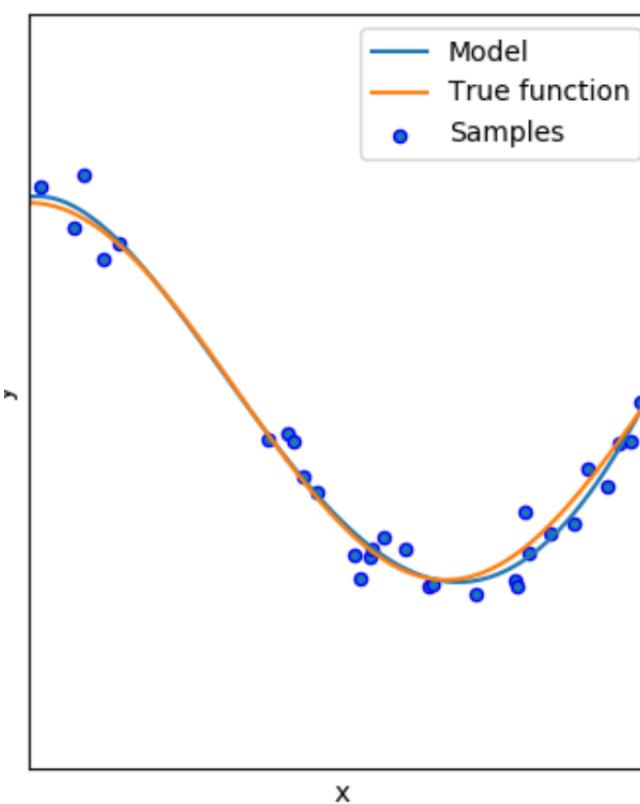
Нелинейная задача

$$a(x) = w_0 + w_1 x$$



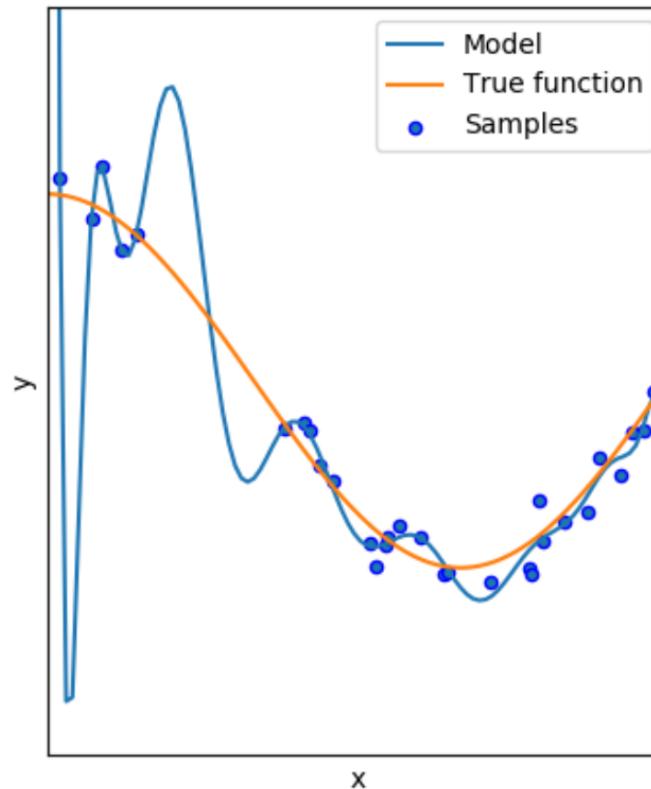
Нелинейная задача

$$a(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$$



Нелинейная задача

$$a(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + \cdots + w_{15} x^{15}$$



Симптом переобучения

$$a(x) = 0.5 + 13458922x - 43983740x^2 + \dots$$

- Большие коэффициенты — симптом переобучения
- Эмпирическое наблюдение

Градиентные методы

Градиент

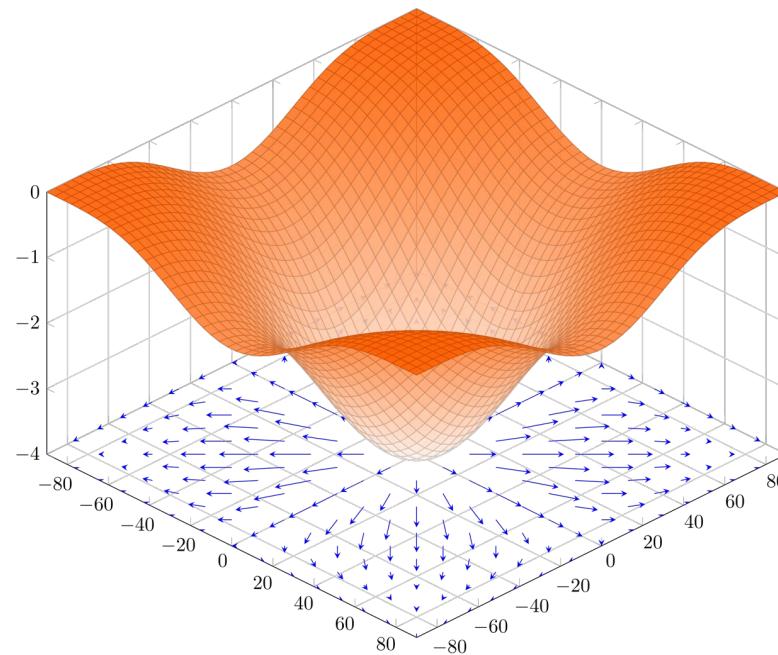
- Градиент — вектор частных производных

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_d} \right)$$

- У градиента есть важное свойство!

Важное свойство

- Зафиксируем точку x_0
- В какую сторону функция быстрее всего растёт?



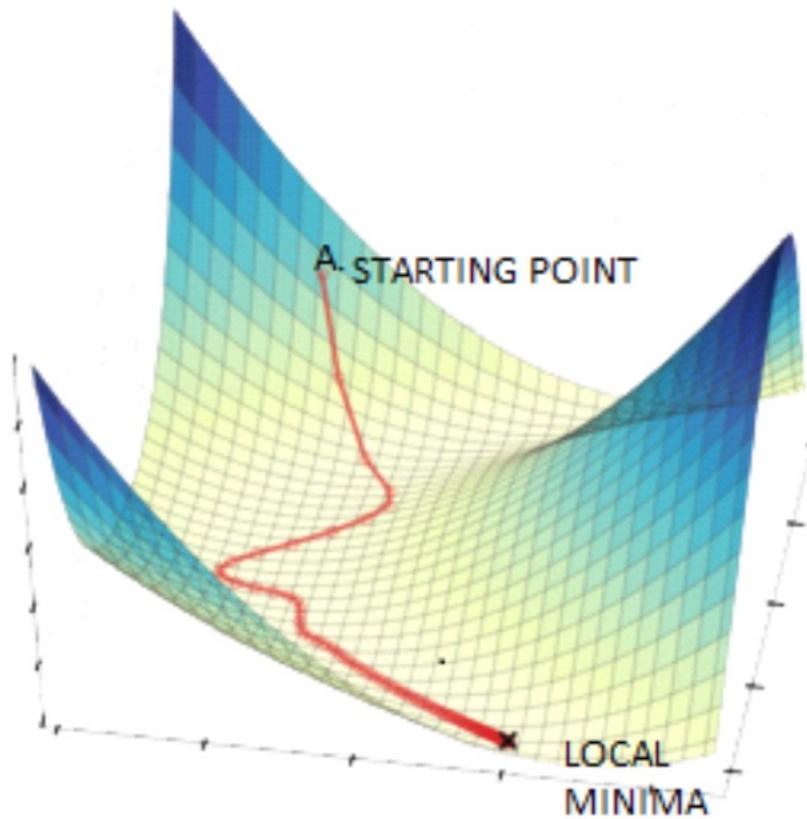
Важное свойство

- Зафиксируем точку x_0
- В какую сторону функция быстрее всего растёт?
- В направлении градиента!
- А быстрее всего убывает в сторону антиградиента

Как это пригодится?



Как это пригодится?



Градиентный спуск

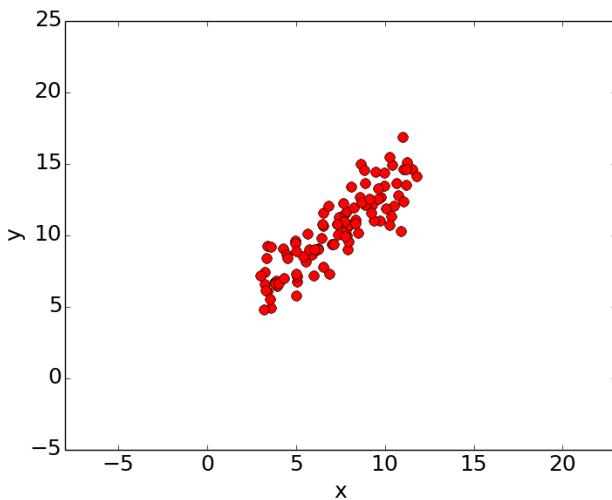
- Стартуем из случайной точки
- Сдвигаемся по антиградиенту
- Повторяем, пока не окажемся в точке минимума

Парная регрессия

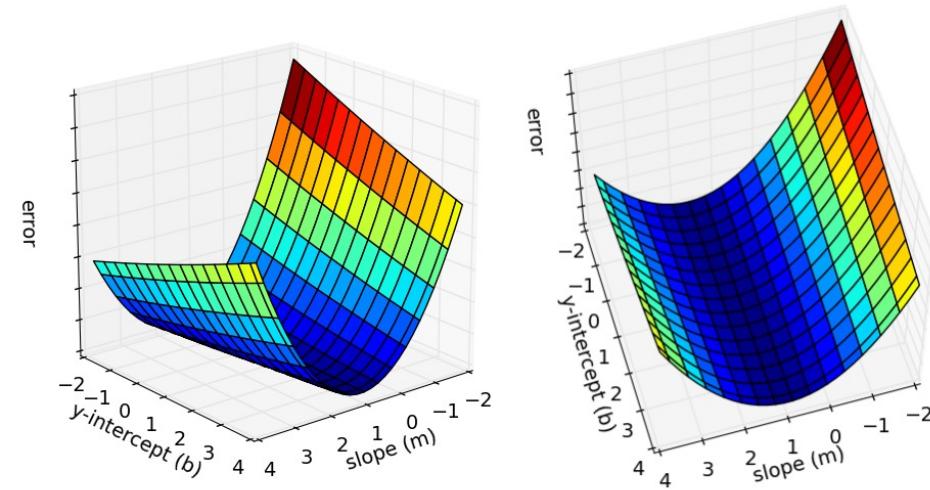
- Простейший случай: один признак
- Модель: $a(x) = w_1 x + w_0$
- Два параметра: w_1 и w_0
- Функционал:

$$Q(w_0, w_1) = \frac{1}{\ell} \sum_{i=1}^{\ell} (w_1 x_i + w_0 - y_i)^2$$

Парная регрессия



Выборка



Функционал ошибки

Начальное приближение

- w^0 — инициализация весов
- Например, из стандартного нормального распределения

Градиентный спуск

- Повторять до сходимости:

$$w^t = w^{t-1} - \eta \nabla Q(w^{t-1})$$

Новая точка

Размер шага

Градиент в
предыдущей
точке

Сходимость

- Останавливаем процесс, если

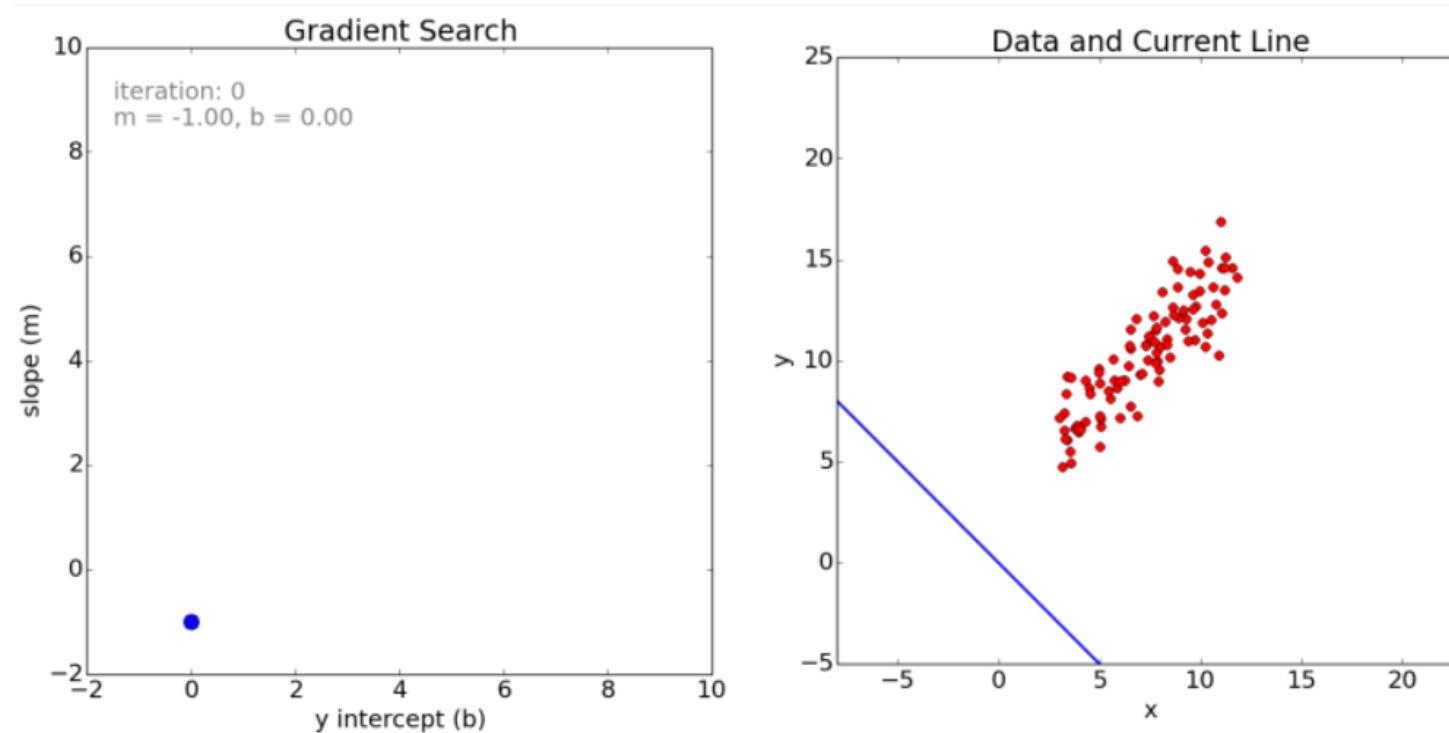
$$\|w^t - w^{t-1}\| < \varepsilon$$

- Другой вариант:

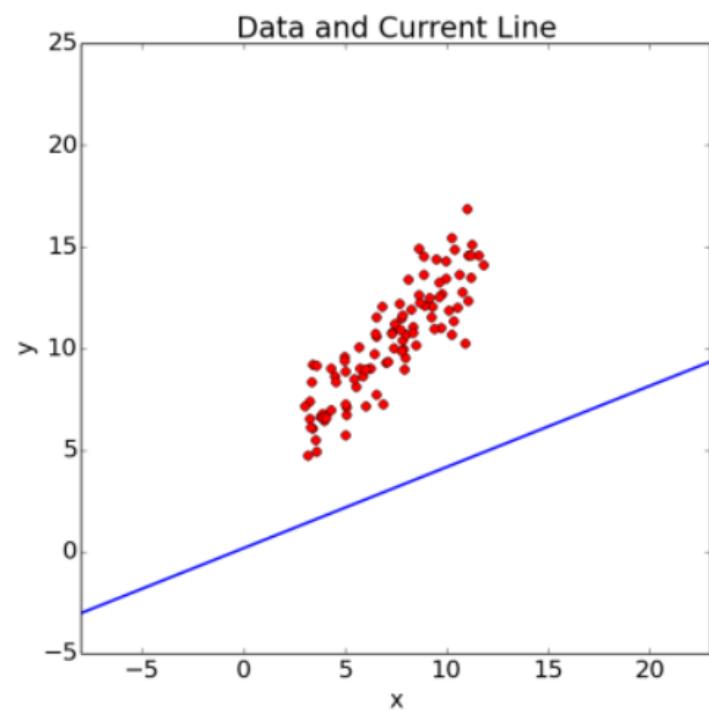
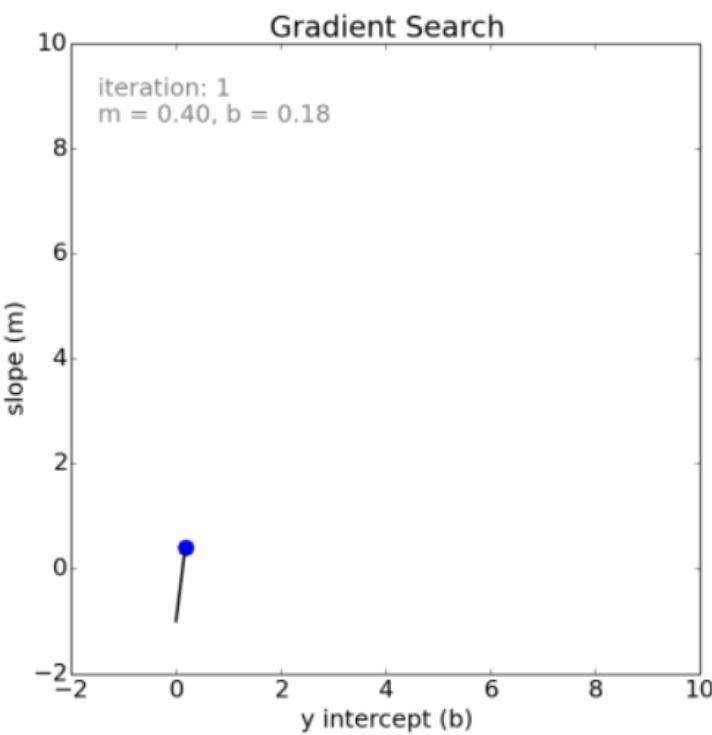
$$\|\nabla Q(w^t)\| < \varepsilon$$

- Или пока ошибка на отложенной выборке уменьшается

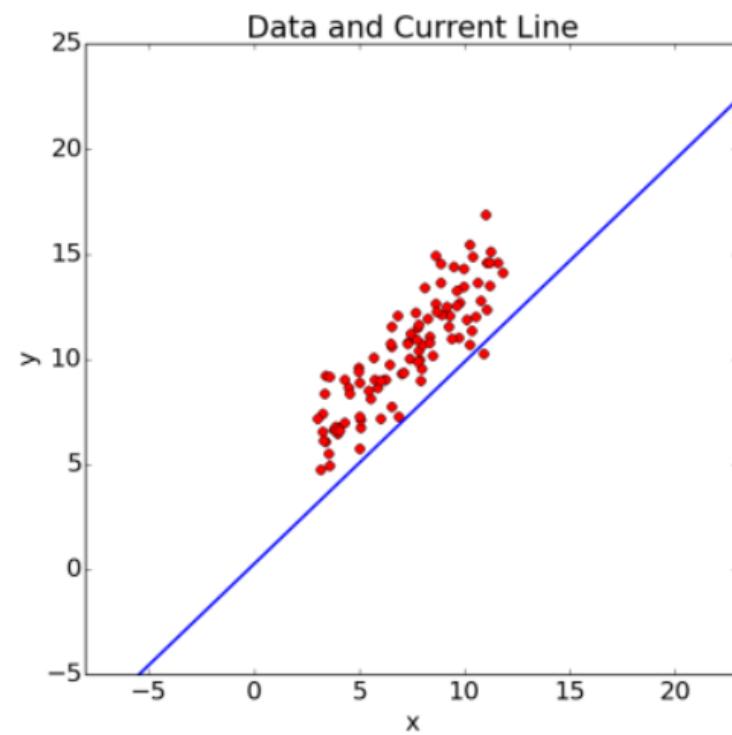
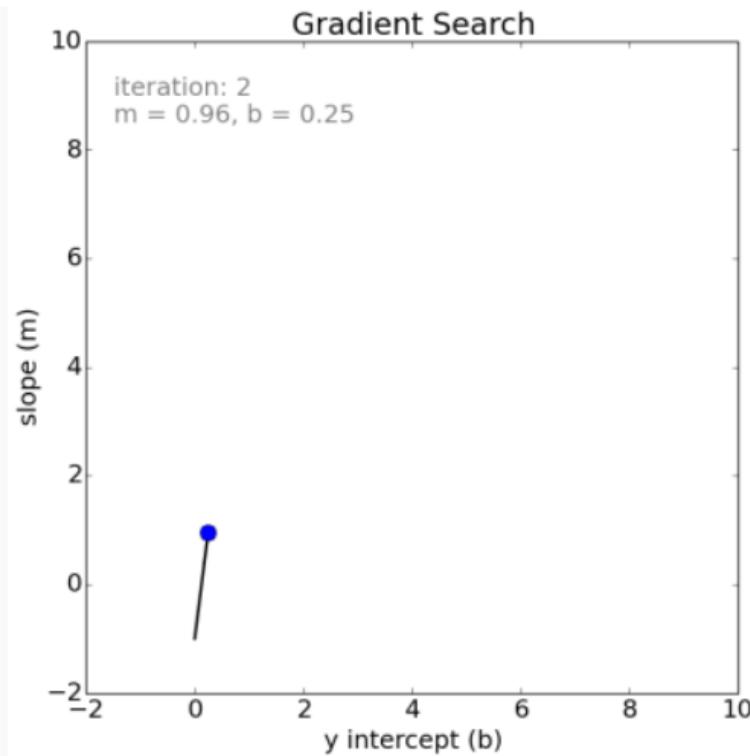
Парная регрессия



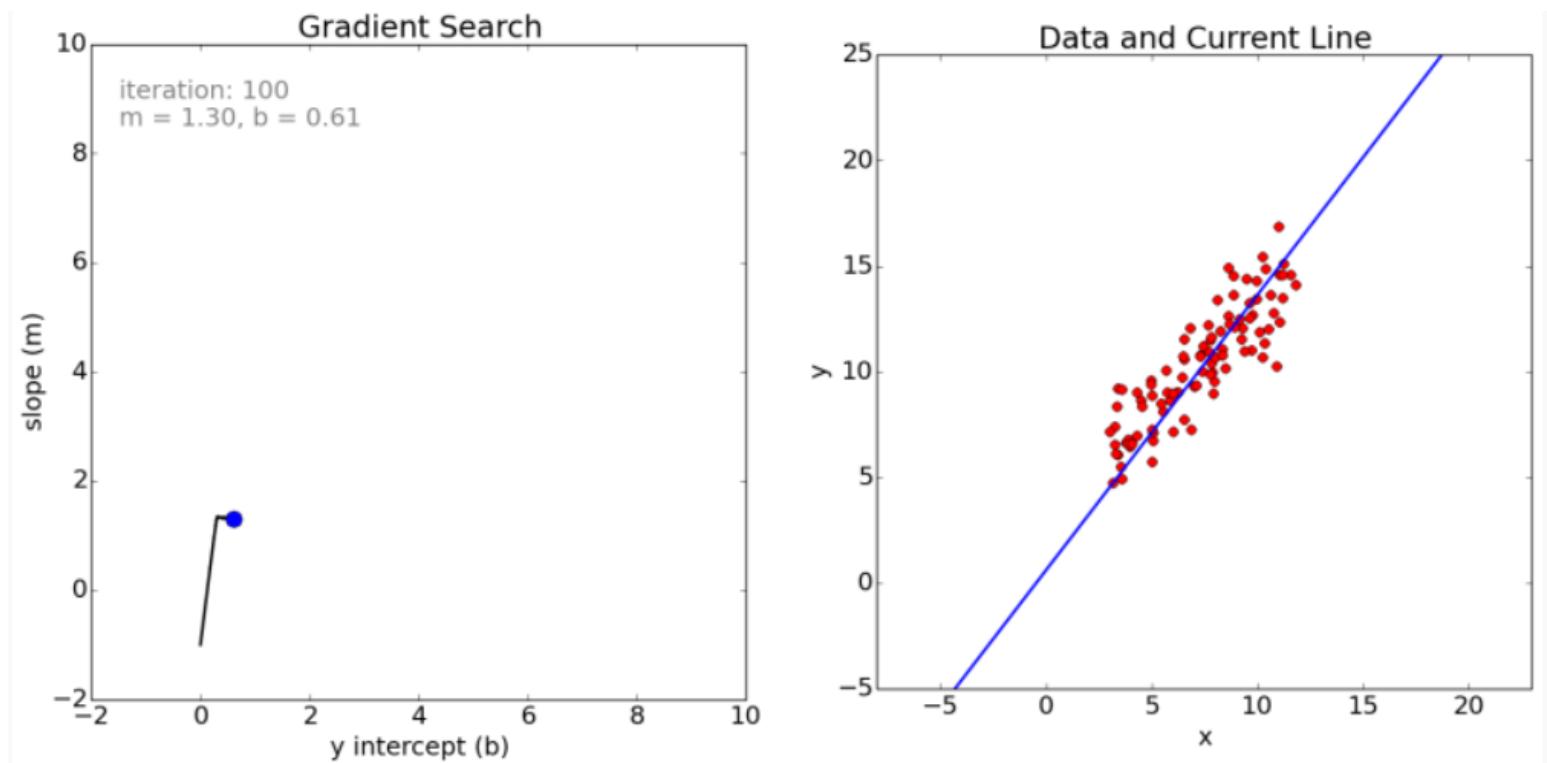
Парная регрессия



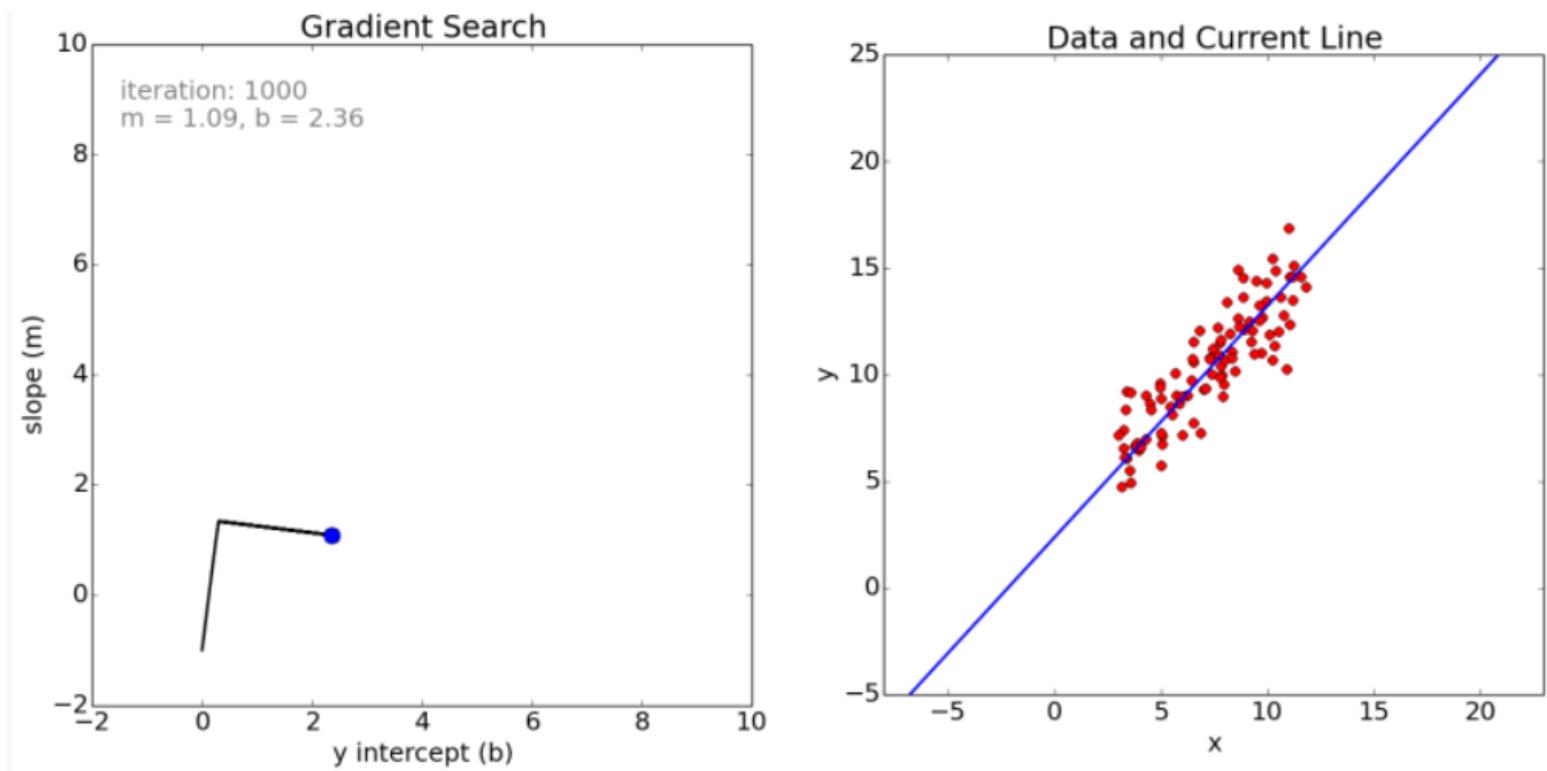
Парная регрессия



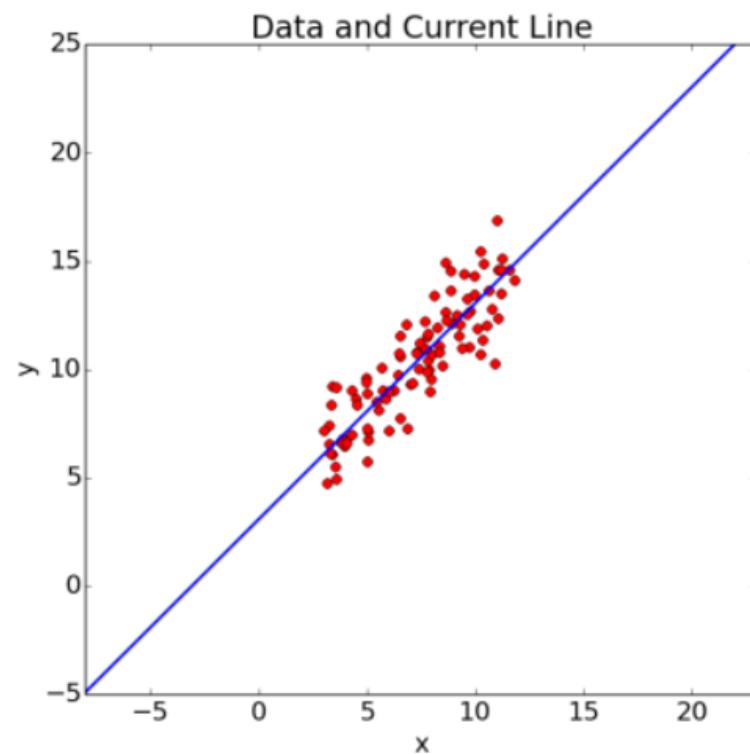
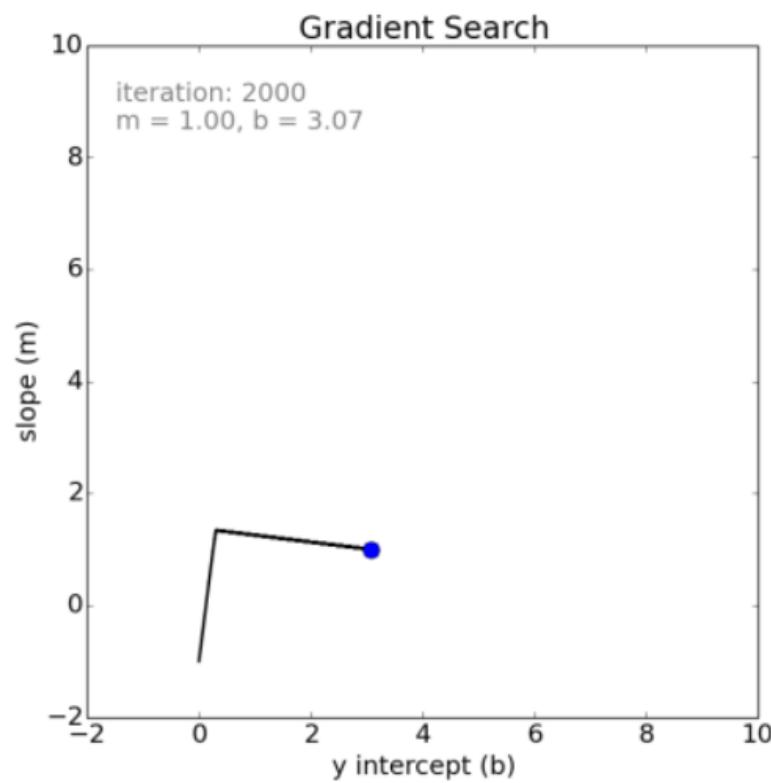
Парная регрессия



Парная регрессия

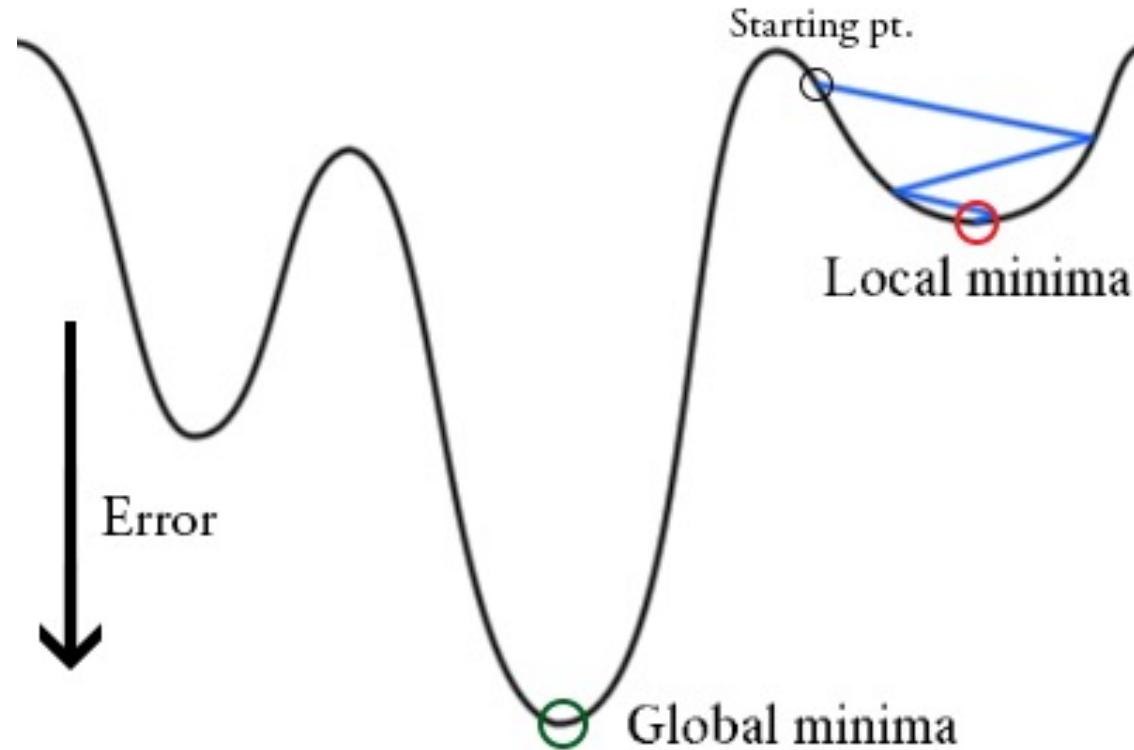


Парная регрессия

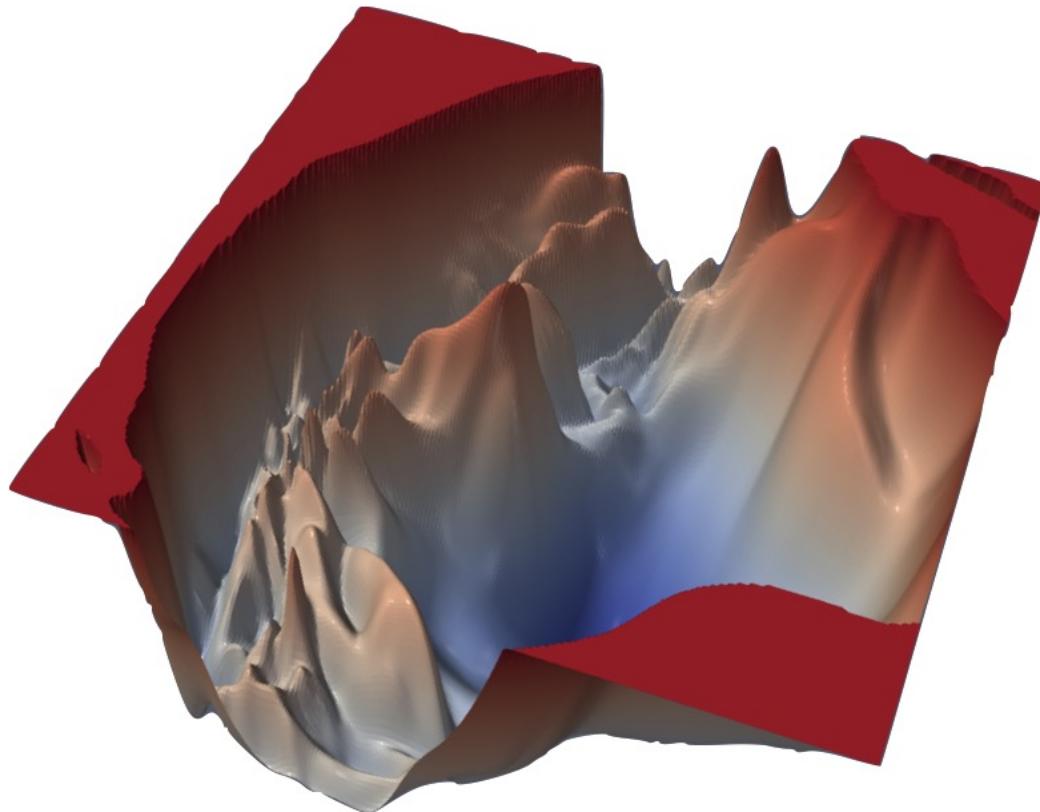


Локальные минимумы

- Градиентный спуск находит только локальные минимумы



Локальные минимумы



Резюме

- Градиент помогает понять, в какую сторону функция меняется быстрее всего
- Движение по антиградиенту позволяет найти локальный минимум

Глубинное обучение



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

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.

Prizes

Evaluation

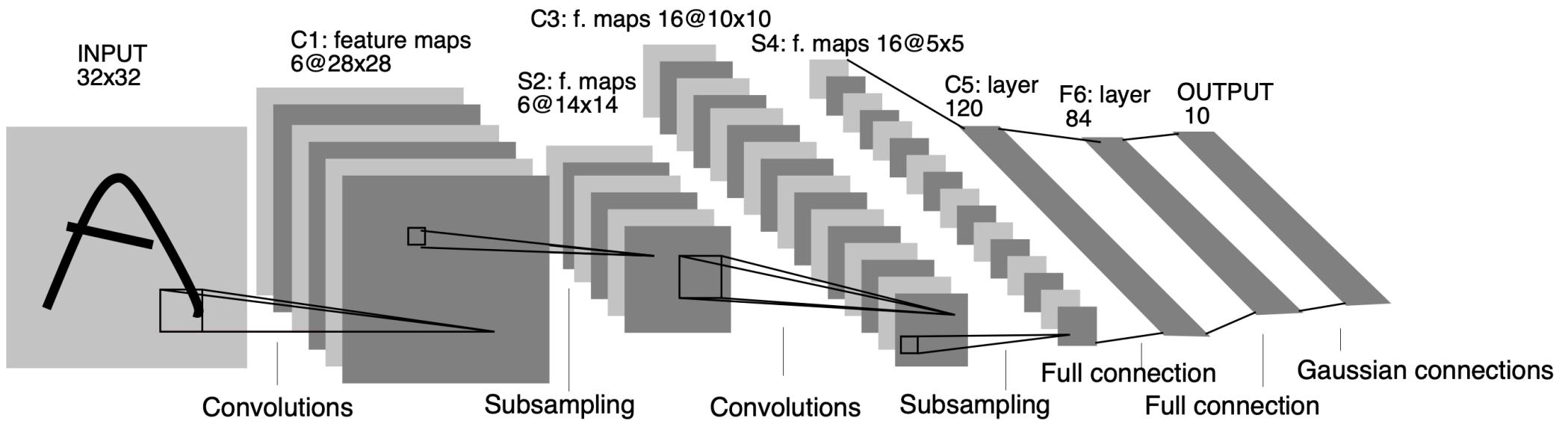
Winners



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

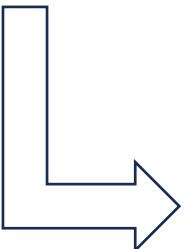
1. Подсчитываем статистику, как часто то или иное слово встречается после данного
2. Генерируем следующее слово из этого распределения

"Manure, almond gelato and frozen pies, you are also had it was in one but it will post office buildings sucks). 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

```
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;
};

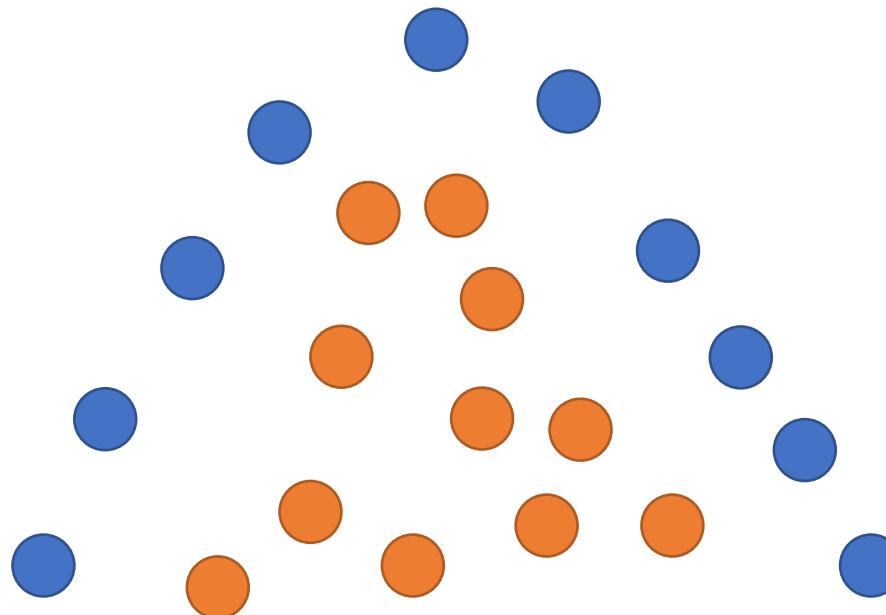
// Color interpolation helper
const interpolateColor = (color1, color2, factor) => {
  const result = color1.slice(1).match(/.{2}/g).map((hex, index) => {
    const value1 = parseInt(hex, 16);
    const value2 = parseInt(color2.slice(1).match(/.{2}/g)[index], 16);
    const value = Math.round(value1 + (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;
  };
  window.addEventListener('resize', handleResize);
  return () => {
    window.removeEventListener('resize', handleResize);
  };
});
```

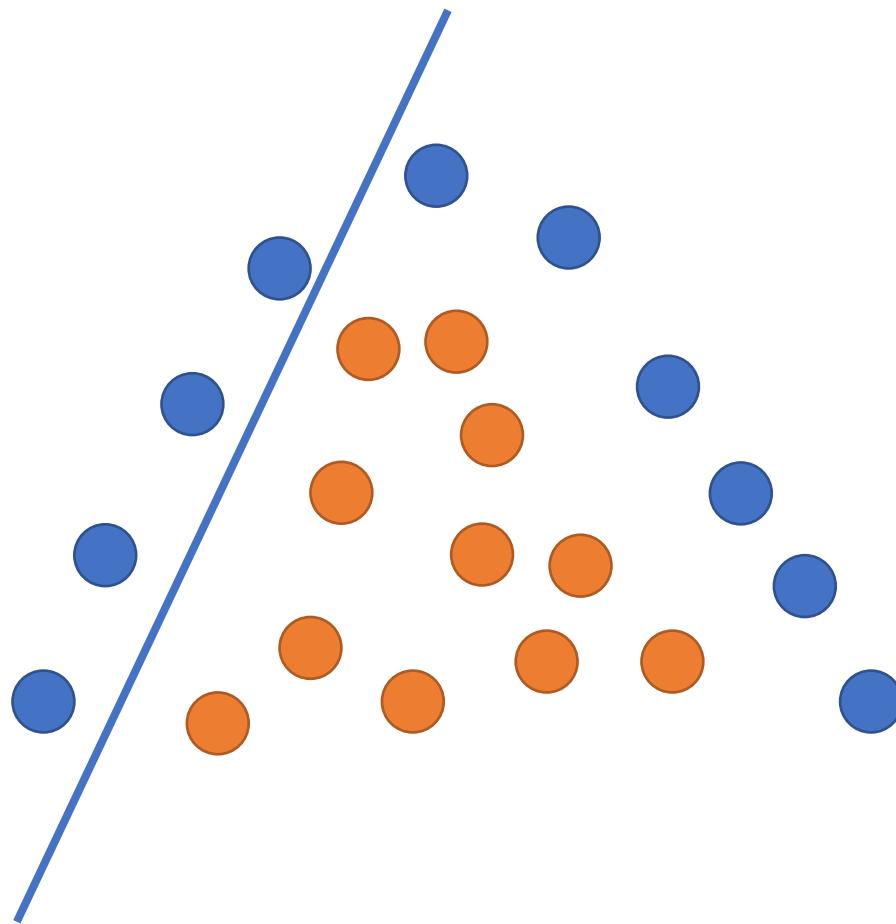
Успехи в глубинном обучении

- Изображения и видео
- Трёхмерное компьютерное зрение
- Тексты
- Звук
- Генерация данных

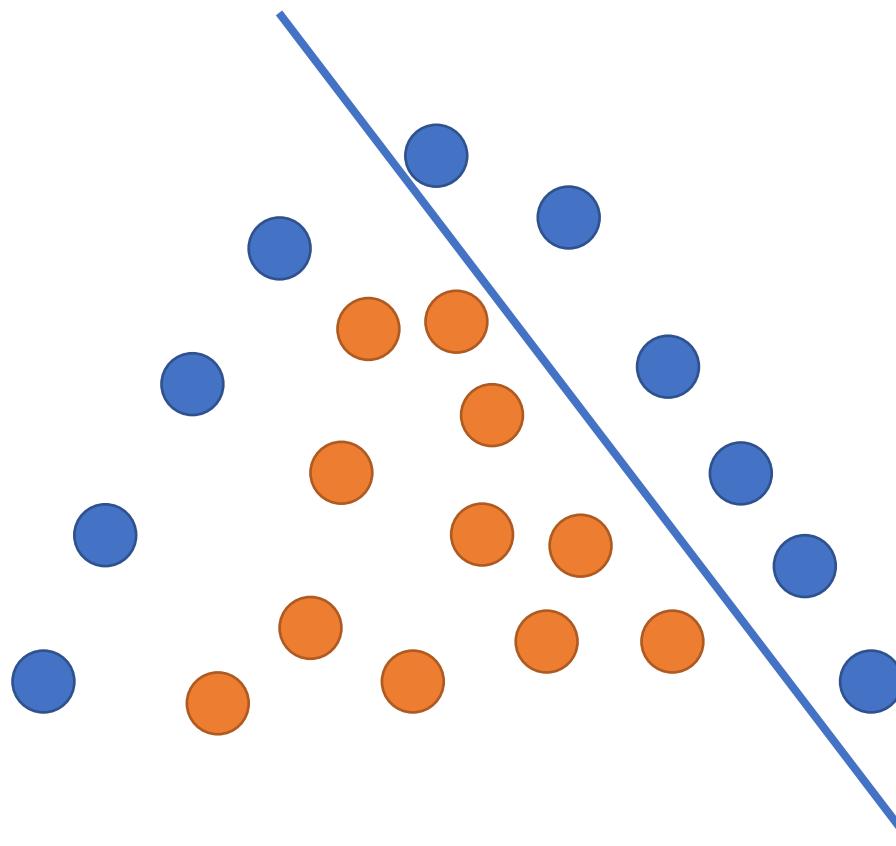
Нелинейные закономерности



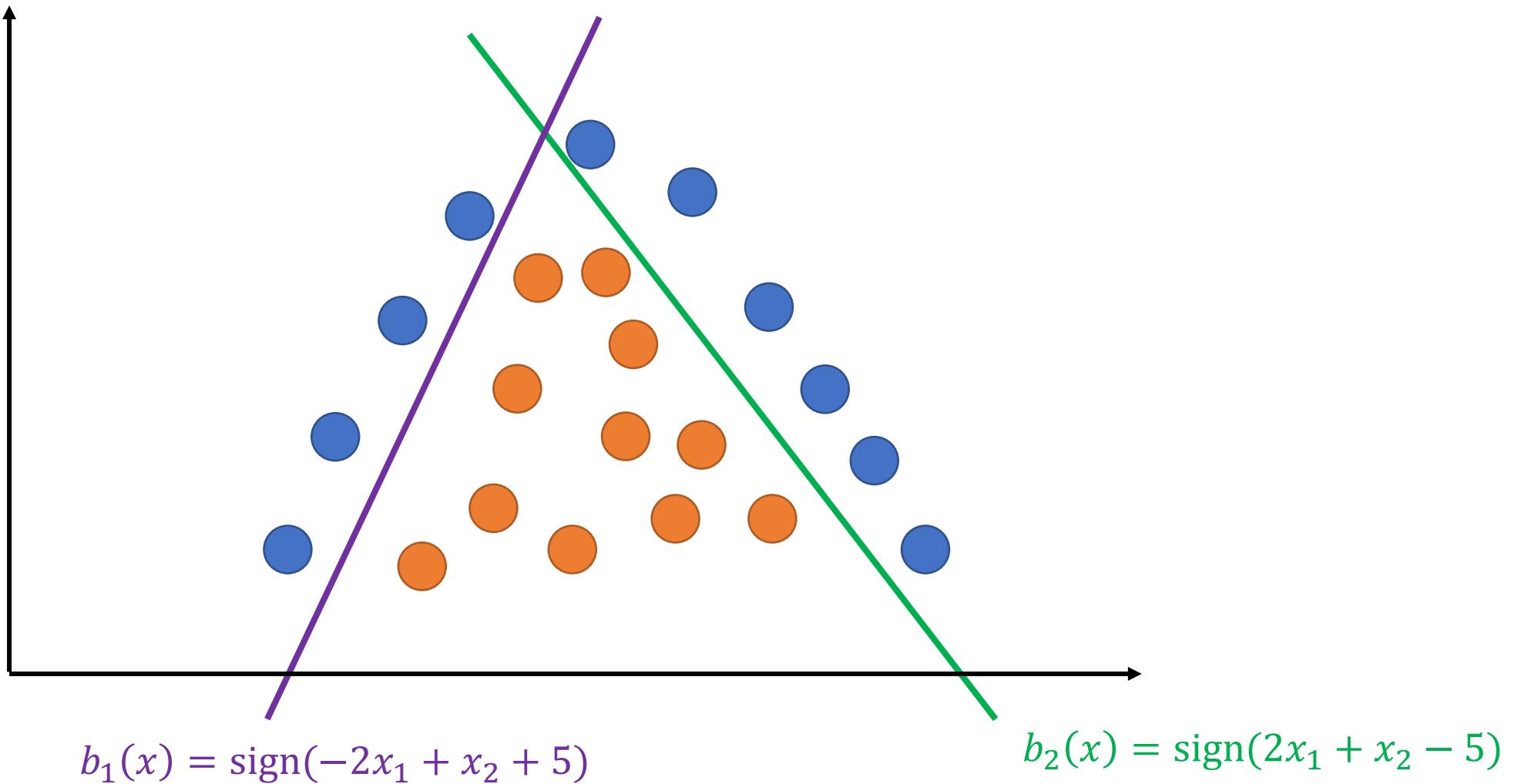
Нелинейные закономерности



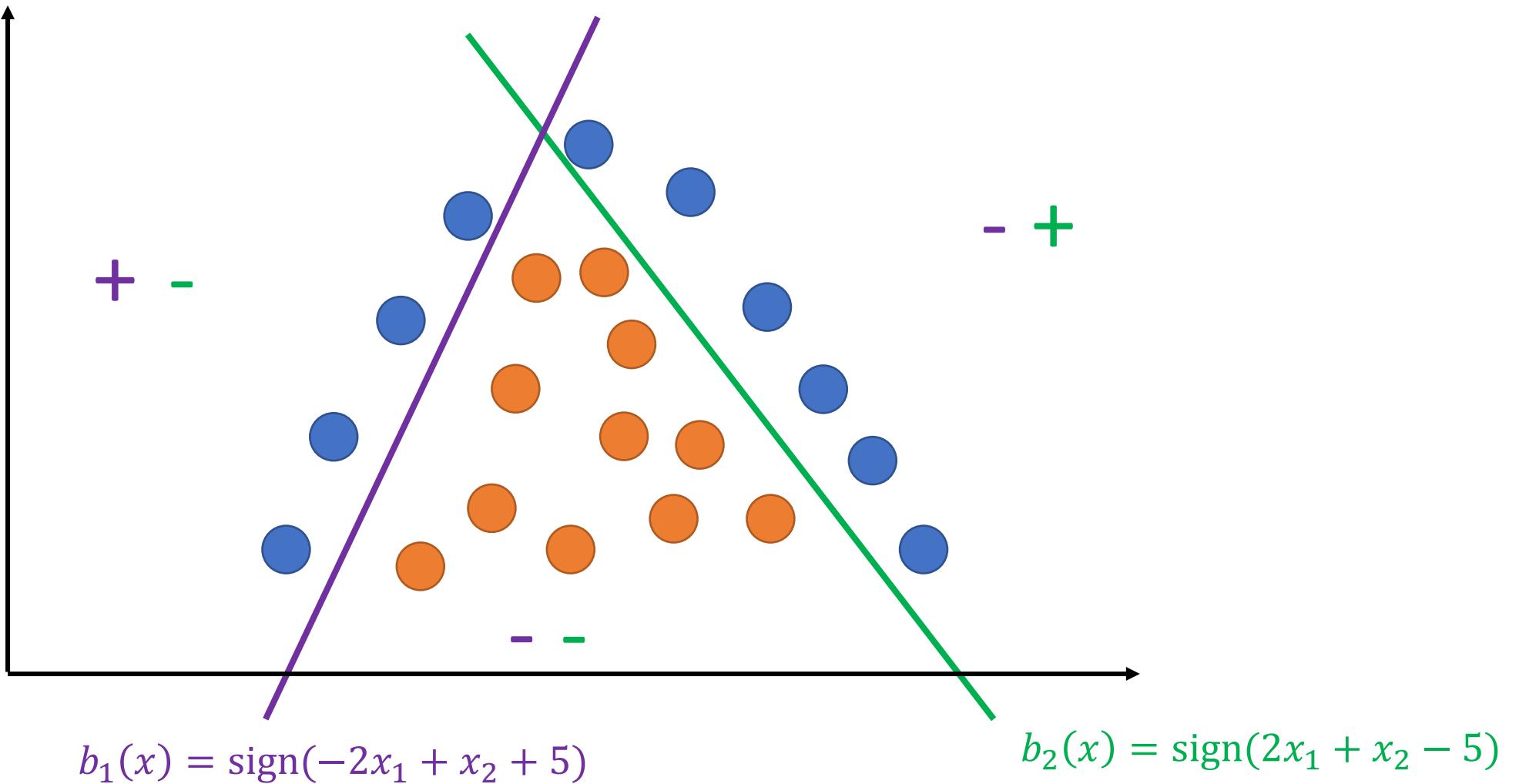
Нелинейные закономерности



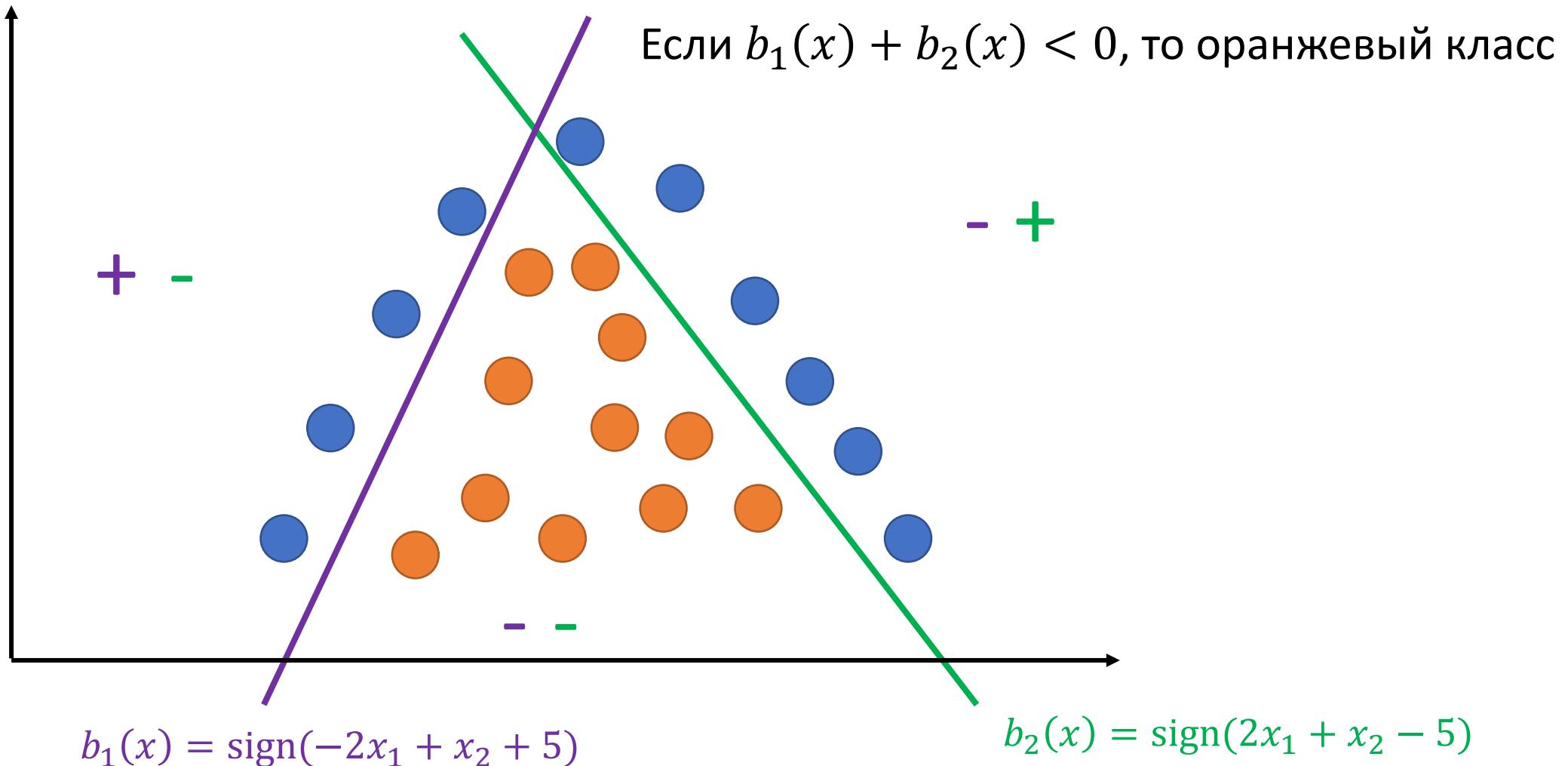
Нелинейные закономерности



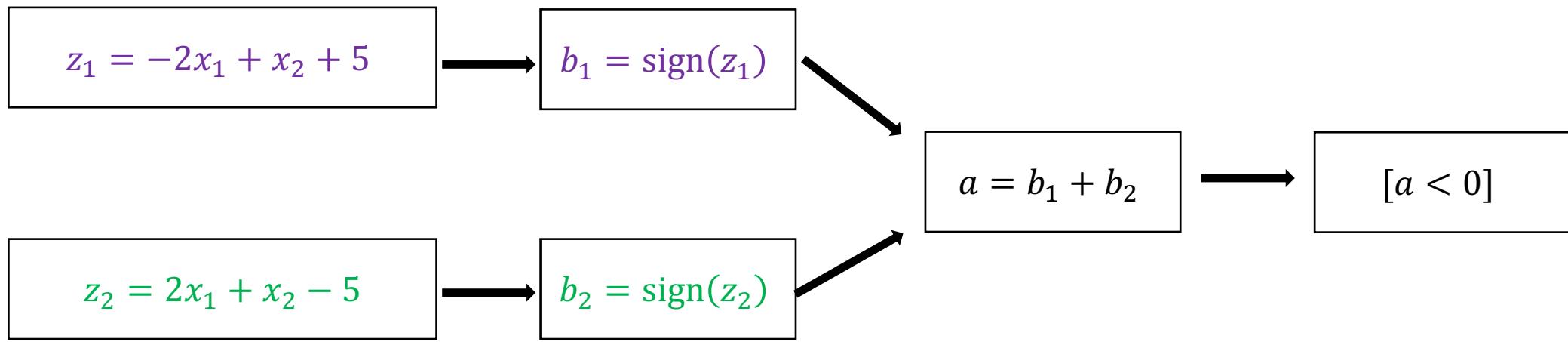
Нелинейные закономерности



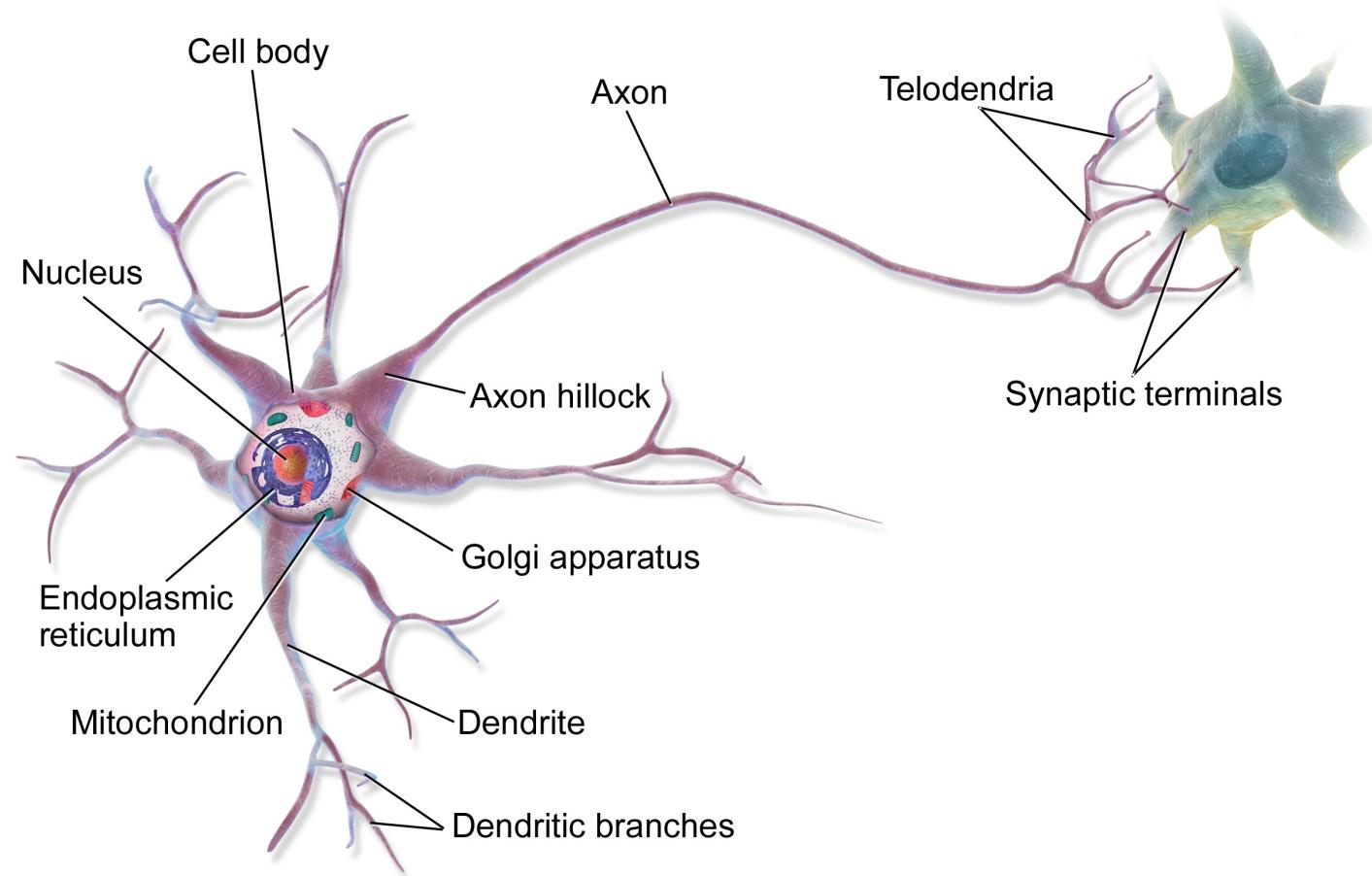
Нелинейные закономерности



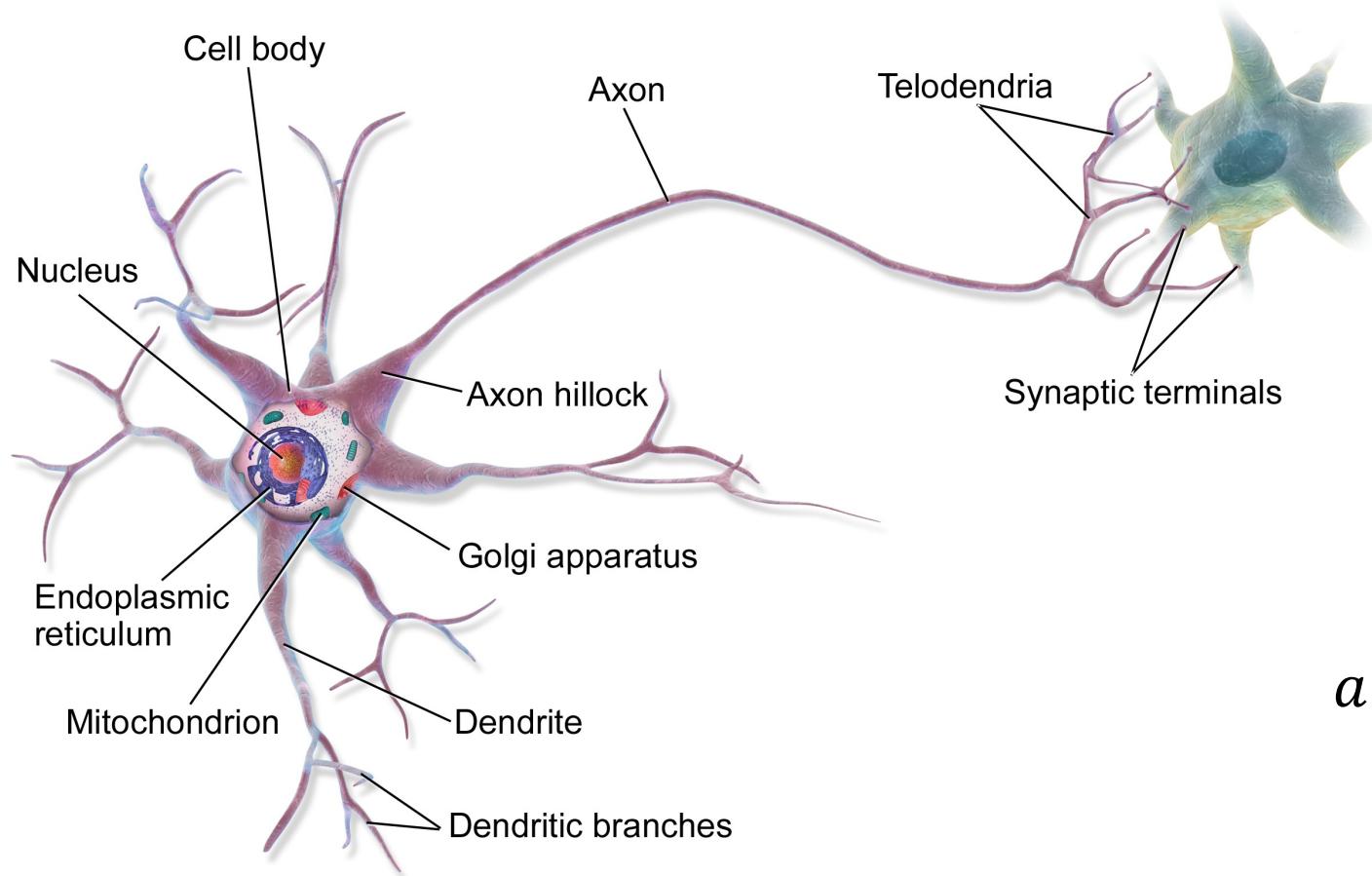
Нелинейные закономерности



Нейрон

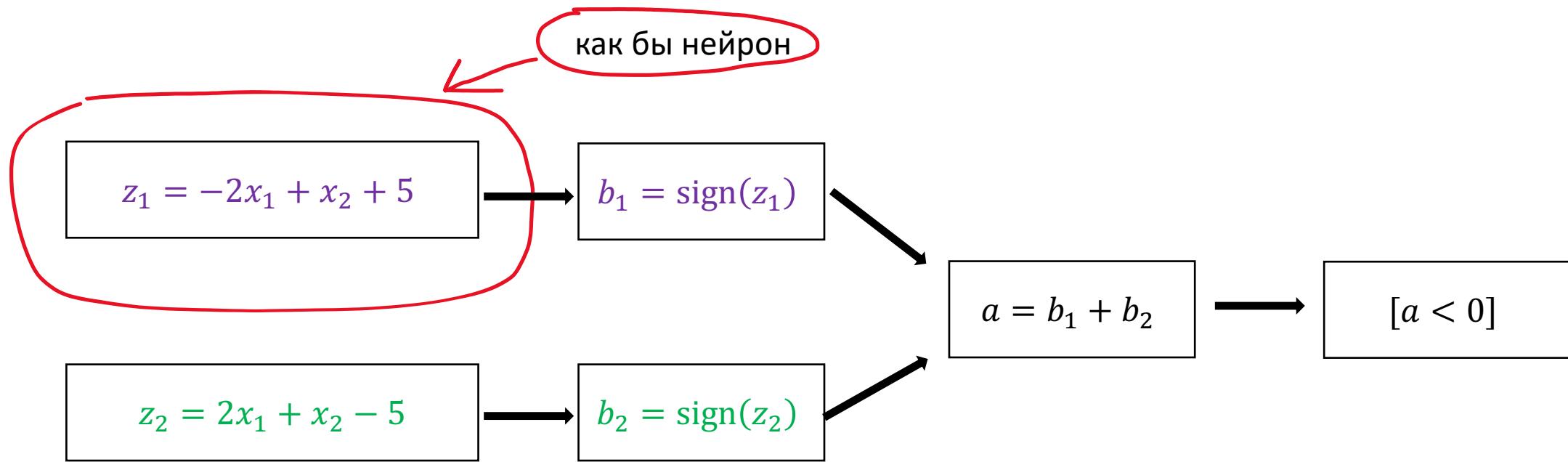


Нейрон

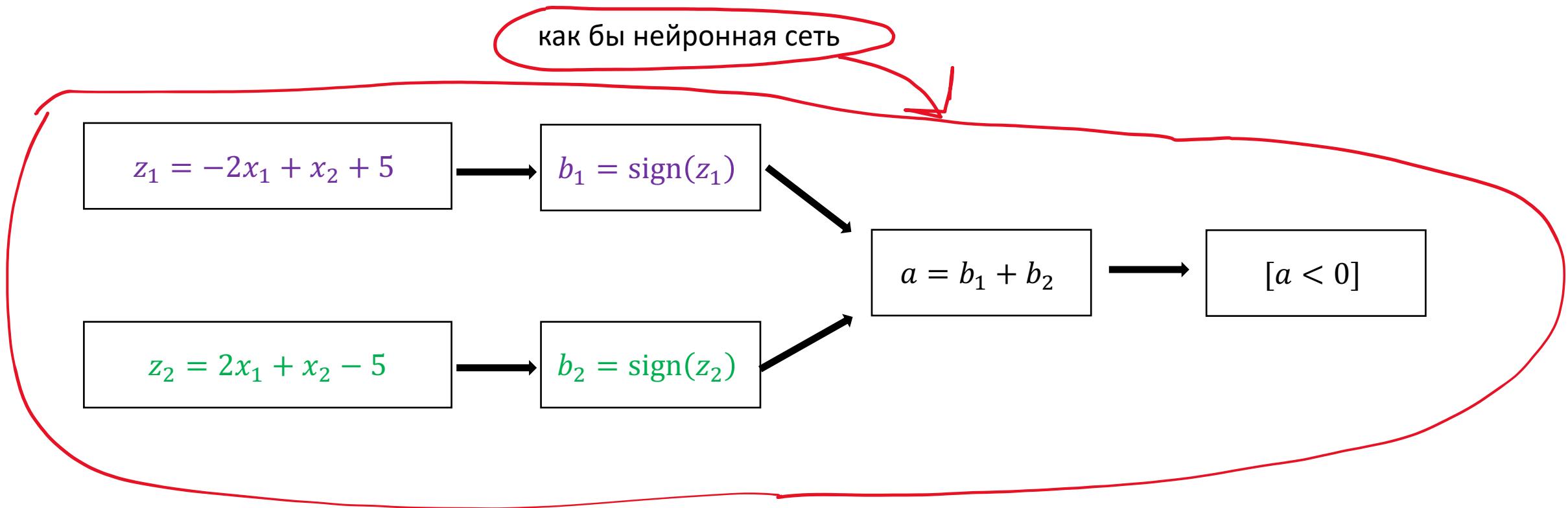


$$a(x) = \sum_{j=1}^d w_j x_j$$

Нелинейные закономерности

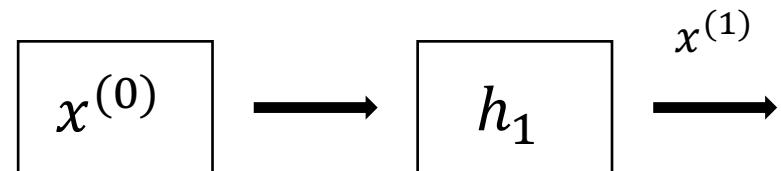


Нелинейные закономерности

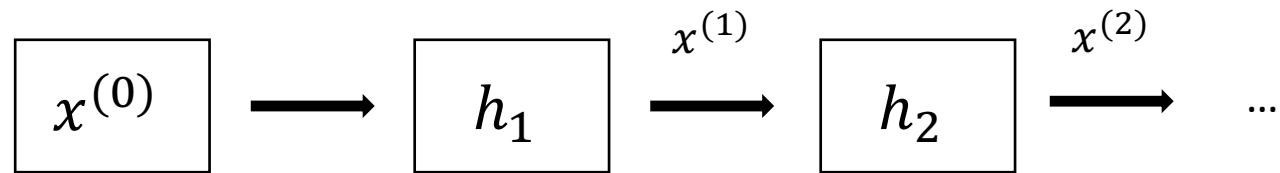


Граф вычислений (или нейронная сеть)

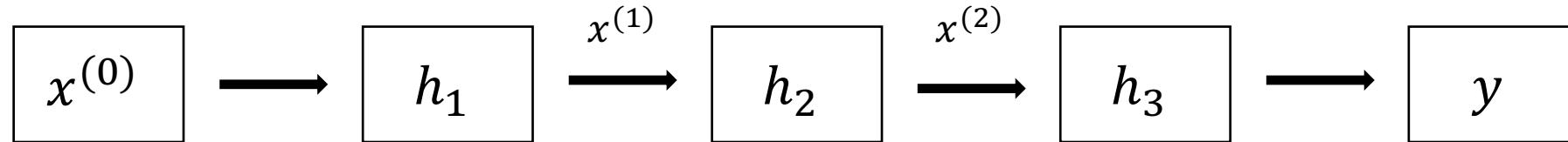
- $x^{(0)}$ — признаки объекта
- $h_1(x)$ — преобразование («слой»)
- $x^{(1)}$ — результат



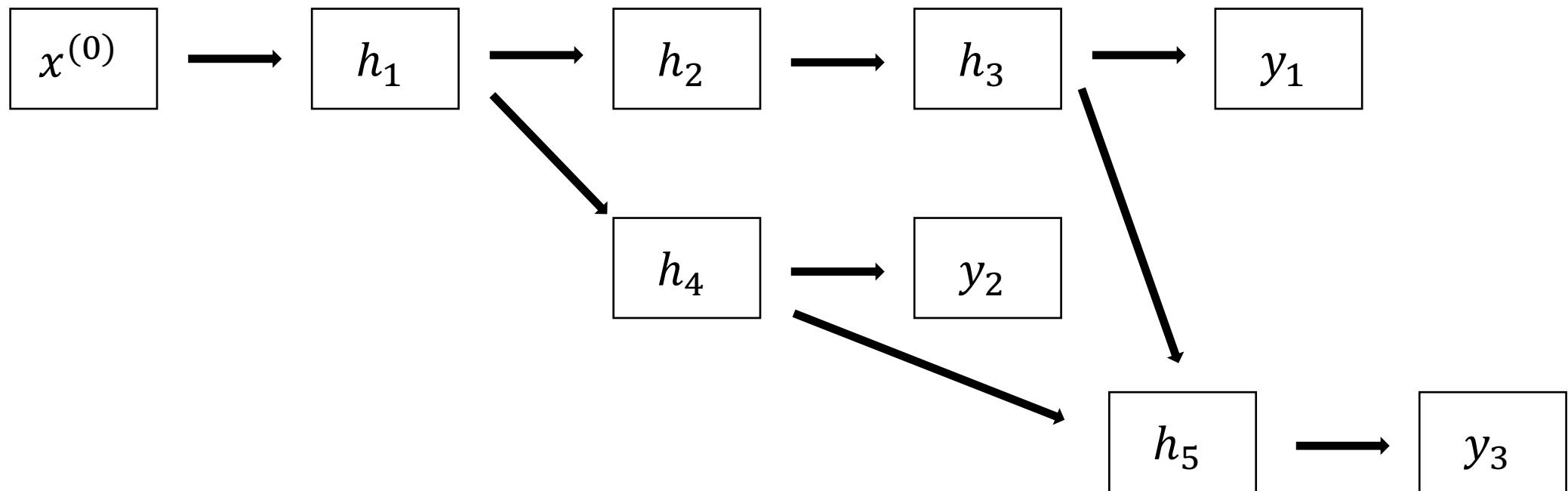
Граф вычислений (или нейронная сеть)



Граф вычислений (или нейронная сеть)



Граф вычислений (или нейронная сеть)



Резюме

- Нейронные сети — модели, состоящие из большого количества слоёв
- Каждый слой — модель, извлекающая новые признаки (в некотором смысле)

Полносвязные слои

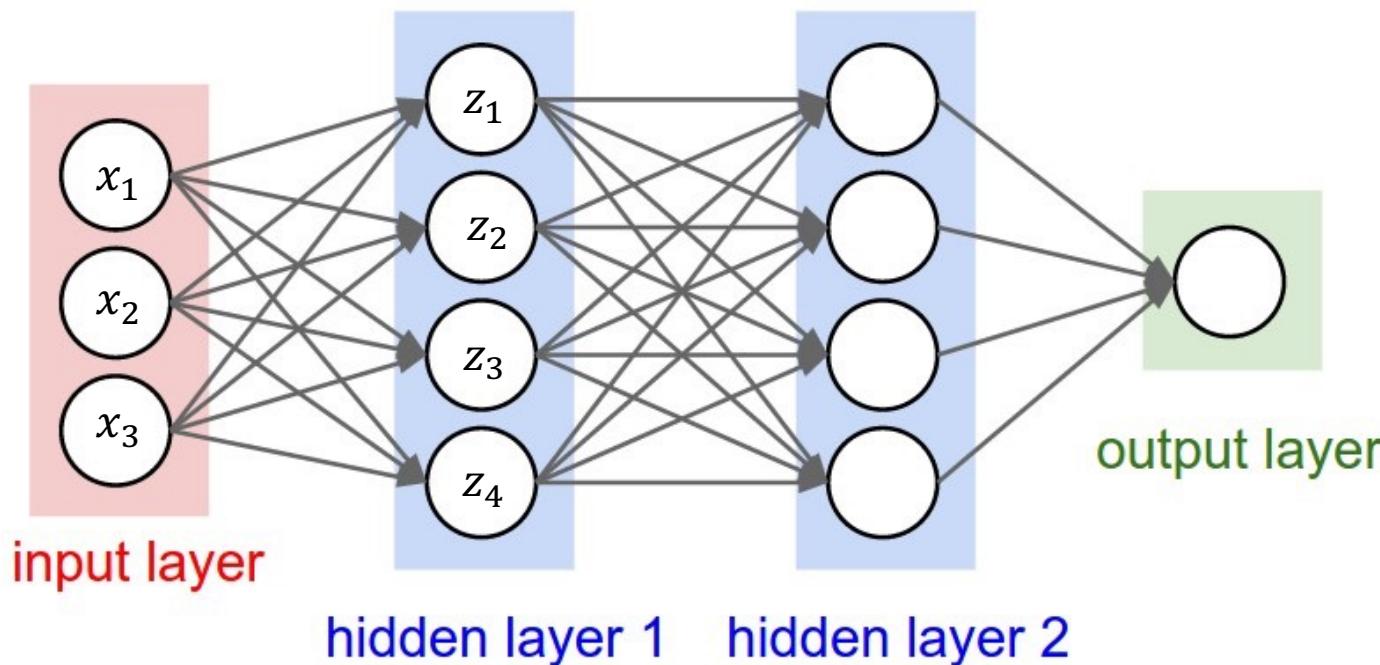
Полносвязный слой (fully connected, FC)

- 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)

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

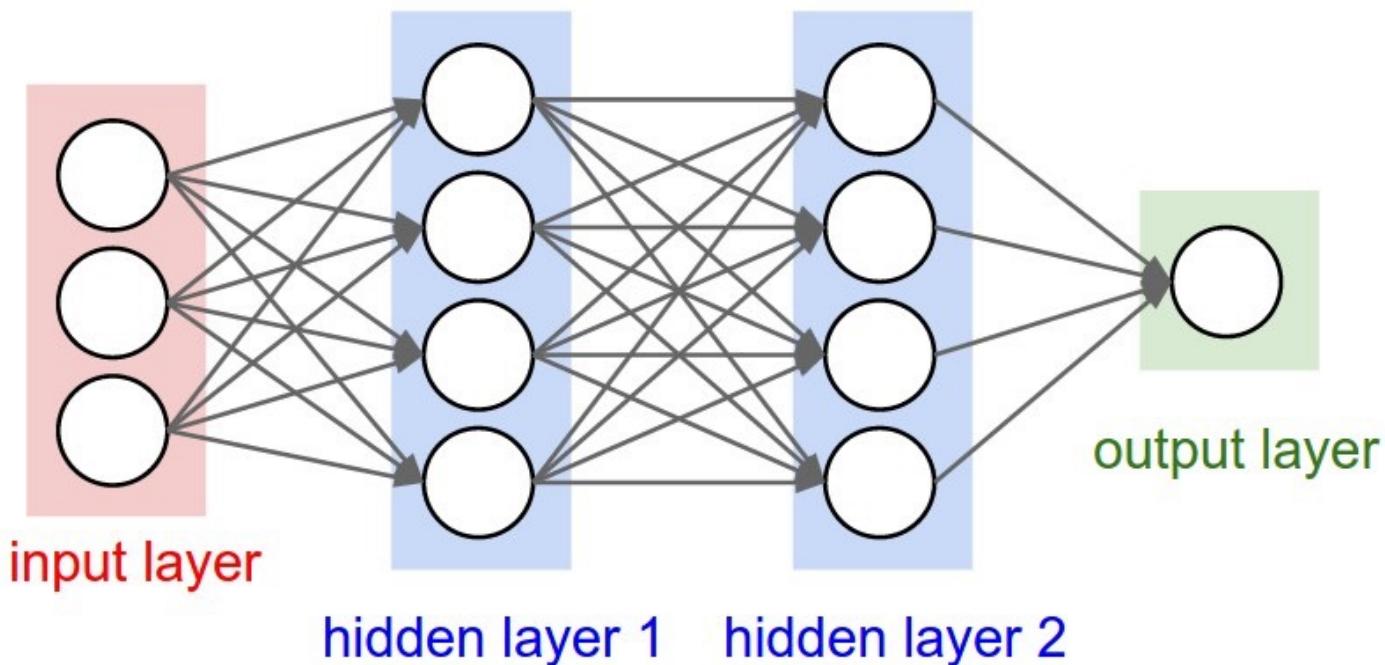
- t линейных моделей, в каждой $(n + 1)$ параметров
- Всего примерно tn параметров в полносвязном слое
- Это очень много: если у нас 1.000.000 входных признаков и 1000 выходов, то это 1.000.000.000 параметров

Важный вопрос в DL

Как объединить слои в мощную модель?

Нелинейность

- Рассмотрим два полно связанных слоя



Нелинейность

- Рассмотрим два полносвязных слоя

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

- То есть это ничем не лучше одного полносвязного слоя

Нелинейность

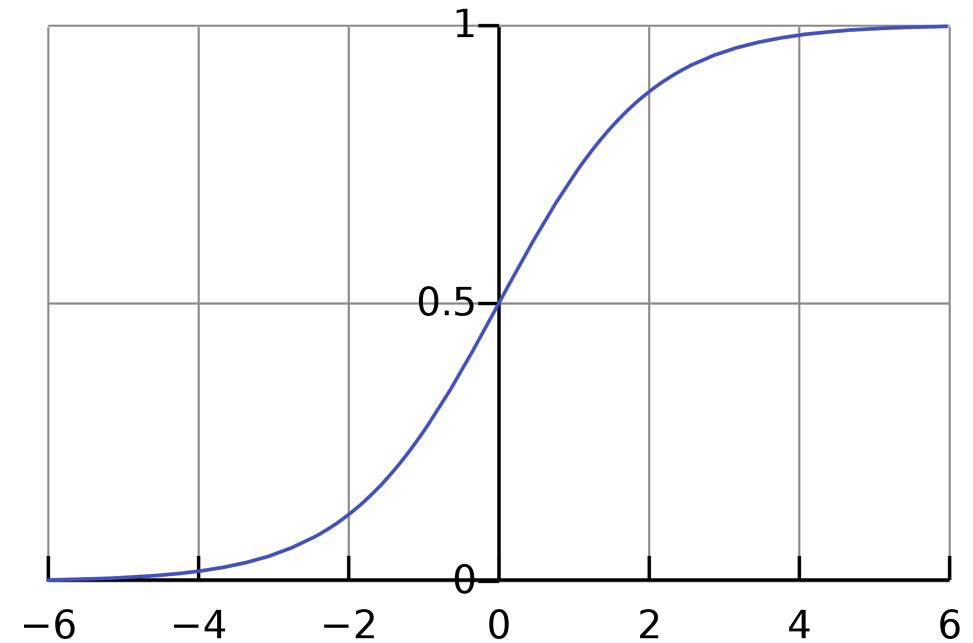
- Нужно добавлять нелинейную функцию после полносвязного слоя

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

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)

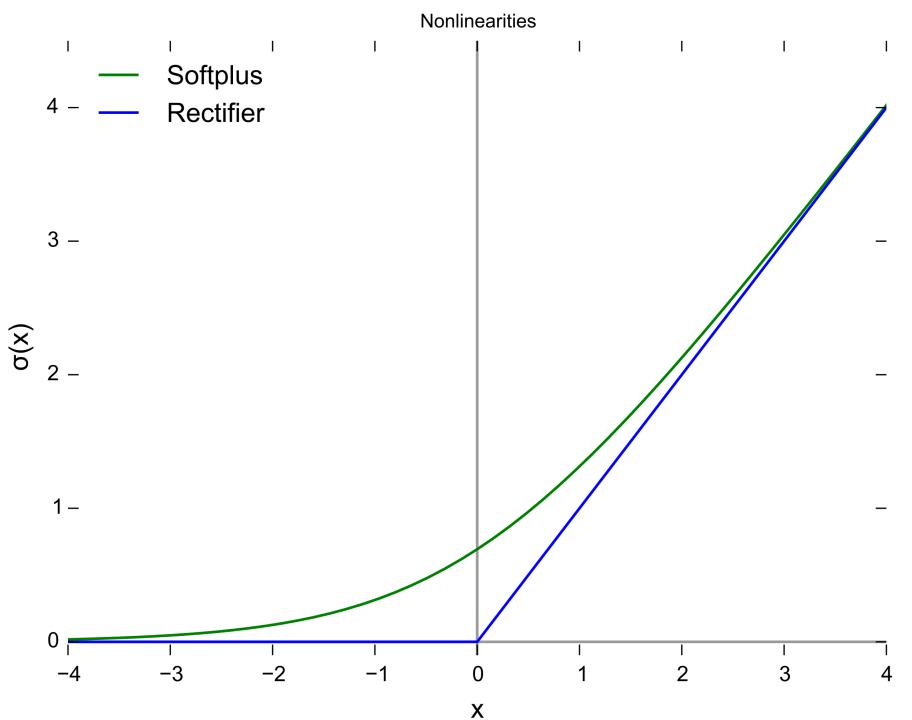


Нелинейность

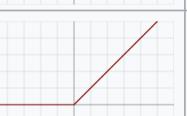
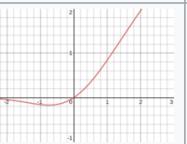
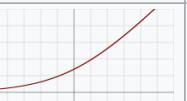
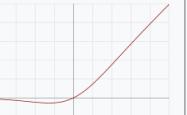
$$z_j = f \left(\sum_{i=1}^n w_{ji} x_i + b_j \right)$$

Вариант 2: $f(x) = \max(0, x)$

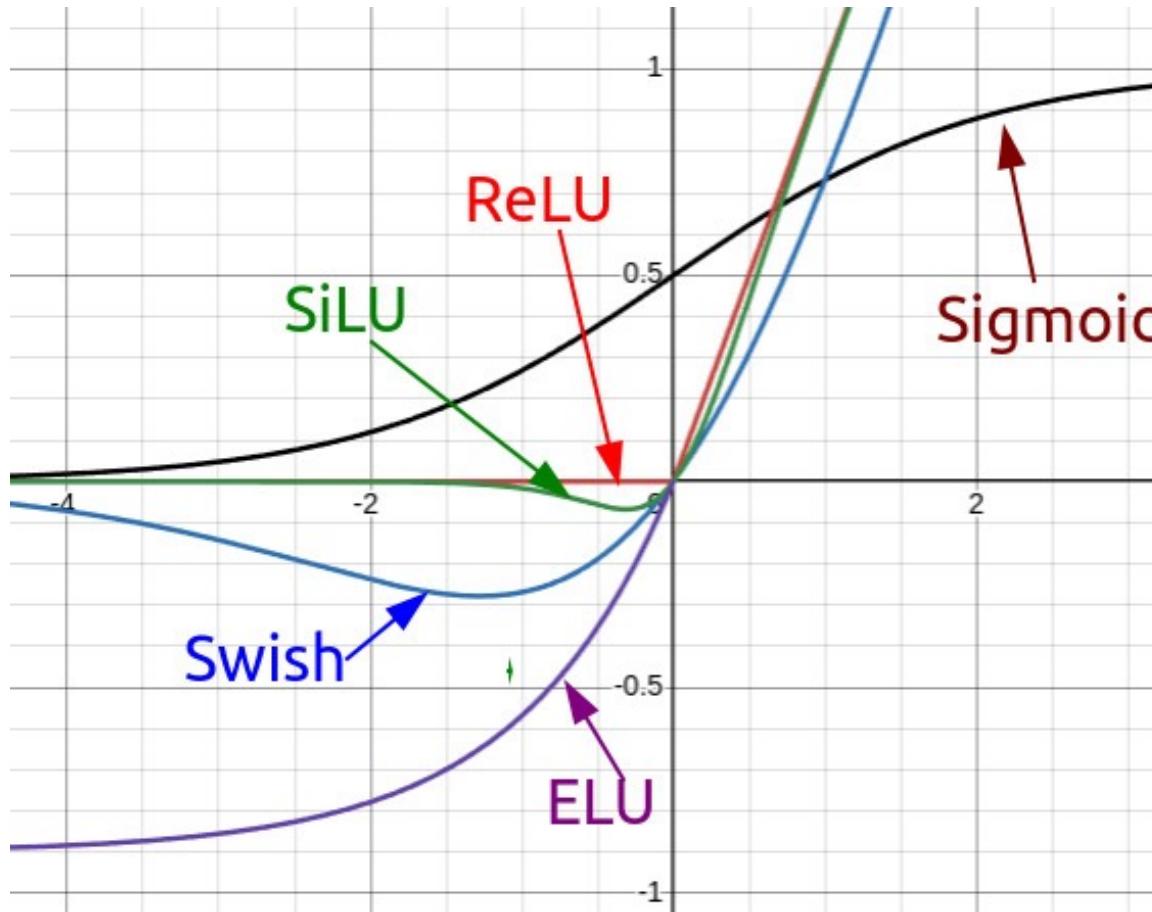
(ReLU, REctified Linear Unit)



Нелинейность

Rectified linear unit (ReLU) ^[9]		$\begin{cases} 0 & \text{if } x \le 0 \\ x & \text{if } x > 0 \end{cases} = \max\{0, x\} = x \mathbf{1}_{x>0}$
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]		$\begin{cases} \alpha(e^x - 1) & \text{if } x \le 0 \\ x & \text{if } x > 0 \end{cases} \quad \text{with parameter } \alpha$
Scaled exponential linear unit (SELU) ^[12]		$\lambda \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \ge 0 \end{cases} \quad \text{with parameters } \lambda = 1.0507 \text{ and } \alpha = 1.67326$
Leaky rectified linear unit (Leaky ReLU) ^[13]		$\begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \ge 0 \end{cases}$
Parameteric rectified linear unit (PReLU) ^[14]		$\begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \ge 0 \end{cases} \quad \text{with parameter } \alpha$
Sigmoid linear unit (SiLU, ^[4] Sigmoid shrinkage, ^[15] SiL, ^[16] or Swish-1 ^[17])		$\frac{x}{1 + e^{-x}}$

Нелинейность



Нелинейность

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

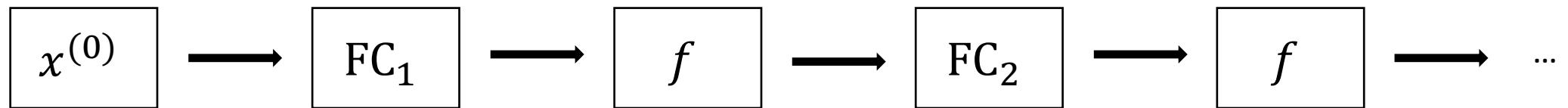
$$\text{GEGLU}(x, W, V, b, c) = \text{GELU}(xW + b) \otimes (xV + c)$$

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

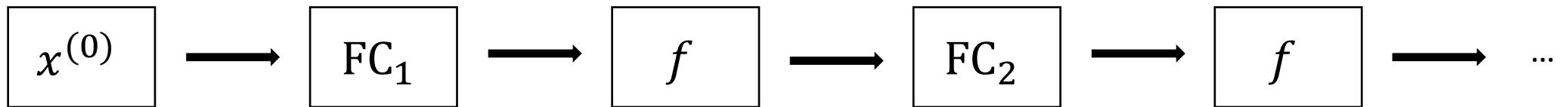
4 Conclusions

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



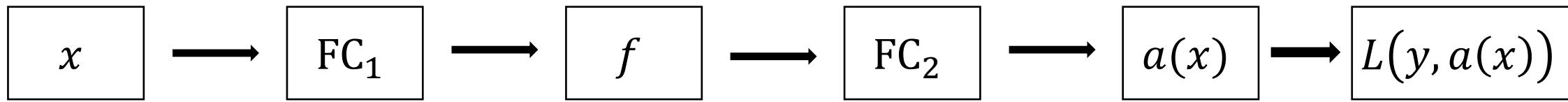
Типичная полносвязная сеть



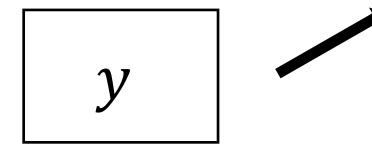
- На входе признаки
- В последнем слое выходов столько, сколько целевых переменных мы предсказываем

Обучение нейронных сетей

- Все слои обычно дифференцируемы, поэтому можно посчитать производные по всем параметрам

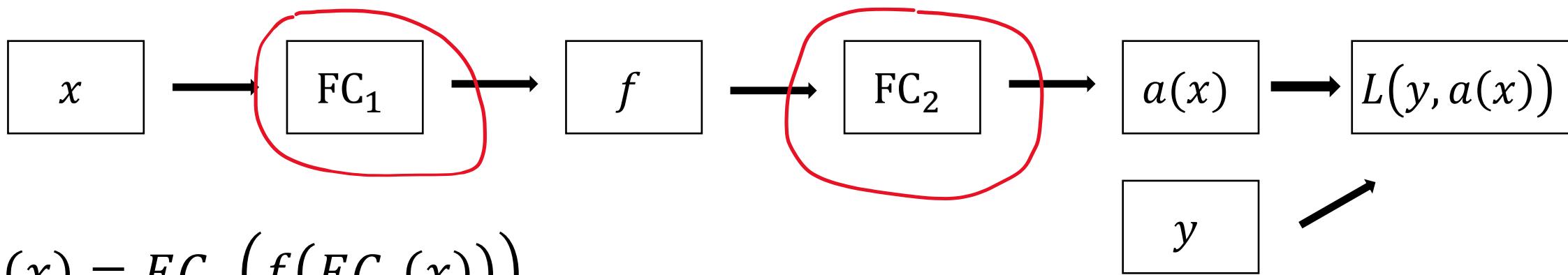


- $a(x) = FC_2(f(FC_1(x)))$
- Где здесь параметры?



Обучение нейронных сетей

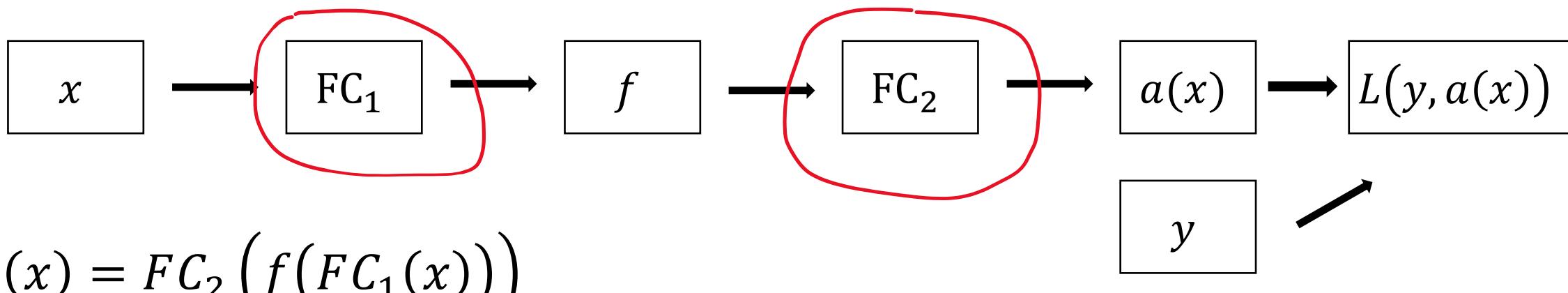
- Все слои обычно дифференцируемы, поэтому можно посчитать производные по всем параметрам



- $a(x) = FC_2(f(FC_1(x)))$
- Где здесь параметры?

Обучение нейронных сетей

- Все слои обычно дифференцируемы, поэтому можно посчитать производные по всем параметрам



$$\cdot a(x) = FC_2(f(FC_1(x)))$$

$$\frac{1}{\ell} \sum_{i=1}^{\ell} L(y_i, a(x_i)) \rightarrow \min_a$$

Backprop

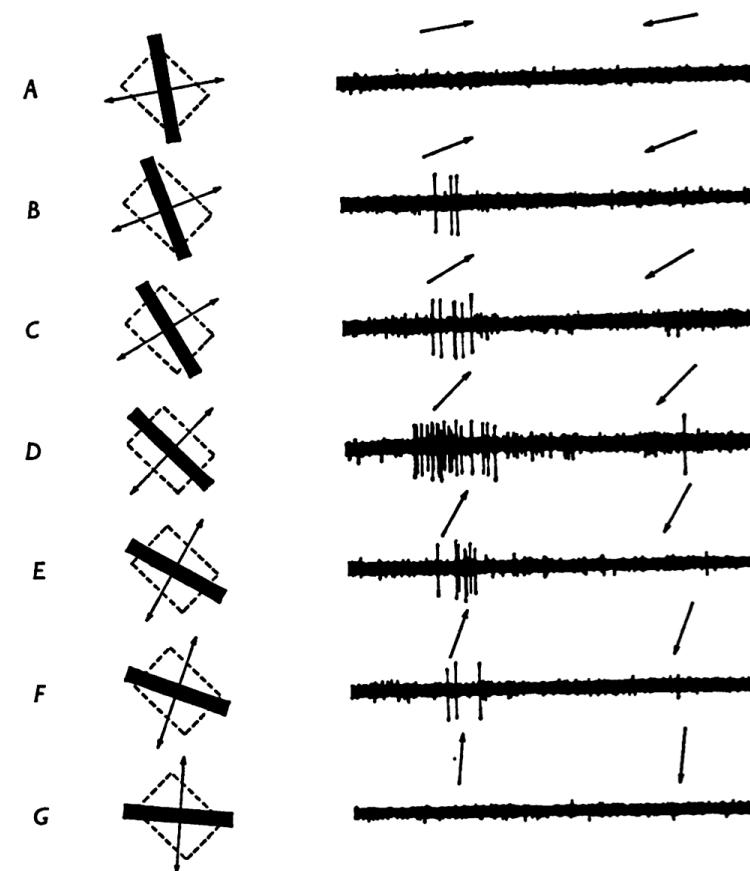
- Во многие формулы входят одни и те же производные
- В backprop каждая частная производная вычисляется один раз — вычисление производных по слою N сводится к перемножению матрицы производных по слою $N+1$ и некоторых векторов

Резюме

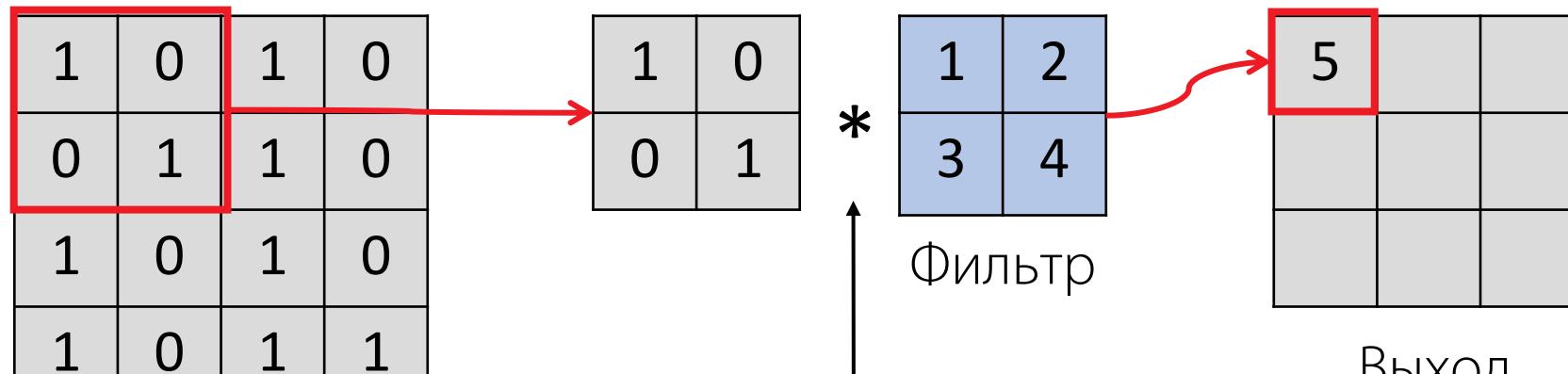
- Backprop — метод вычисления градиентов для нейронных сетей
- Градиент ошибки по слою зависит от всех более поздних слоёв
- Требуется много вычислений

Свёртки

Эксперименты со зрительной корой



Свёртка

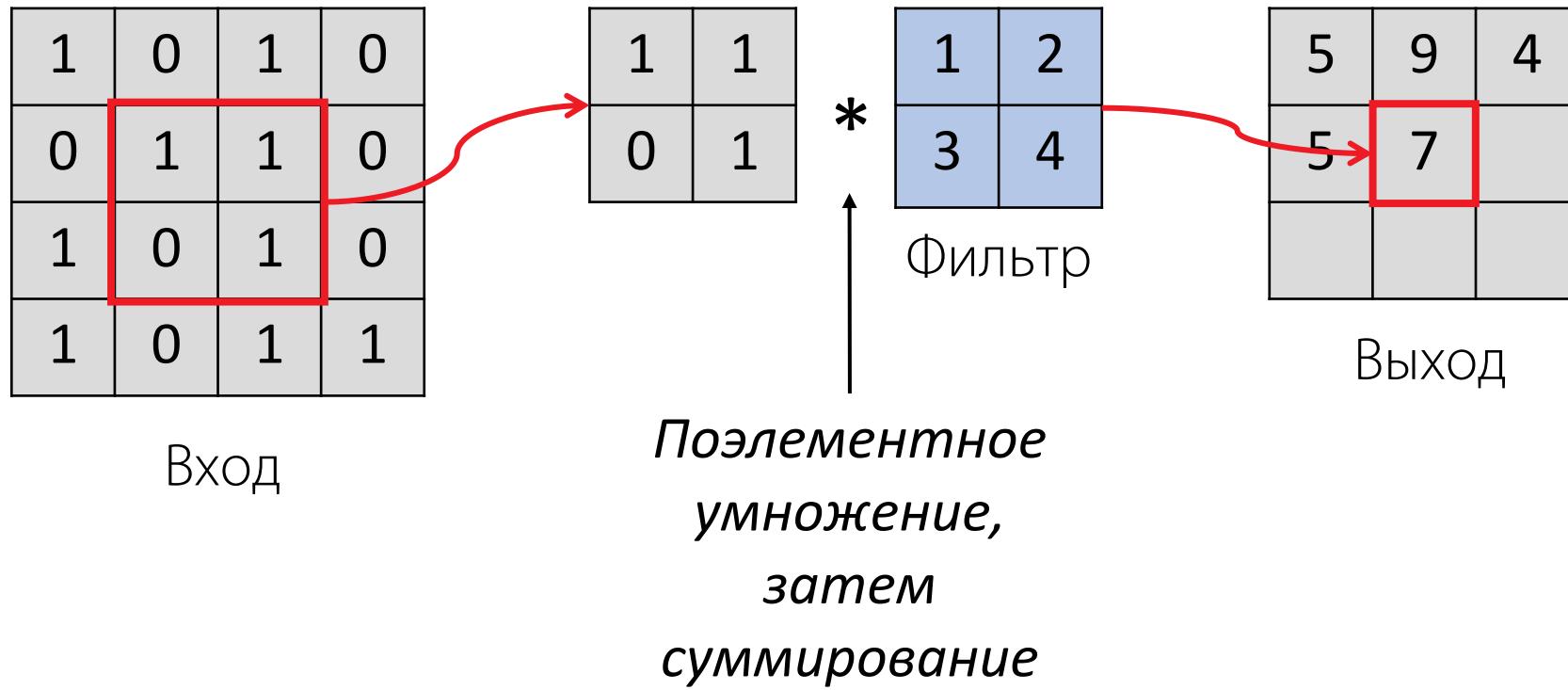


Вход

*Поэлементное
умножение,
затем
суммирование*

Выход

Свёртка



Свёртка

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 2$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 2$$

$$\begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 6$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 1$$

$$\begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 10$$

$$\begin{bmatrix} 0 & 2 \\ 3 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 0$$

Свёртка

- Операция свёртки выявляет наличие на изображении паттерна, который задаётся фильтром
- Чем сильнее на участке изображения представлен паттерн, тем больше будет значение свёртки

Максимум свёртки инвариантен к сдвигам

0	0	0	0
0	0	0	0
0	0	1	0
0	0	0	1

Вход

*

1	0
0	1

Фильтр

=

0	0	0
0	1	0
0	0	2

Выход

Max = 2



1	0	0	0
0	1	0	0
0	0	0	0
0	0	0	0

Вход

*

1	0
0	1

Фильтр

=

2	0	0
0	1	0
0	0	0

Выход

Не меняется



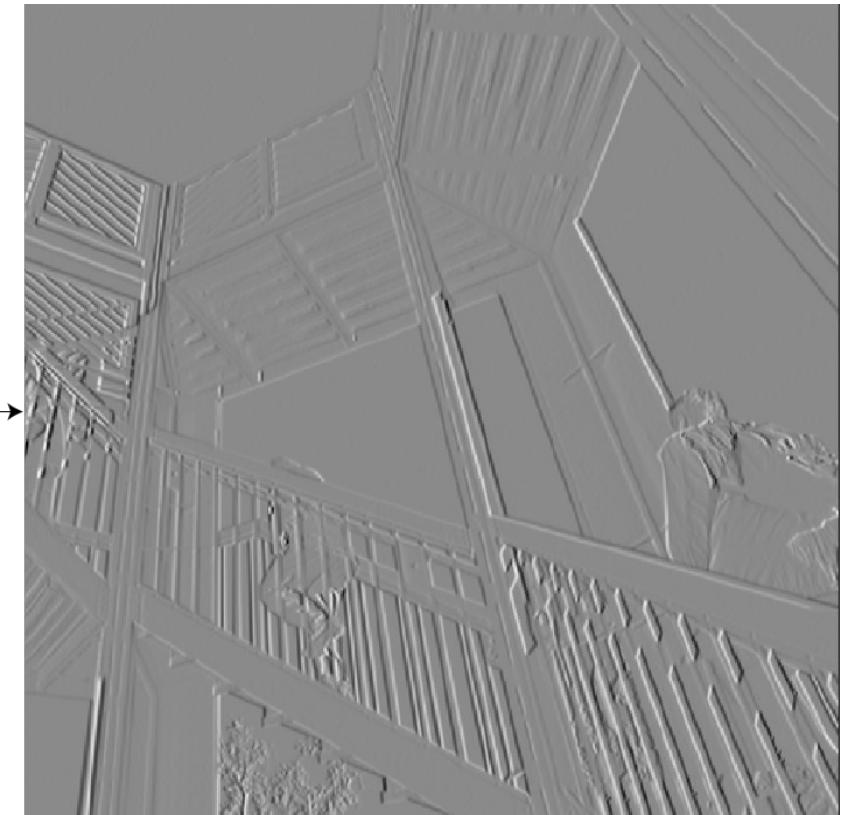
Max = 2

Свёртки в компьютерном зрении



$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Horizontal Sobel kernel

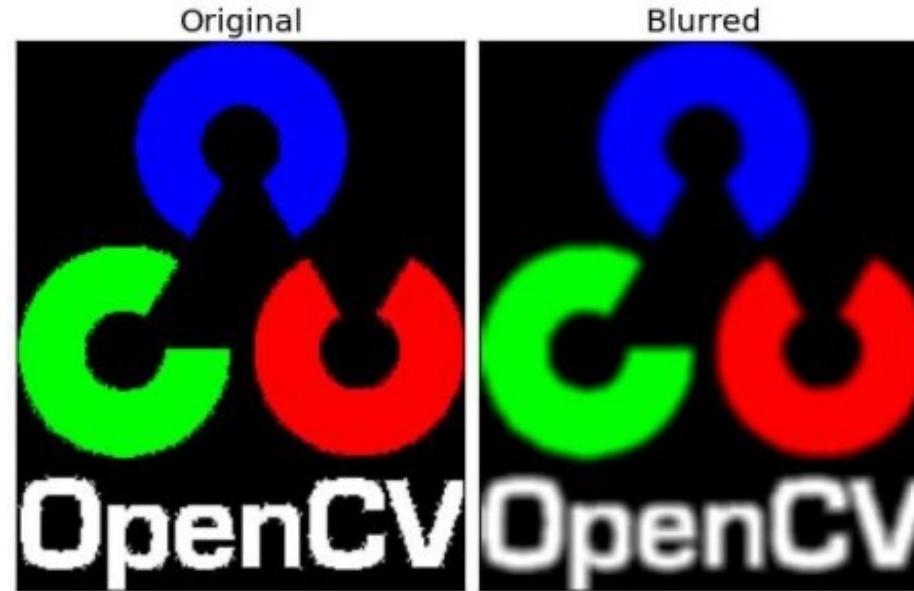


Свёртки в компьютерном зрении



$$\begin{array}{|c|c|c|} \hline \bullet 0 & \bullet 0 & \bullet 0 \\ \hline \bullet 0 & \bullet 1 & \bullet 0 \\ \hline \bullet 0 & \bullet 0 & \bullet 0 \\ \hline \end{array} + \begin{array}{|c|c|c|} \hline \bullet 0 & \bullet 0 & \bullet 0 \\ \hline \bullet 0 & \bullet 1 & \bullet 0 \\ \hline \bullet 0 & \bullet 0 & \bullet 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|} \hline \bullet 1 & \bullet 1 & \bullet 1 \\ \hline \bullet 1 & \bullet 1 & \bullet 1 \\ \hline \bullet 1 & \bullet 1 & \bullet 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline \bullet 0 & \bullet 0 & \bullet 0 \\ \hline \bullet 0 & \bullet 2 & \bullet 0 \\ \hline \bullet 0 & \bullet 0 & \bullet 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|} \hline \bullet 1 & \bullet 1 & \bullet 1 \\ \hline \bullet 1 & \bullet 1 & \bullet 1 \\ \hline \bullet 1 & \bullet 1 & \bullet 1 \\ \hline \end{array}$$

Свёртки в компьютерном зрении



$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Резюме

- Свёртки — специализированные слои, подходящие для изображений
- Учитывают специфику: локальные характеристики, поиск паттернов