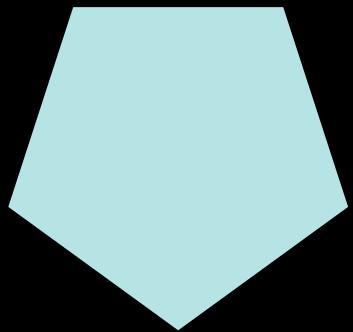




Георгий Сарапулов

Введение в глубокое обучение

Классическое обучение с учителем



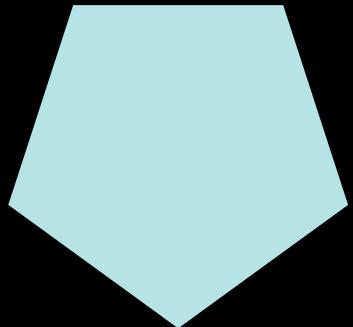
Объект



Свойство

Классическое обучение с учителем

Модель

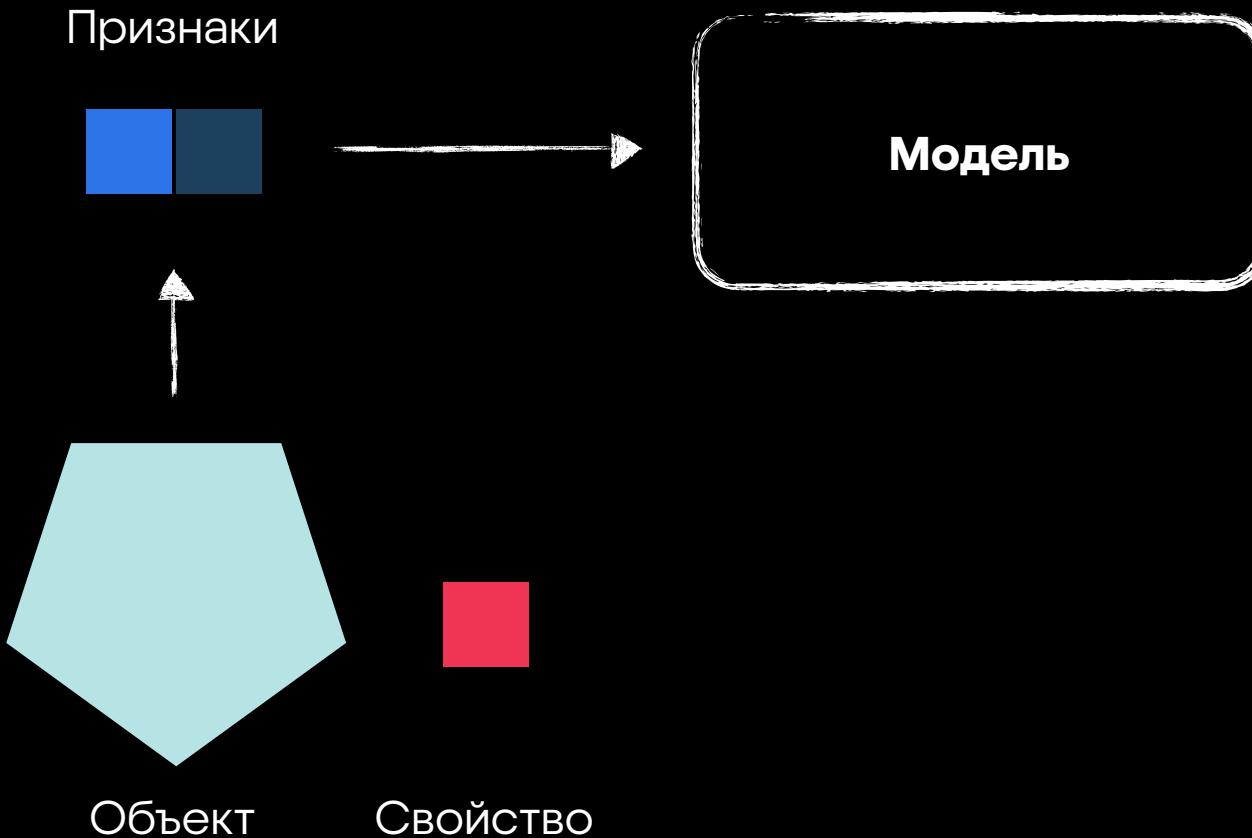


Объект

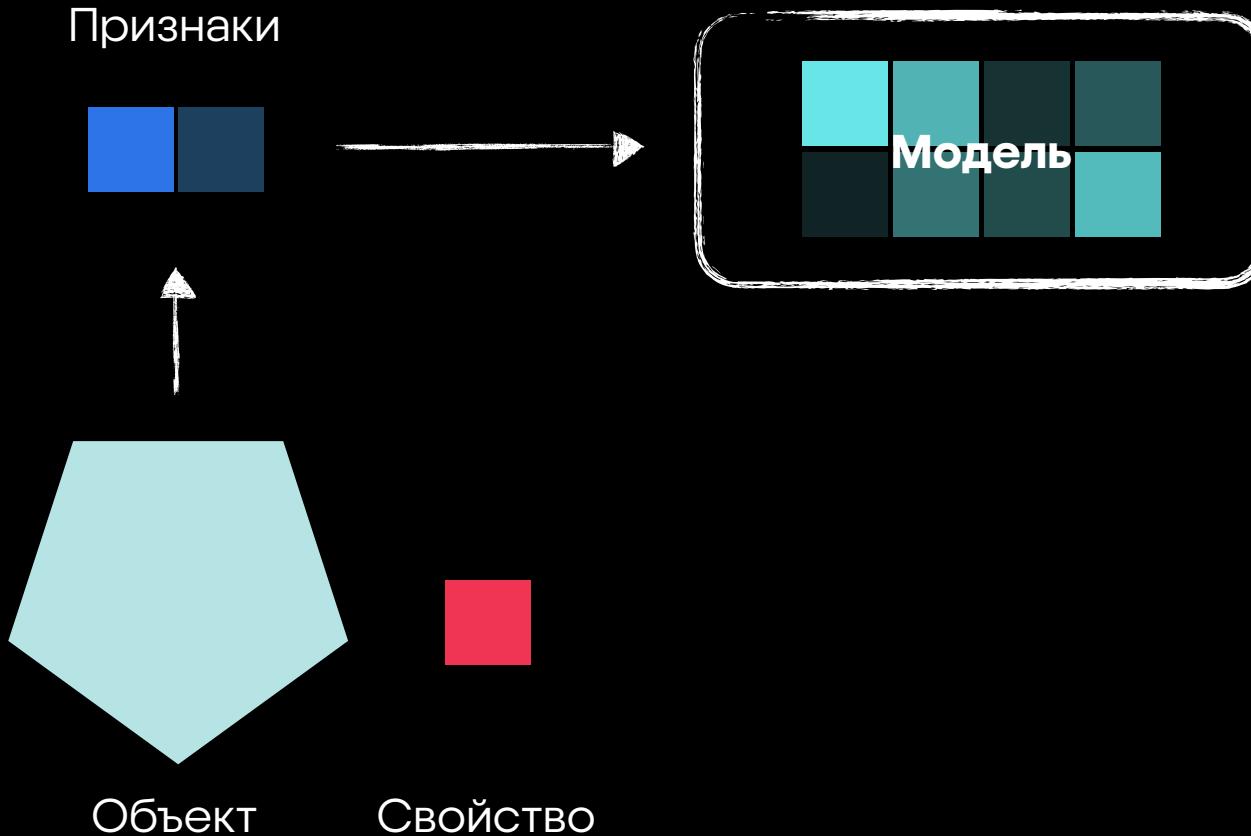


Свойство

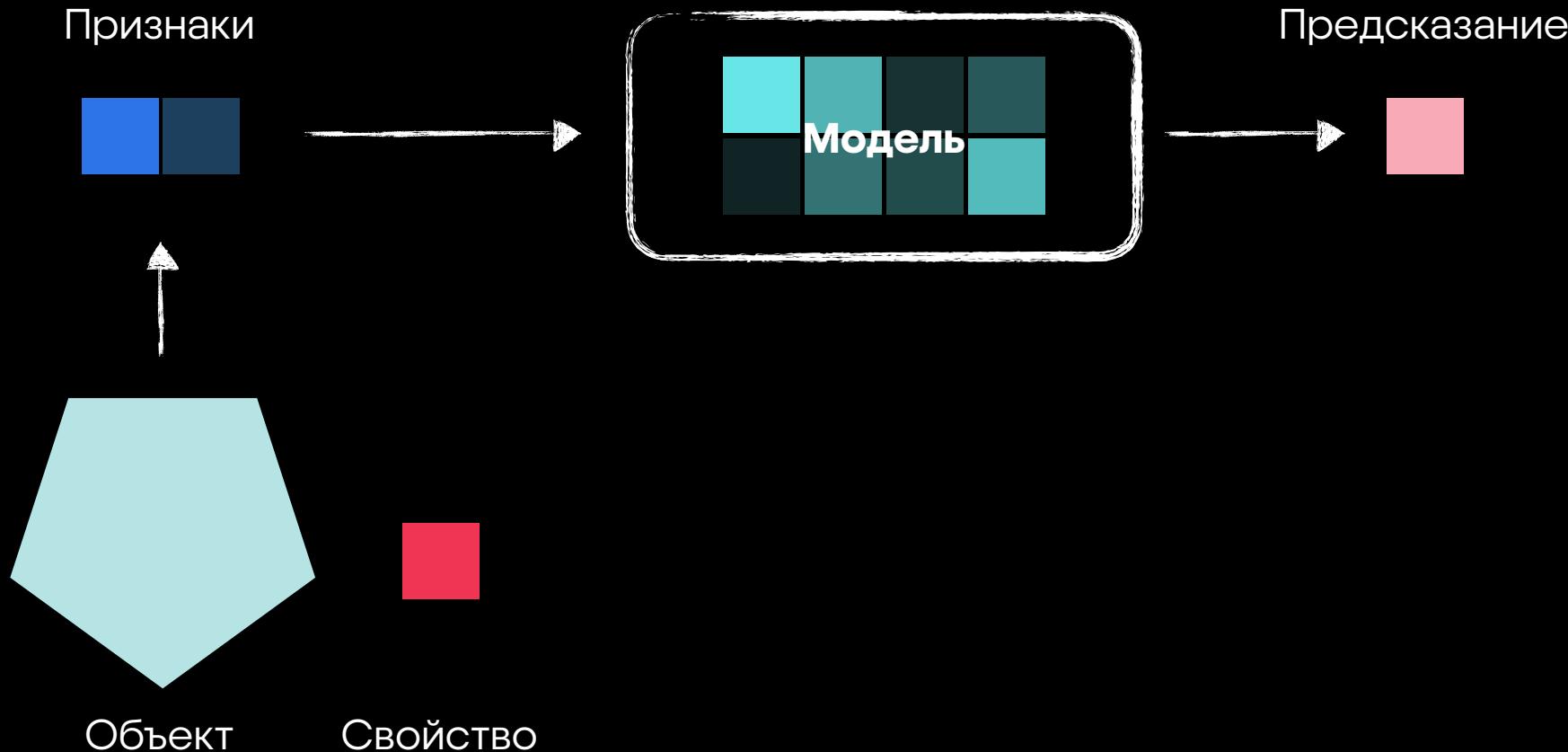
Классическое обучение с учителем



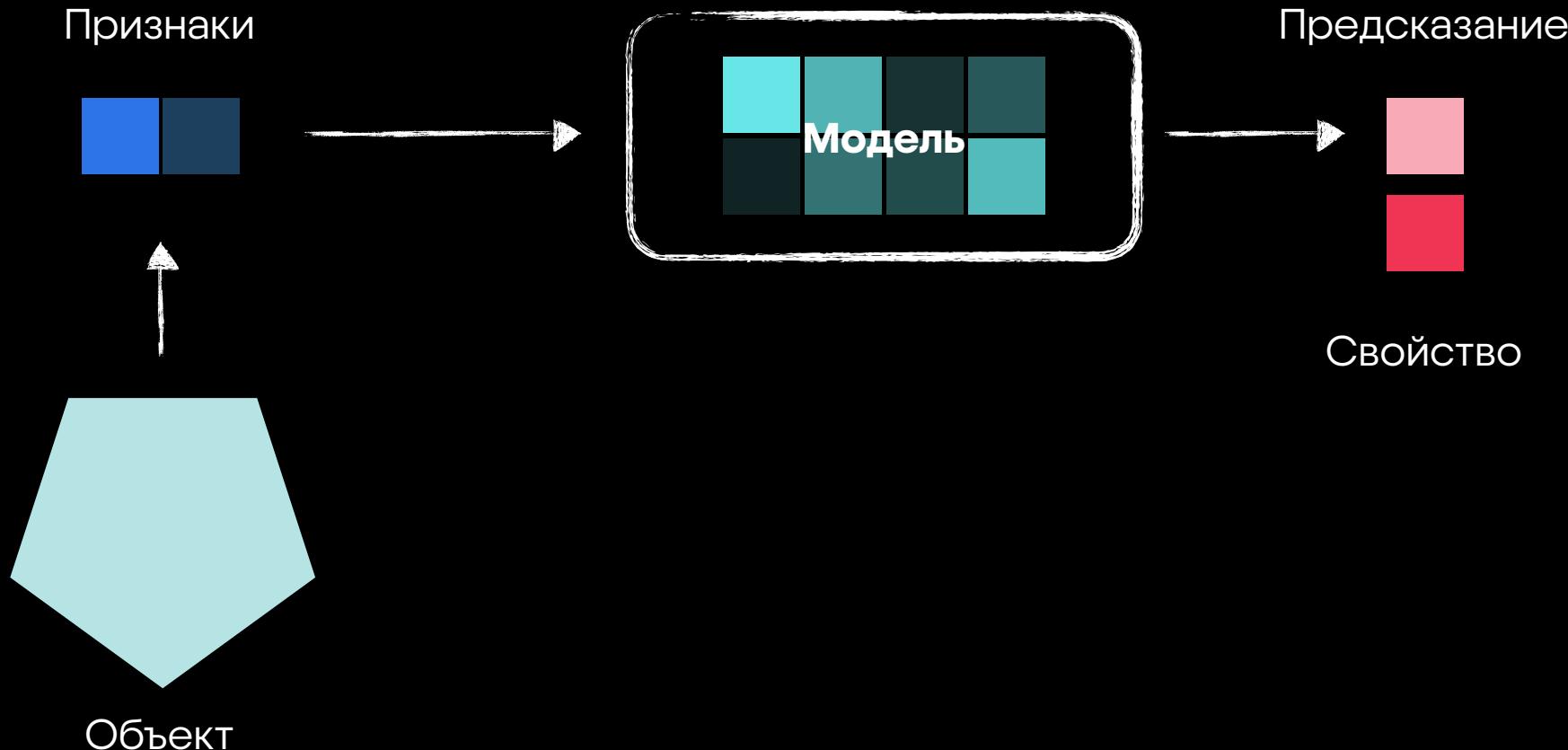
Классическое обучение с учителем



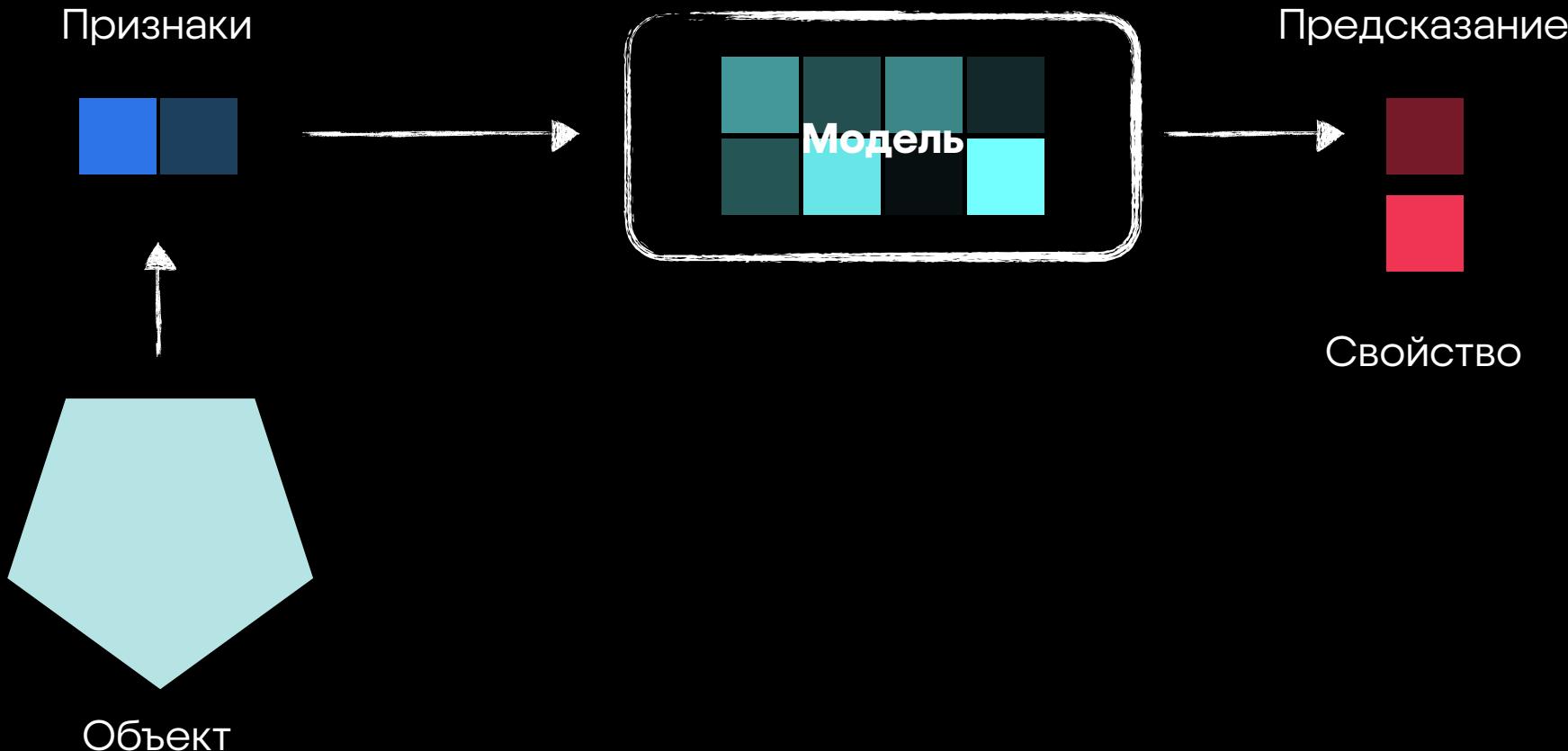
Классическое обучение с учителем



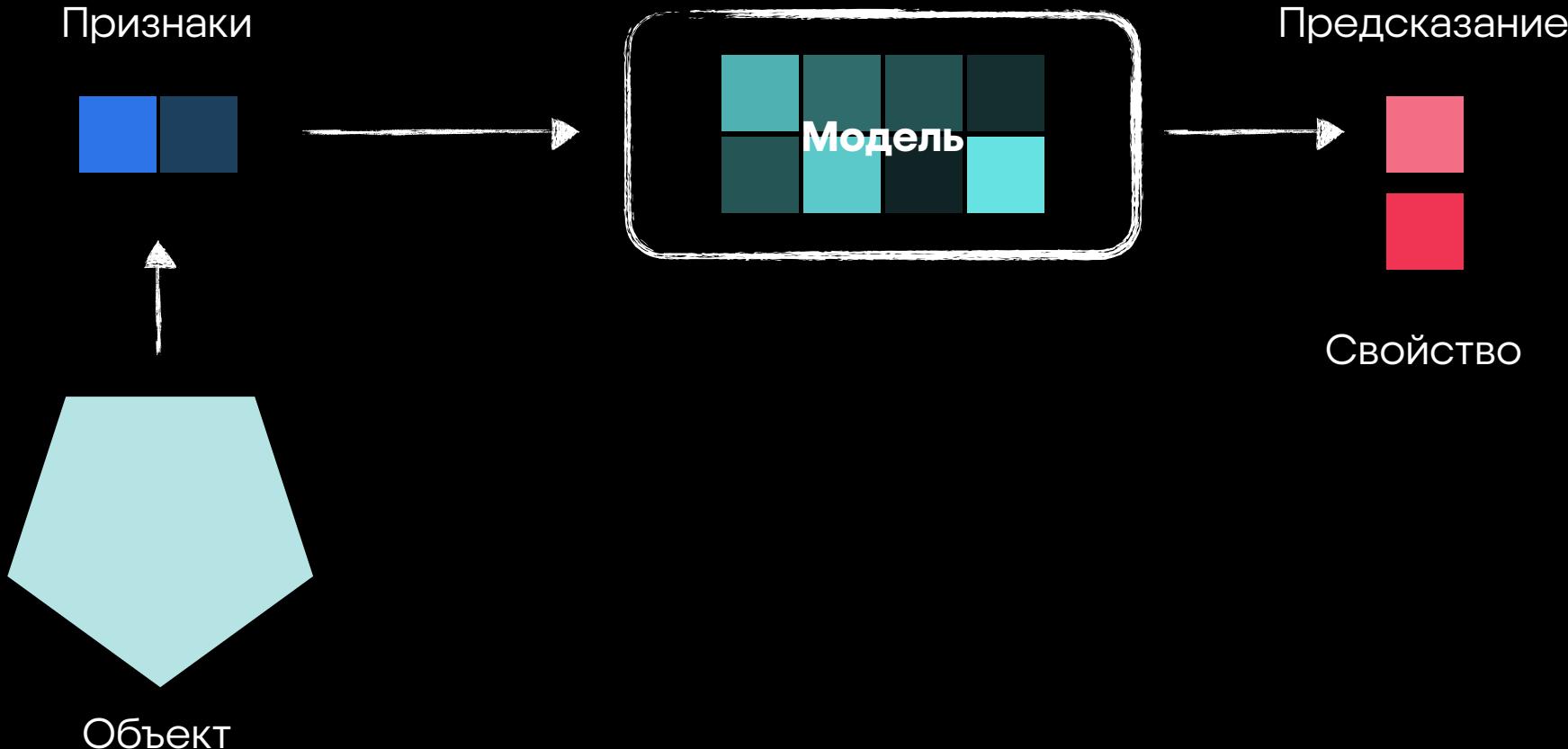
Классическое обучение с учителем



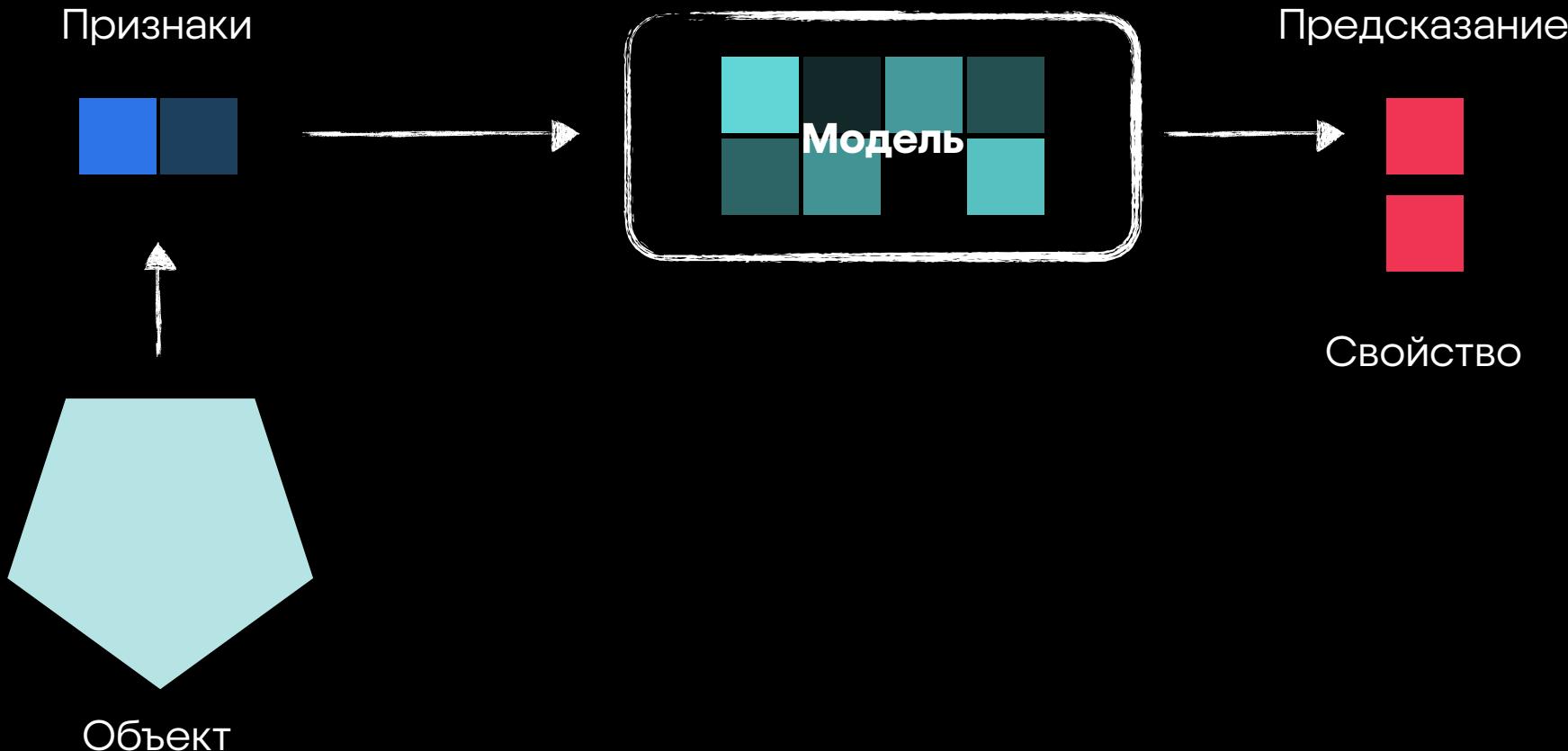
Классическое обучение с учителем



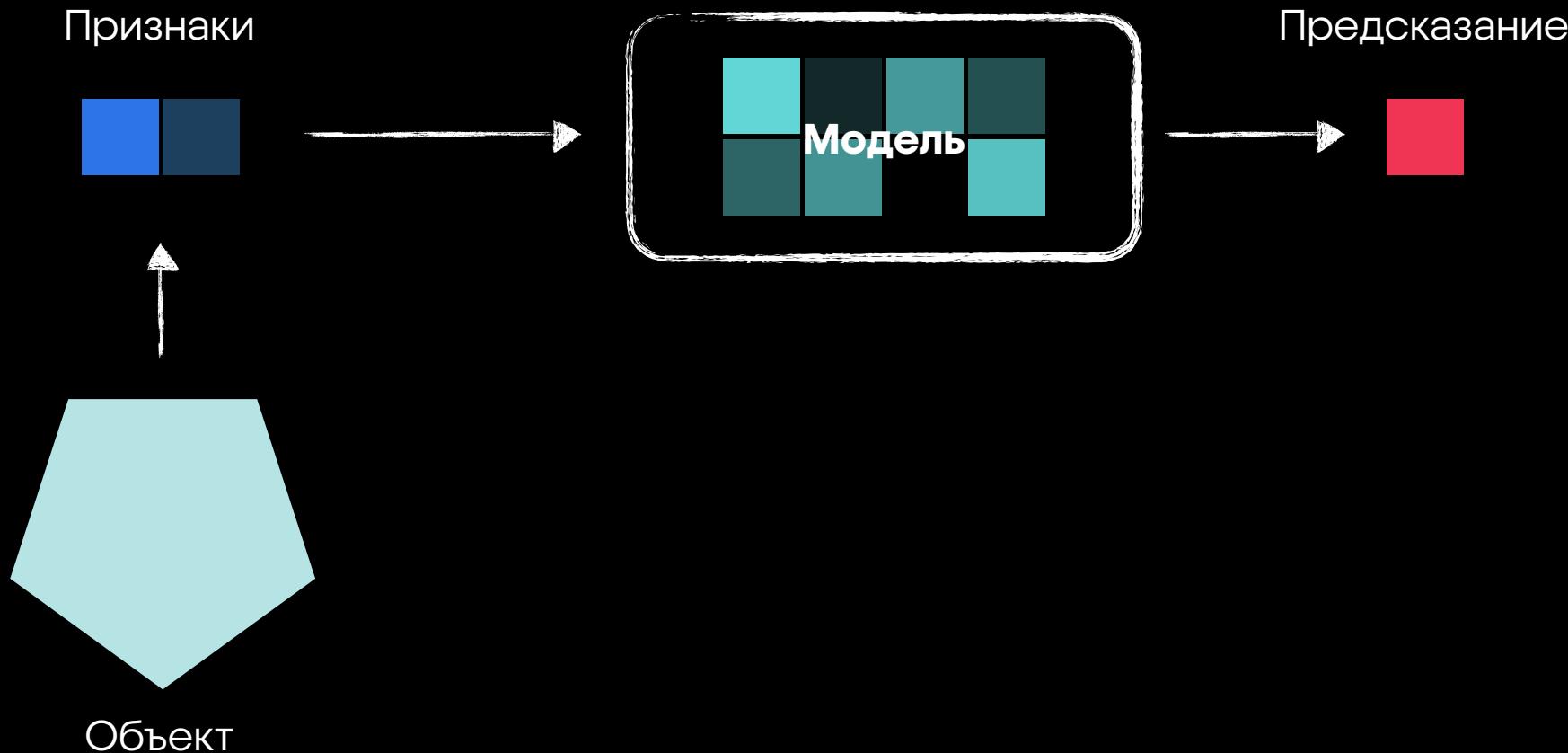
Классическое обучение с учителем



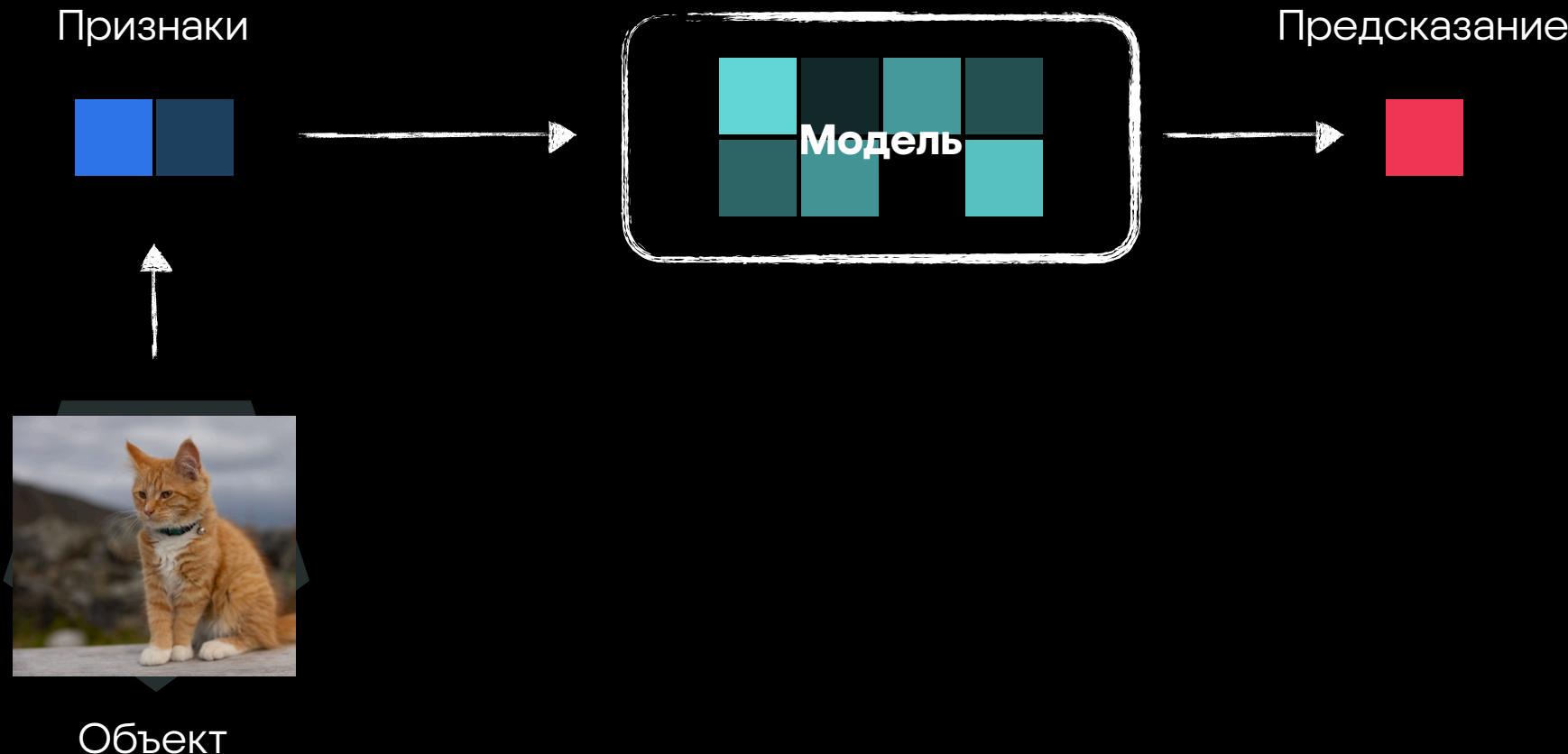
Классическое обучение с учителем



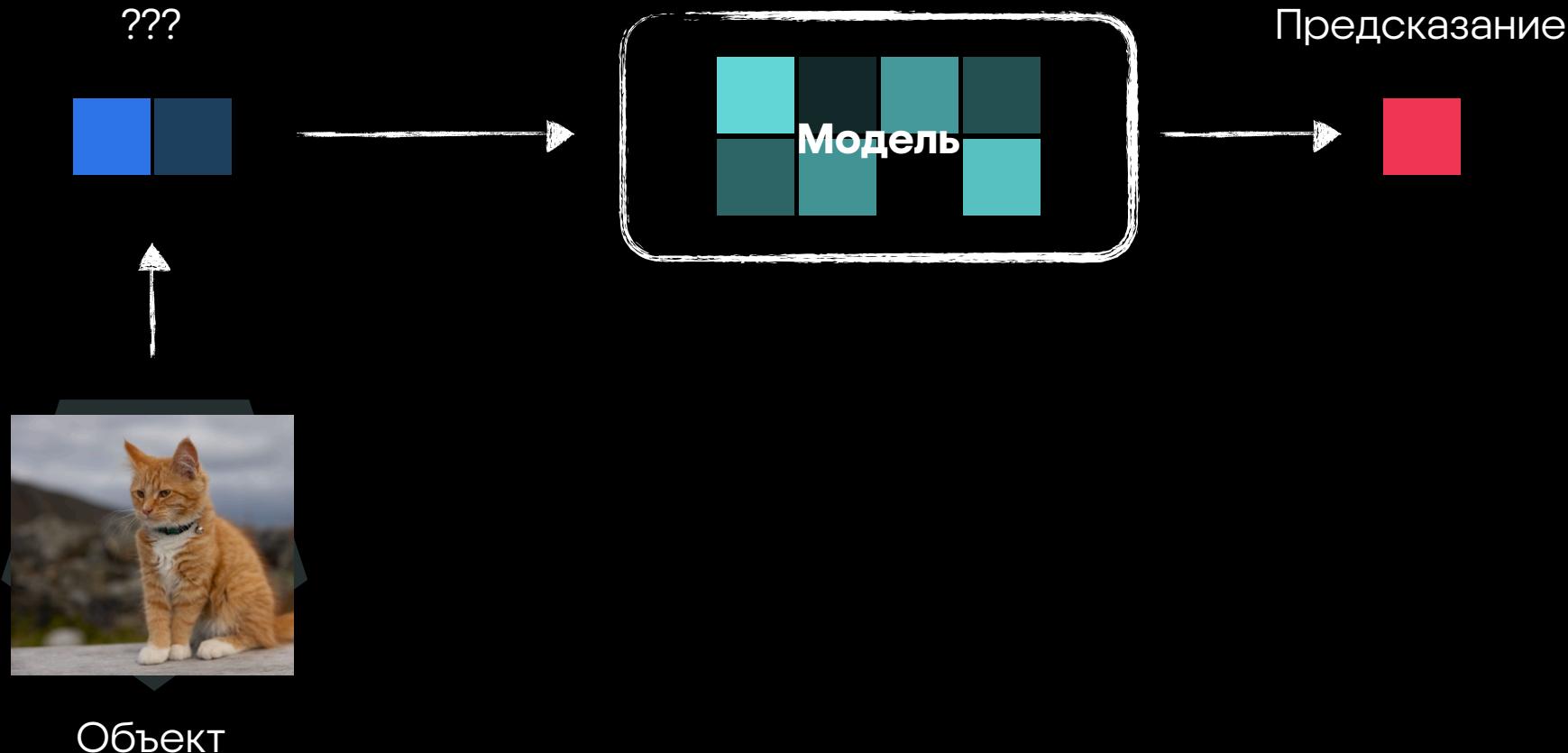
Сложные объекты



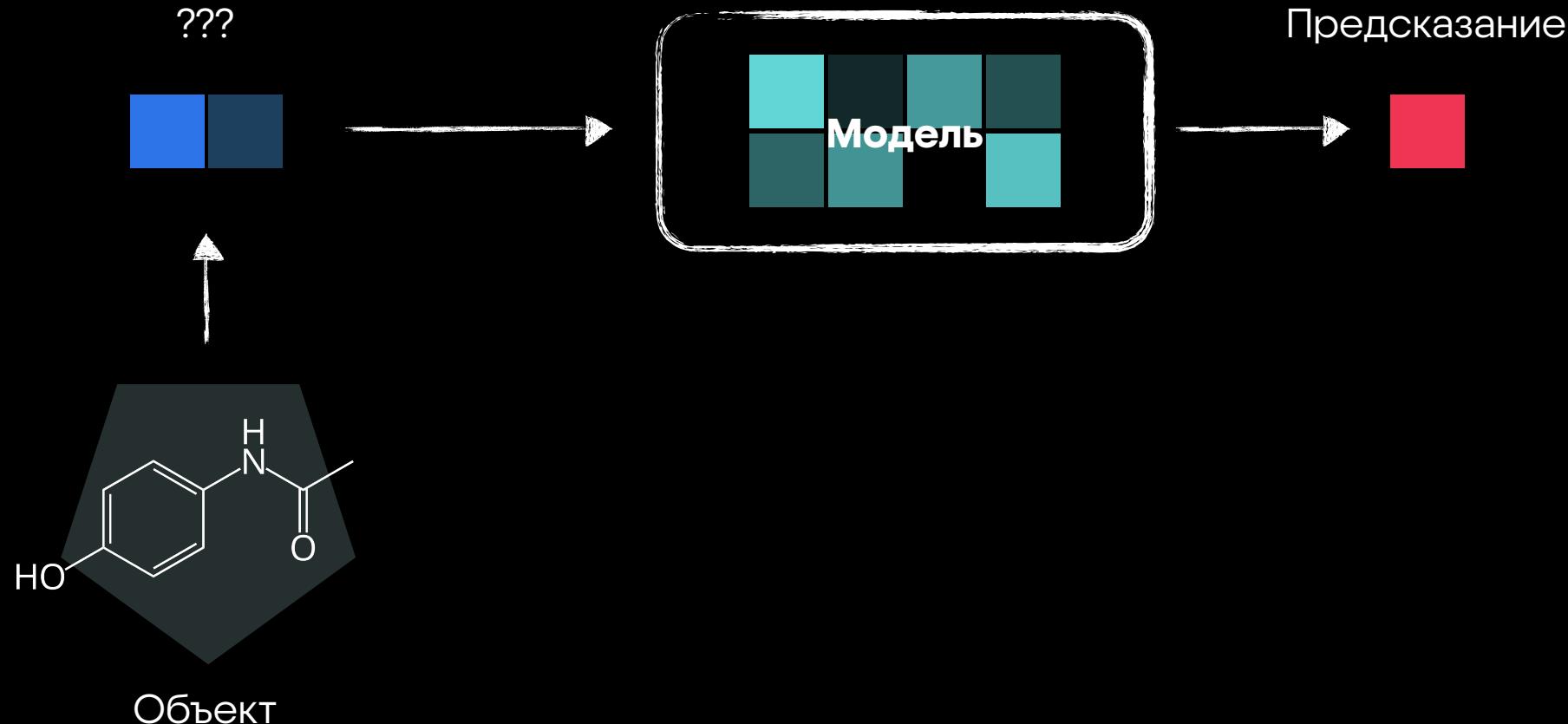
Сложные объекты



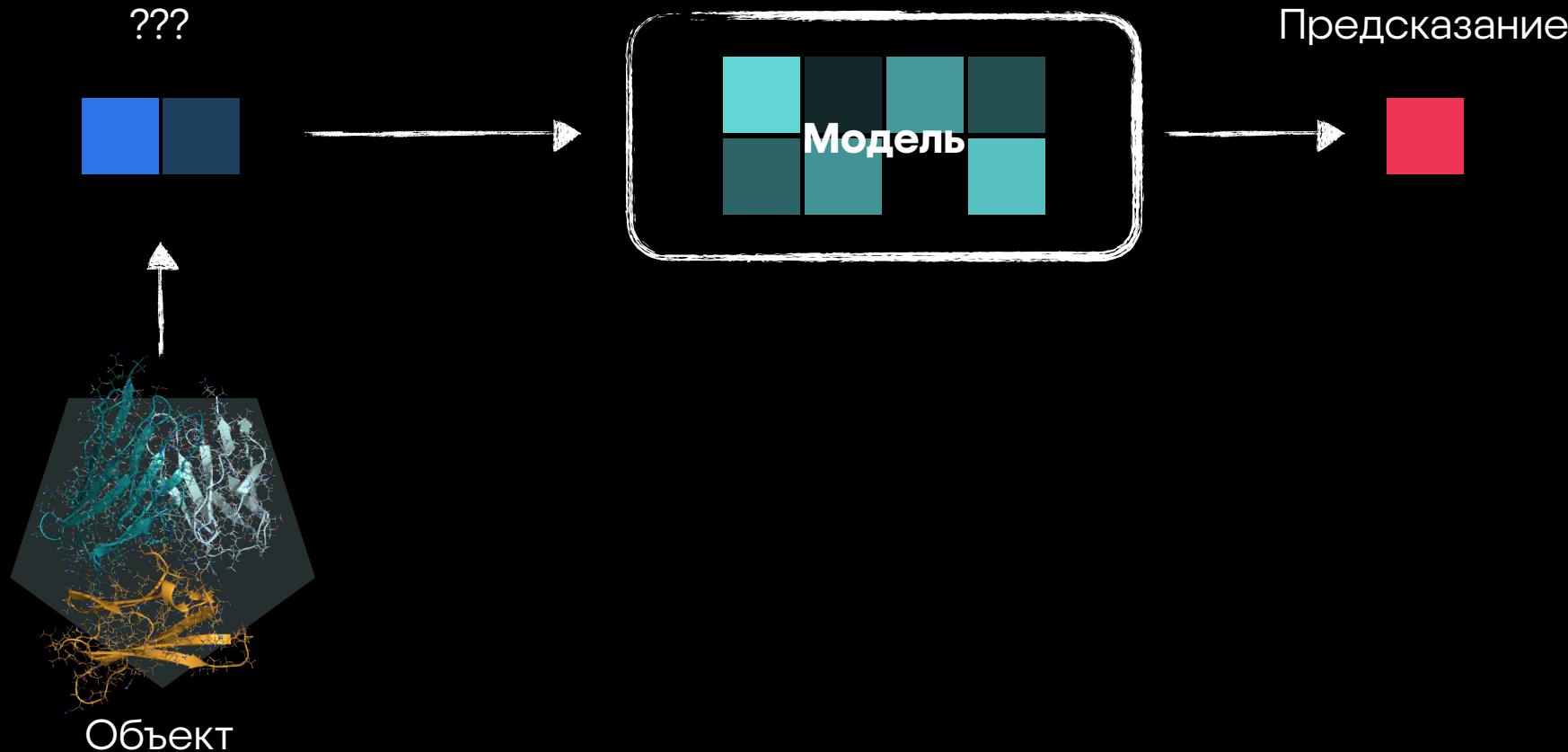
Сложные объекты



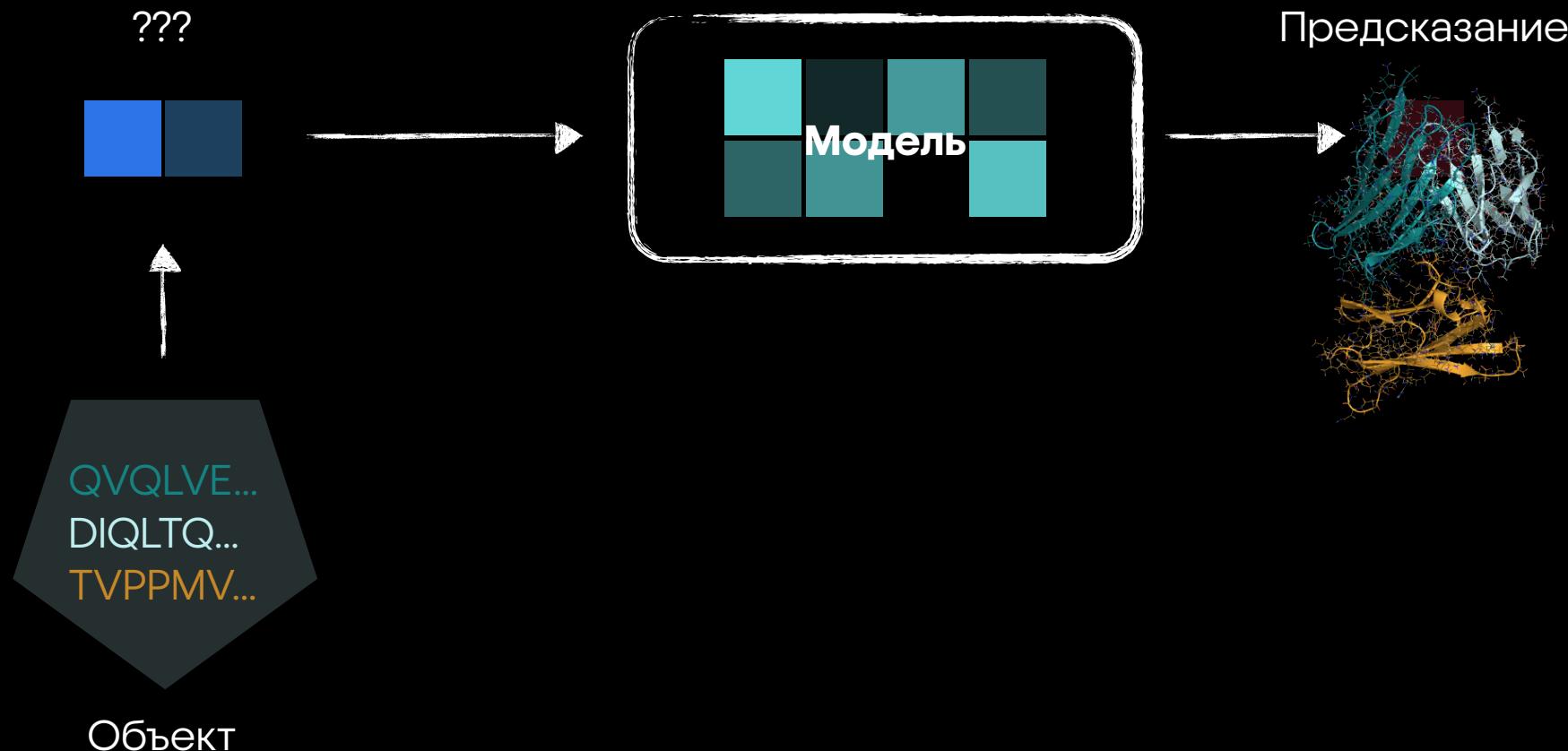
Сложные объекты



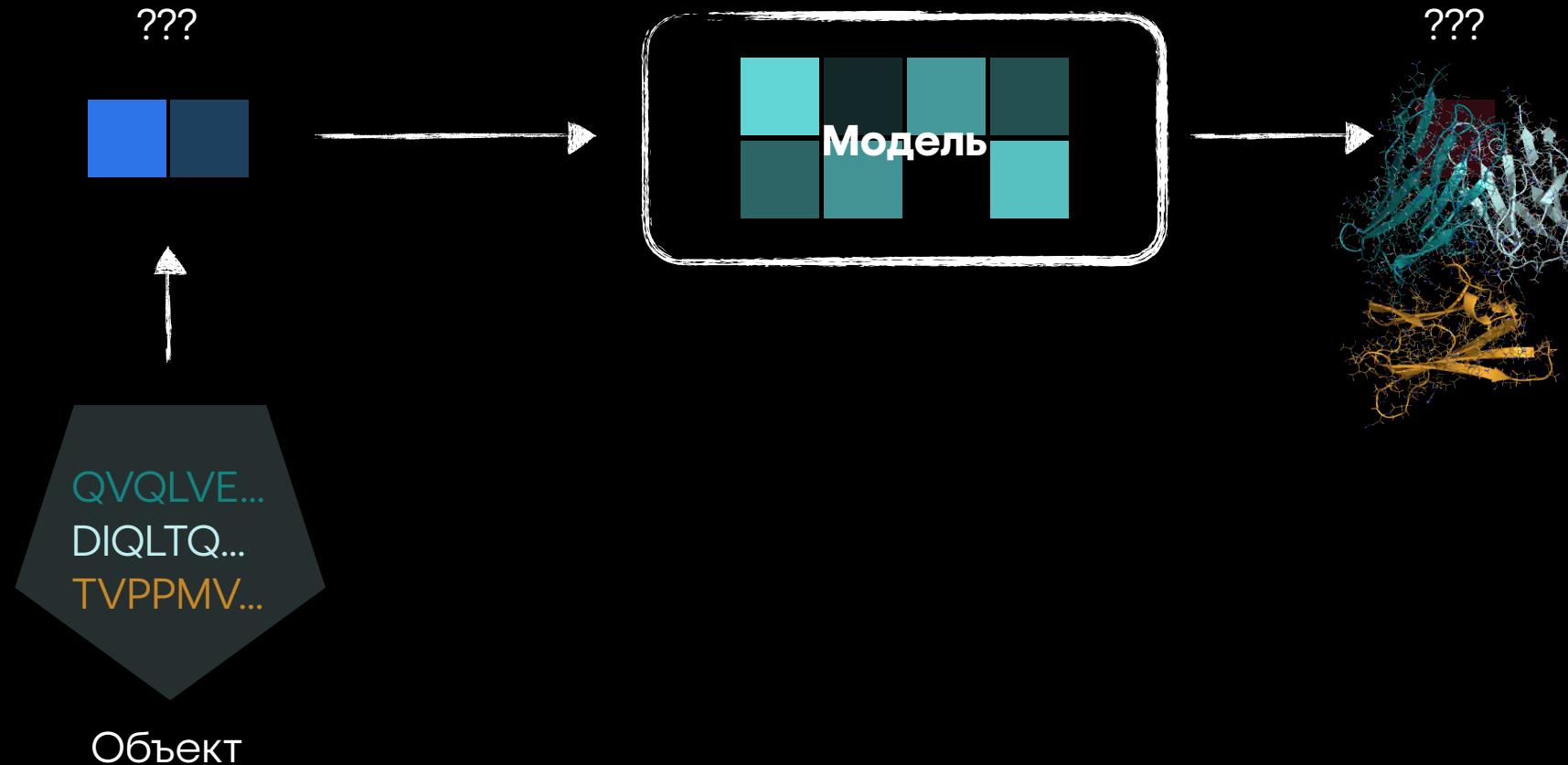
Сложные объекты



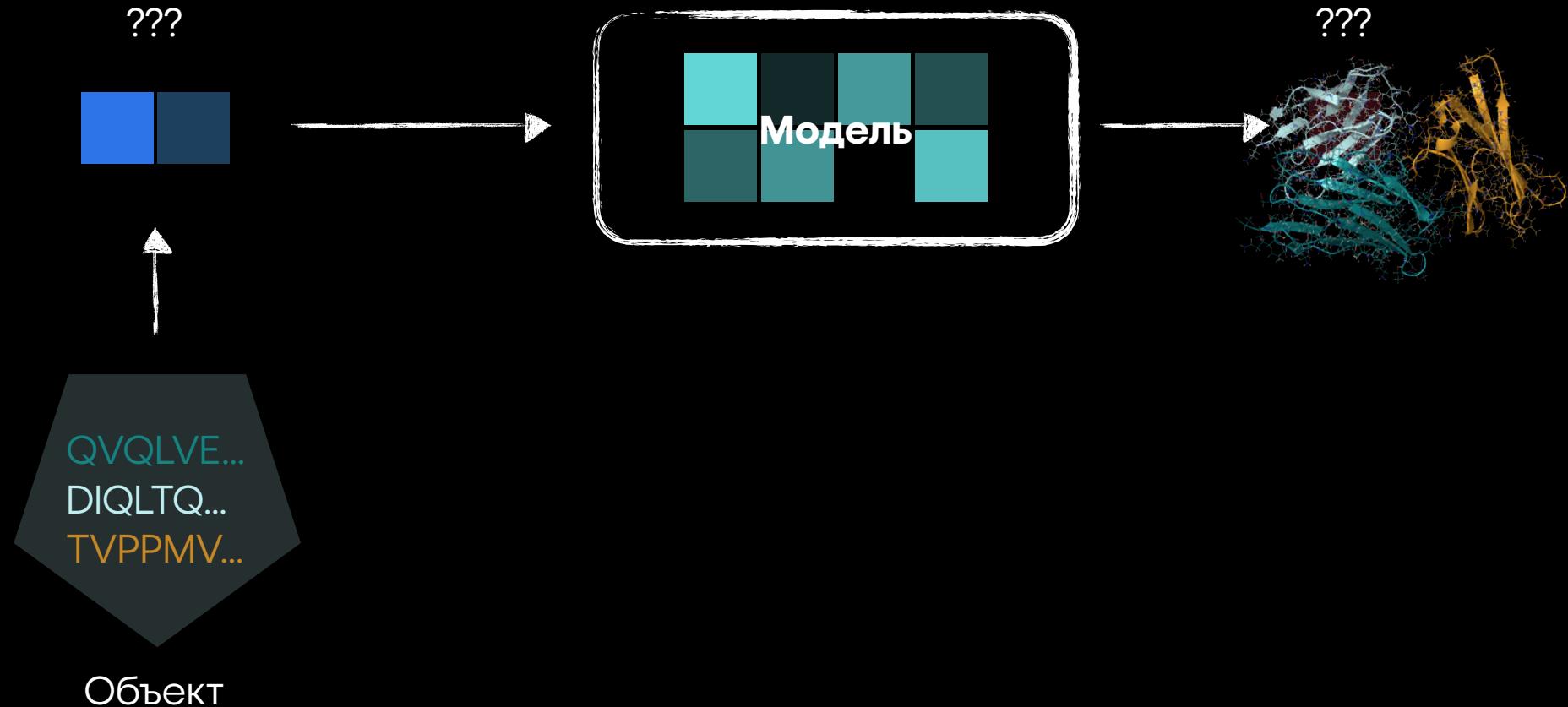
Сложные объекты



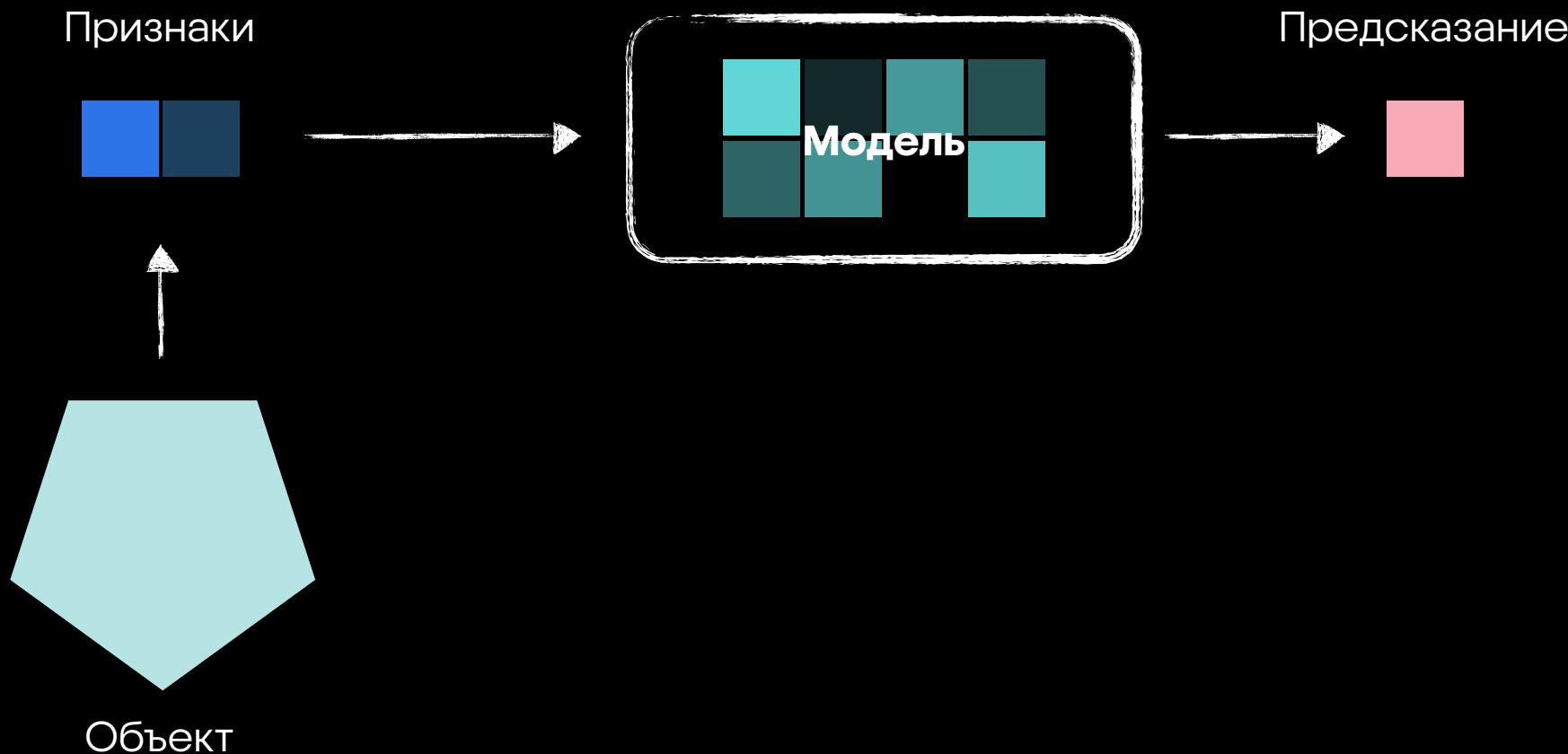
Сложные объекты



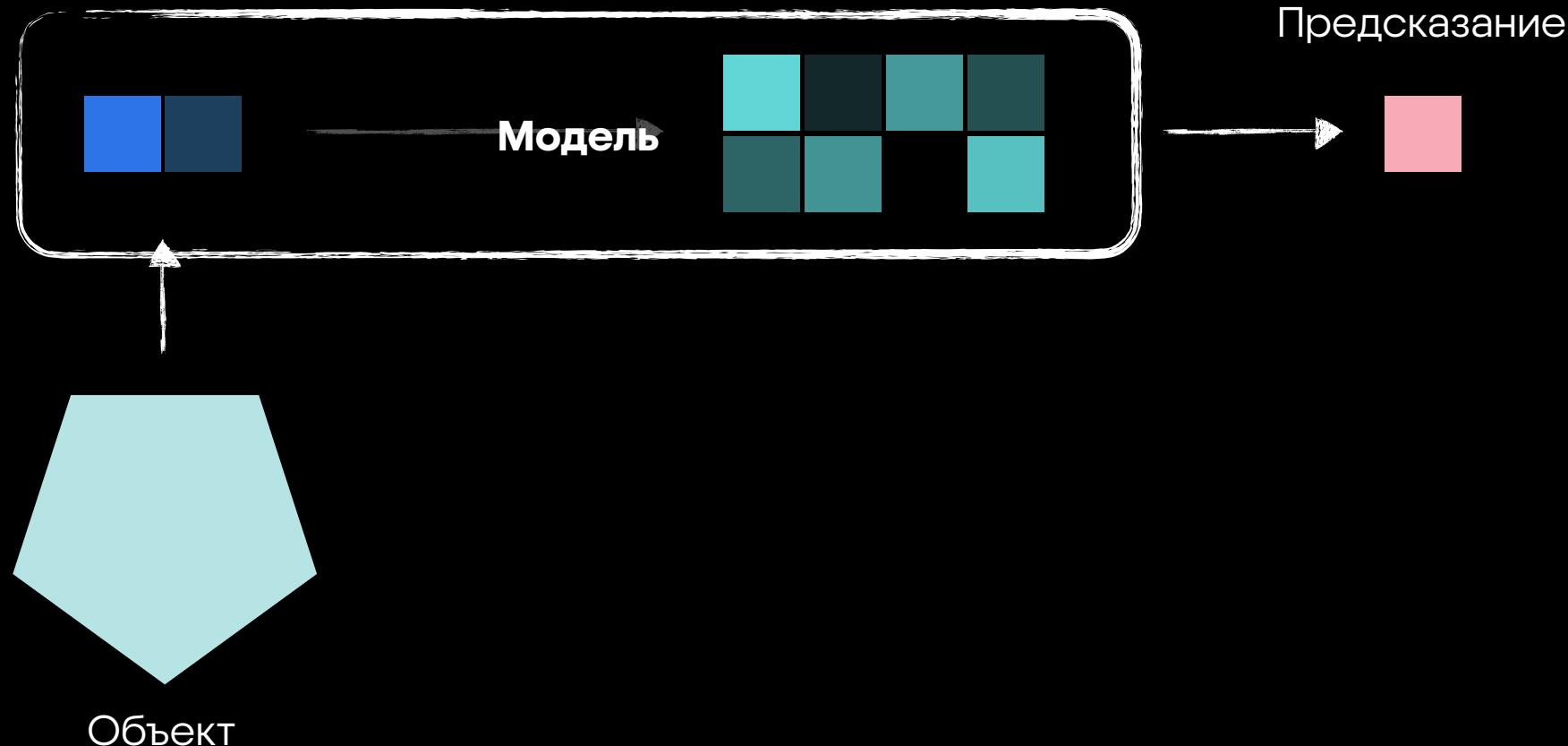
Сложные объекты



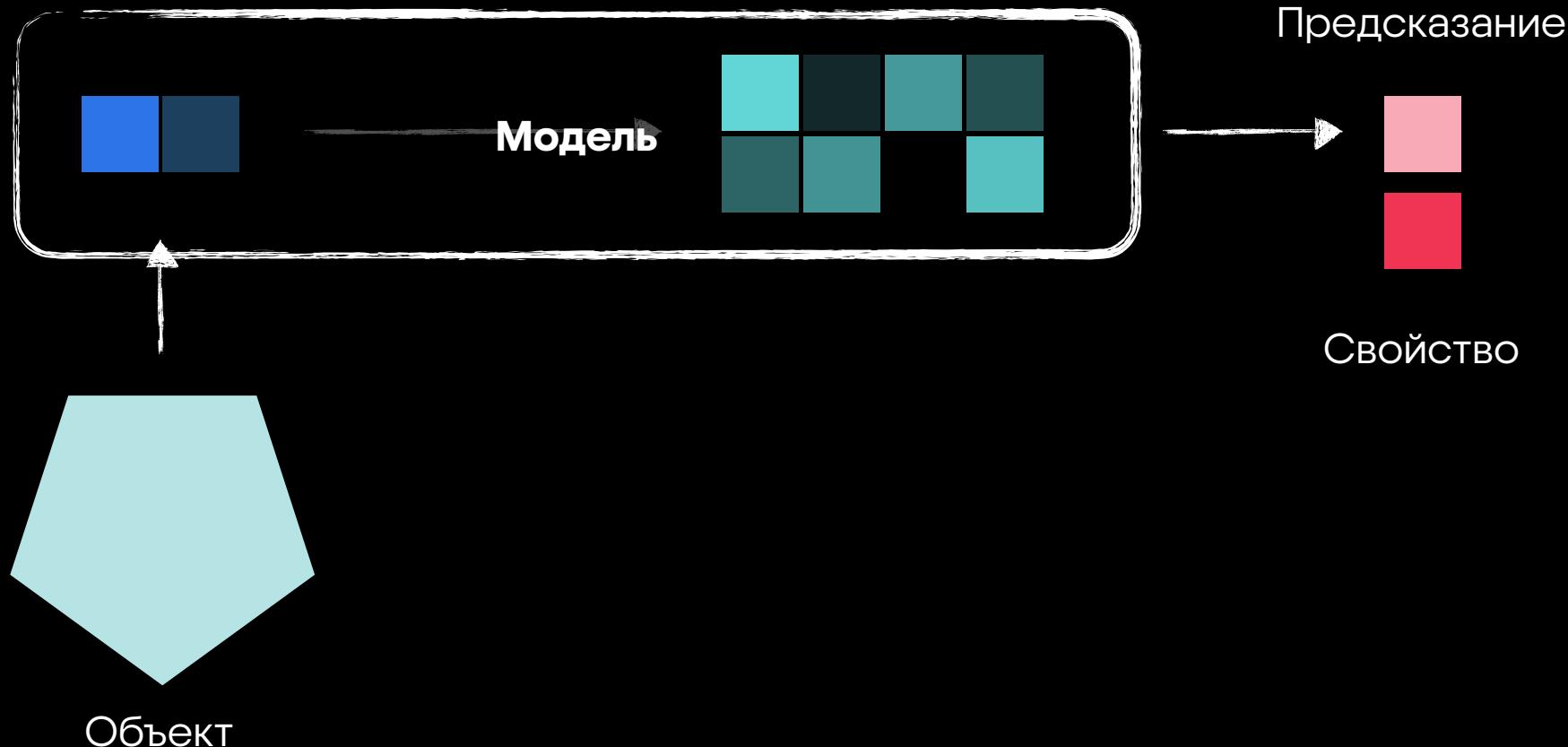
Сложные объекты



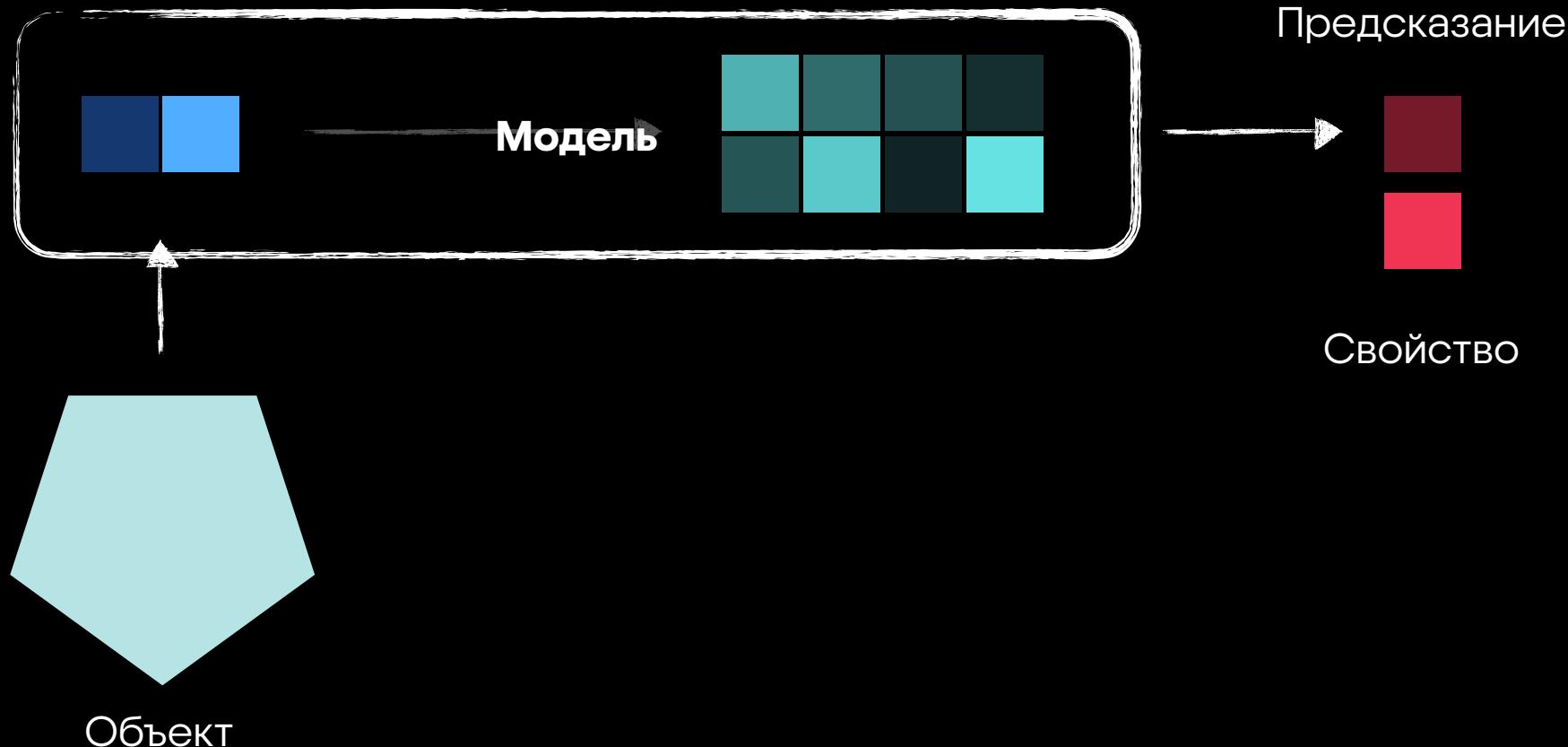
Обучение представлений данных



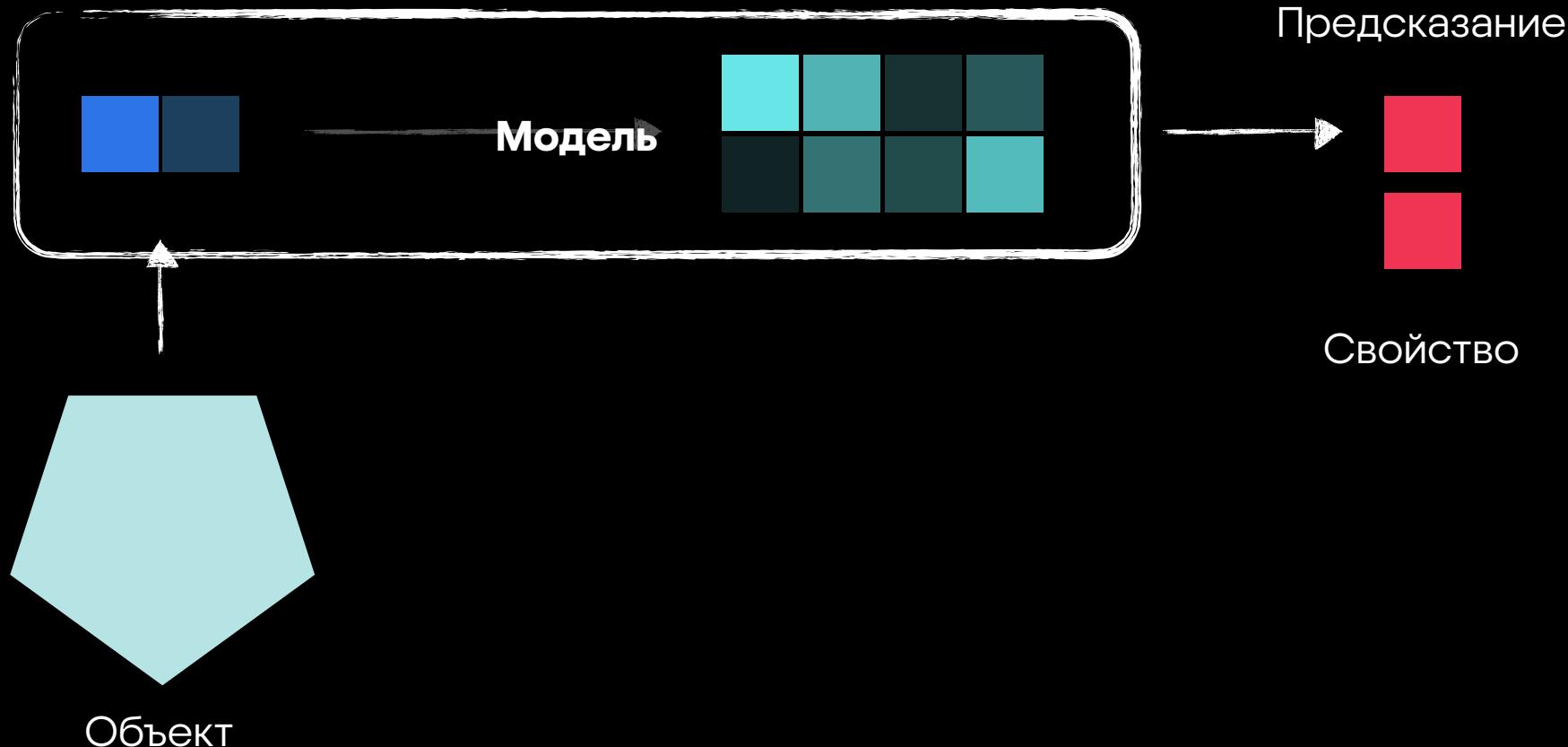
Обучение представлений данных



Обучение представлений данных



Обучение представлений данных



Обучение представлений данных



Изображения

Обучение представлений данных



QVQLVE...
DIQLTQ...
TVPPMV...

Последовательности



Изображения

Обучение представлений данных



План курса

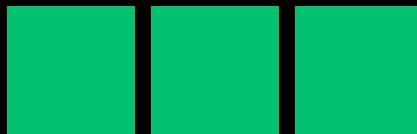
Основы нейронных сетей



Свёрточные сети
в компьютерном зрении



Рекуррентные сети и трансформеры
для последовательностей



Графовые сети



Генеративные модели
(GAN, VAE, DDPM)



Оценка за курс

8-10 домашних заданий — 100 баллов

- 50 баллов — зачёт
- 80 баллов — отлично за экзамен

О чём стоит помнить:

1. Задания занимают от 2 до 10 часов (в среднем 6-8)
2. Мягкий дедлайн — 2 недели, далее — половина баллов

Организационная информация

Основная коммуникация: Telegram-чат курса:

<https://t.me/+gE6l2CPqOcNlODAy>

Материалы и домашние задания:

<https://github.com/norsage/dl-mcs.git>



Для работы понадобятся:

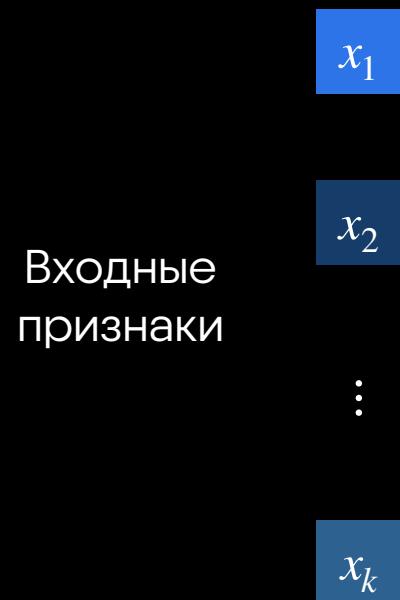
- аккаунт на GitHub — для домашних работ; создайте приватный репозиторий и добавьте <https://github.com/norsage> в collaborators (Settings -> Collaborators -> Add people)
- Google Colab

Работа на своём ПК

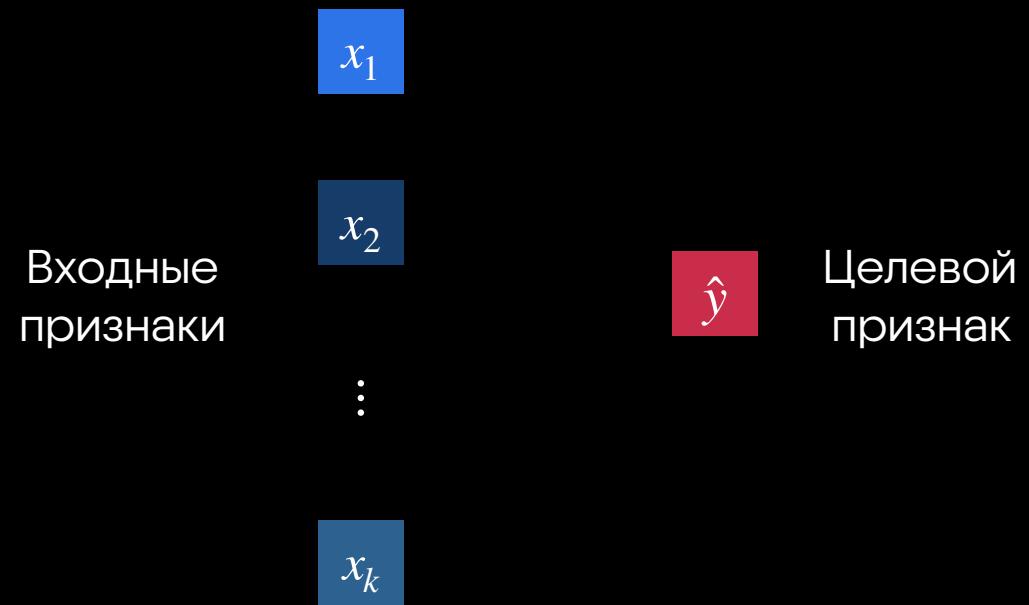
- linux / windows с WSL / macos
- python 3.13
- инструкции по установке окружения — в репозитории курса
- наличие видеокарты Nvidia или Macbook серии M позволит вам обойтись без Google Colab

1. Основы нейронных сетей

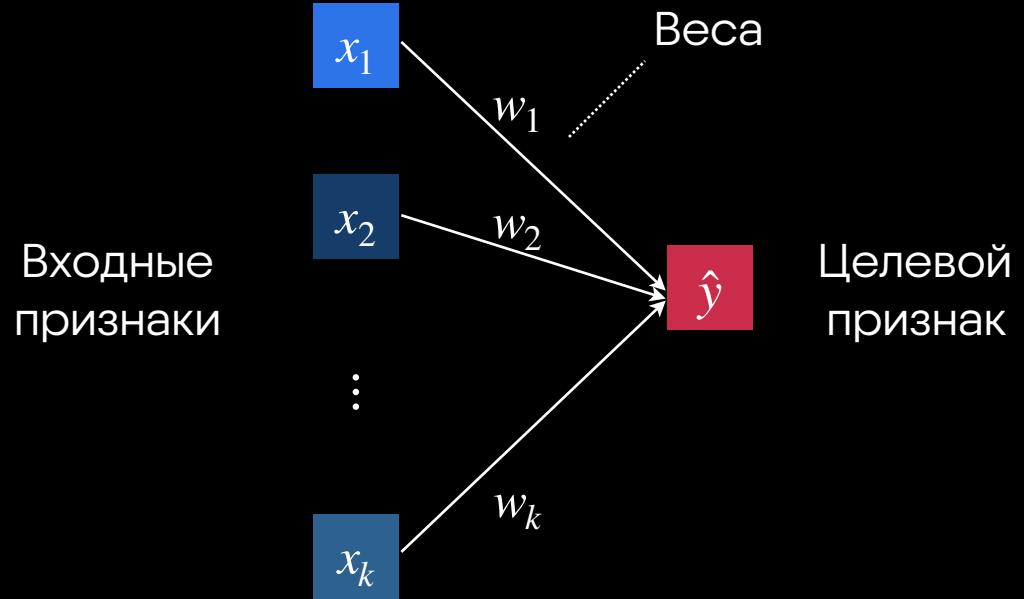
От линейной регрессии к модели нейрона



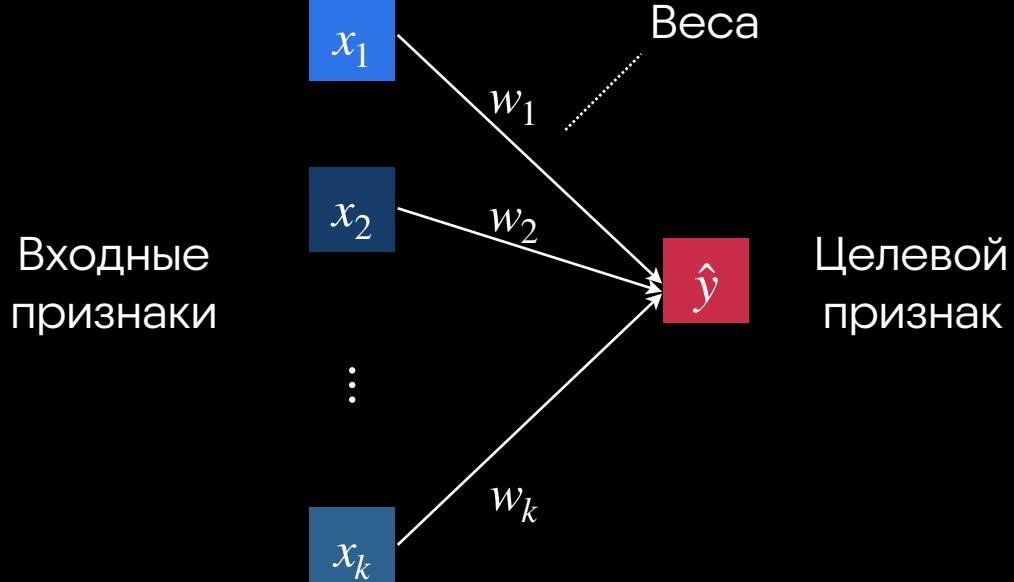
От линейной регрессии к модели нейрона



От линейной регрессии к модели нейрона



От линейной регрессии к модели нейрона

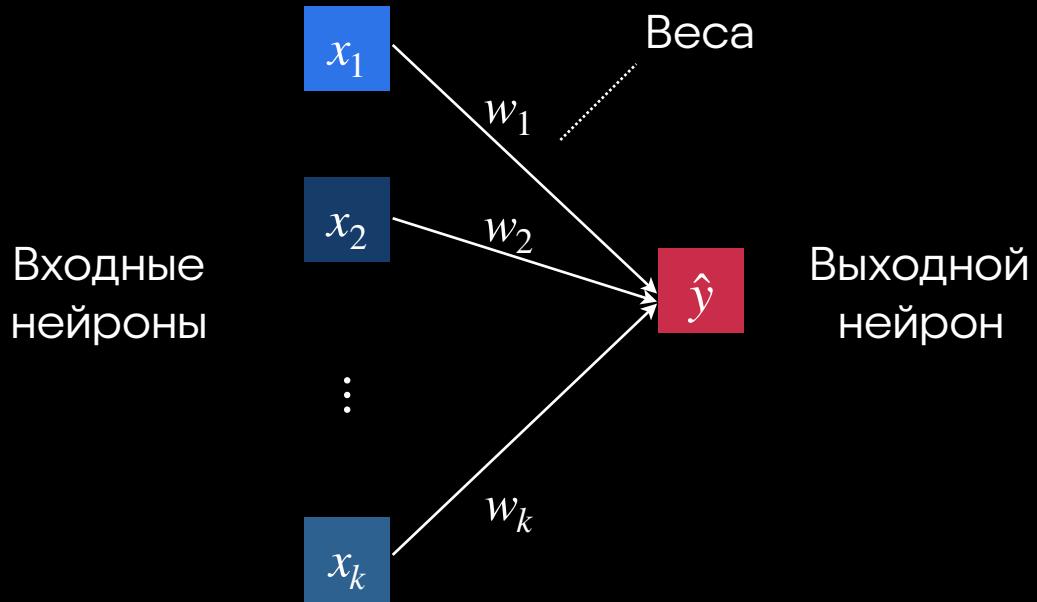


$$\hat{y} = \sum_{i=1}^k w_i x_i$$

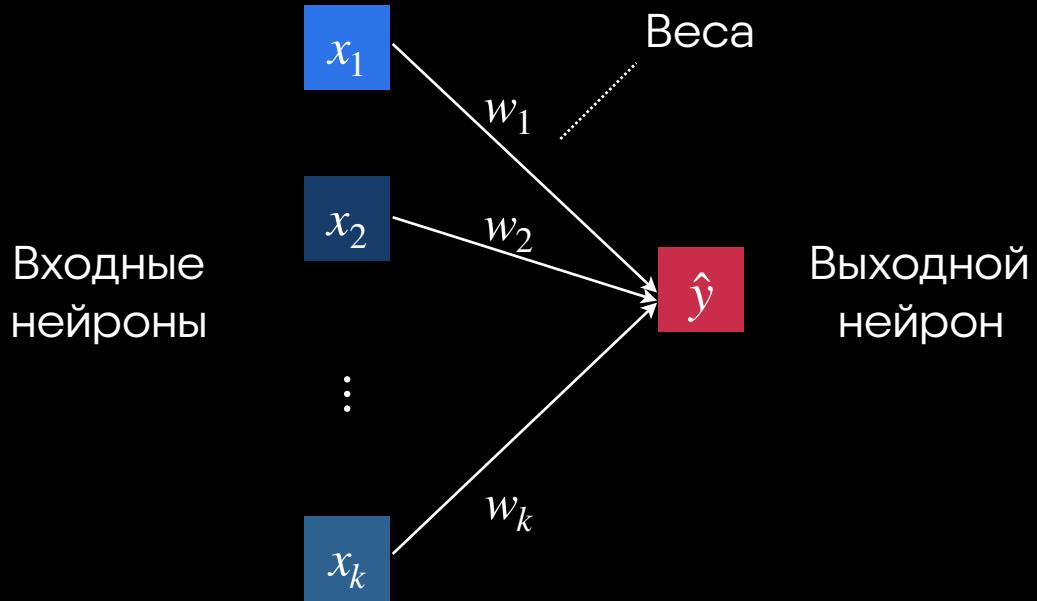
В матричной форме:

$$\hat{y} = \mathbf{w}^T \mathbf{x}$$

От линейной регрессии к модели нейрона

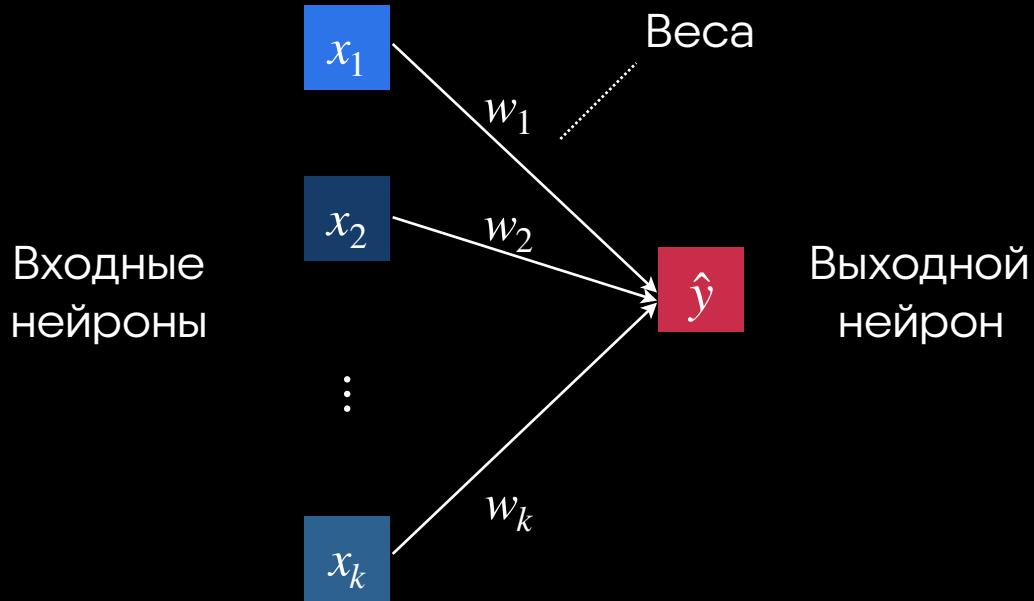


От линейной регрессии к модели нейрона



$$a = \sum_{i=1}^k w_i x_i \quad \dots \quad \text{Пре-активация}$$

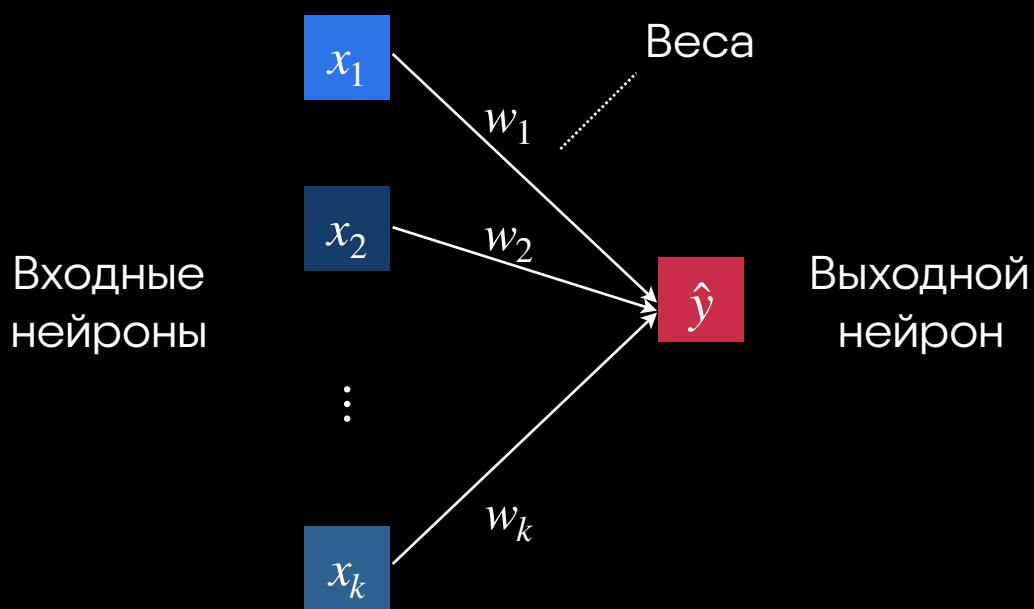
От линейной регрессии к модели нейрона



$$a = \sum_{i=1}^k w_i x_i \quad \dots \quad \text{Пре-активация}$$
$$\hat{y} = f(a) \quad \dots \quad \text{Активация}$$

↓
Функция активации

От линейной регрессии к модели нейрона

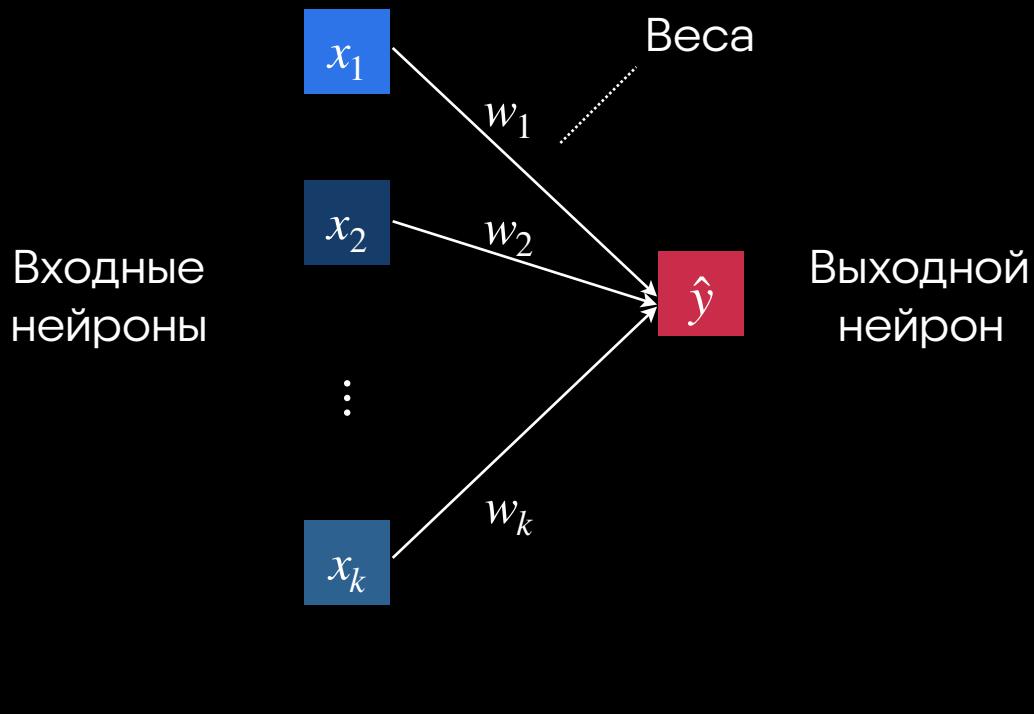


$$a = \sum_{i=1}^k w_i x_i \quad \dots \quad \text{Пре-активация}$$
$$\hat{y} = f(a) \quad \dots \quad \text{Активация}$$

↓
Функция активации

Перцептрон: $f(a) = \begin{cases} 1, & a > 0 \\ 0, & a = 0 \\ -1, & a < 0 \end{cases}$

От линейной регрессии к модели нейрона



$$a = \sum_{i=1}^k w_i x_i \quad \dots \quad \text{Пре-активация}$$
$$\hat{y} = f(a) \quad \dots \quad \text{Активация}$$

Функция активации

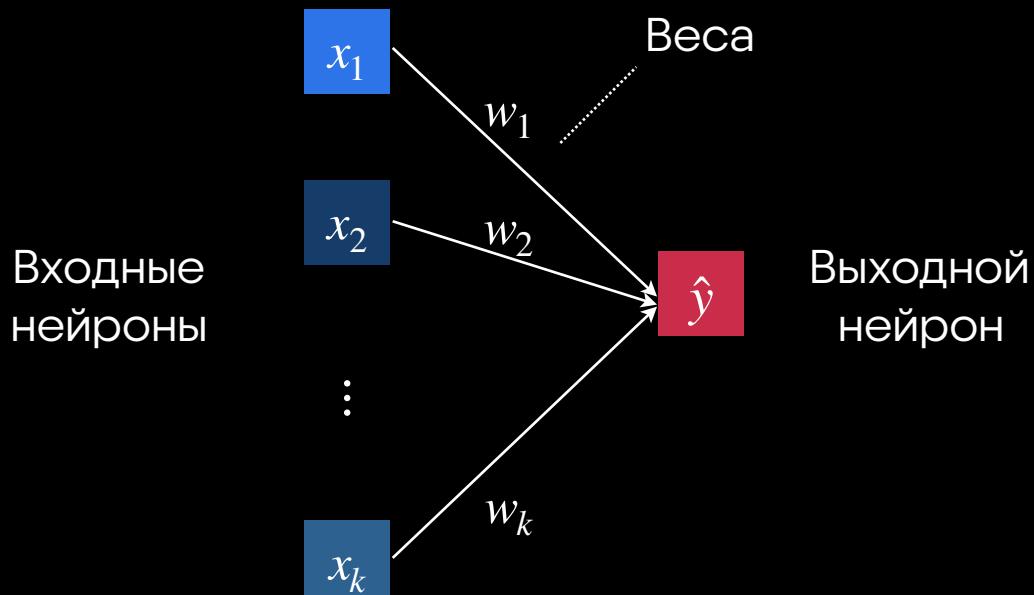
Перцептрон:

$$f(a) = \begin{cases} 1, & a > 0 \\ 0, & a = 0 \\ -1, & a < 0 \end{cases}$$

Линейная регрессия:

$$f(a) = a$$

От линейной регрессии к модели нейрона



$$a = \sum_{i=1}^k w_i x_i \quad \dots \quad \text{Пре-активация}$$

$$\hat{y} = f(a) \quad \dots \quad \text{Активация}$$

Функция активации

Перцептрон:

$$f(a) = \begin{cases} 1, & a > 0 \\ 0, & a = 0 \\ -1, & a < 0 \end{cases}$$

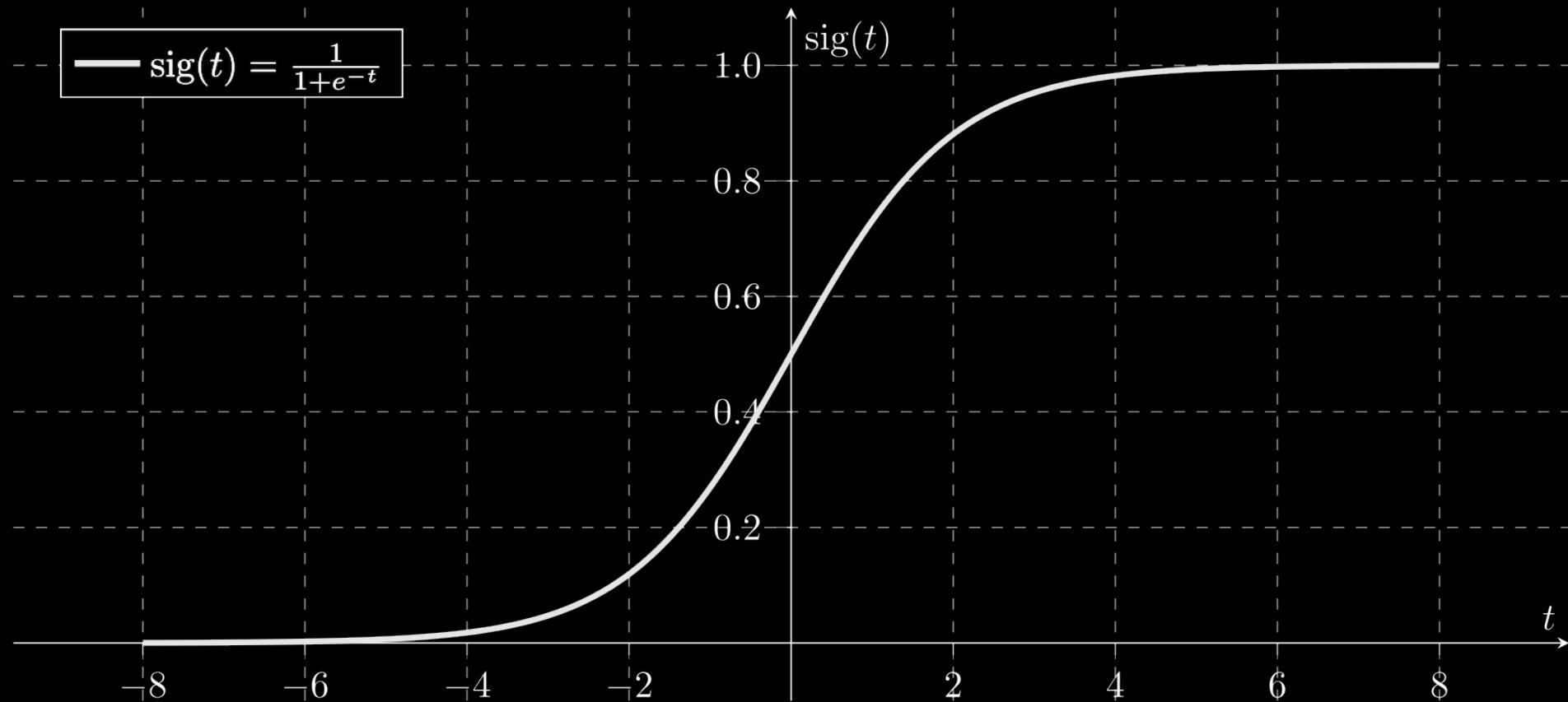
Линейная
регрессия:

$$f(a) = a$$

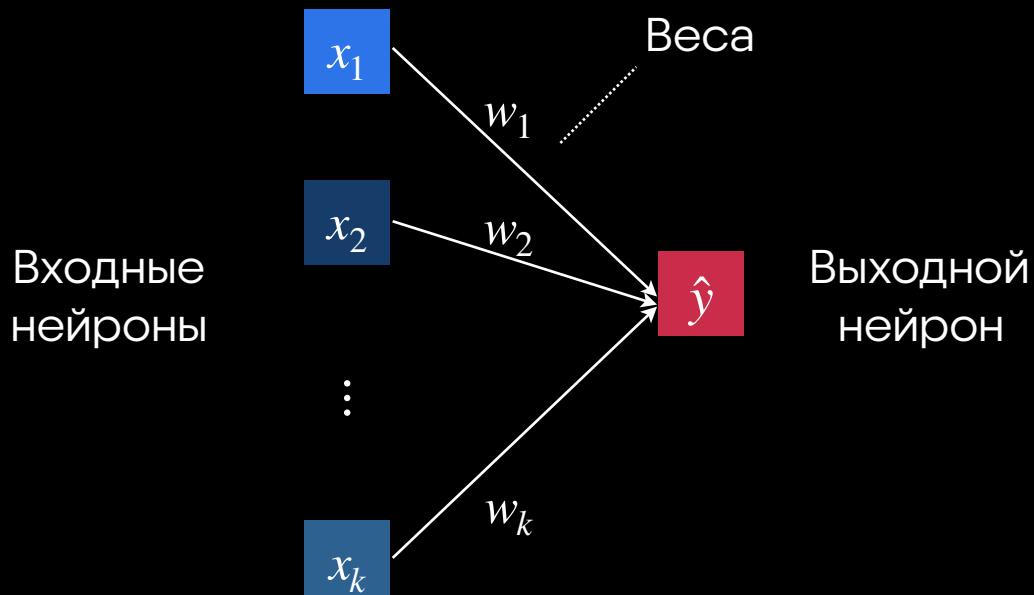
Логистическая
регрессия:

$$f(a) = \sigma(a) = \frac{1}{1 + \exp(-a)}$$

От линейной регрессии к модели нейрона



От линейной регрессии к модели нейрона



$$a = \sum_{i=1}^k w_i x_i \quad \dots \quad \text{Пре-активация}$$
$$\hat{y} = f(a) \quad \dots \quad \text{Активация}$$

↓
Функция активации

Перцептрон:

$$f(a) = \begin{cases} 1, & a > 0 \\ 0, & a = 0 \\ -1, & a < 0 \end{cases}$$

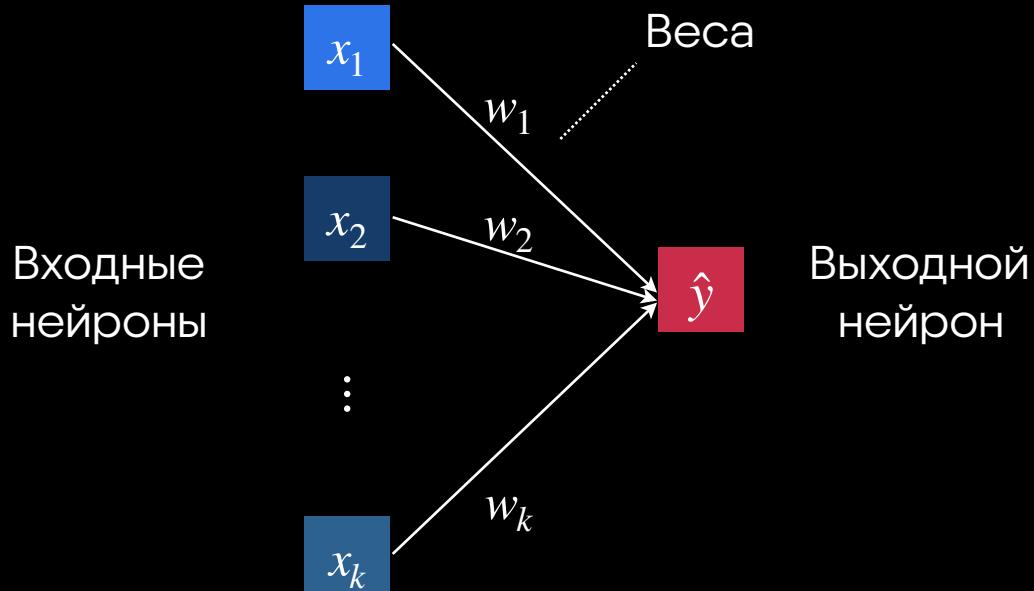
Линейная
регрессия:

$$f(a) = a$$

Логистическая
регрессия:

$$f(a) = \sigma(a) = \frac{1}{1 + \exp(-a)}$$

От линейной регрессии к модели нейрона



$$a = \sum_{i=1}^k w_i x_i \quad \dots \quad \text{Пре-активация}$$
$$\hat{y} = f(a) \quad \dots \quad \text{Активация}$$

↓
Функция активации

В матричной форме:

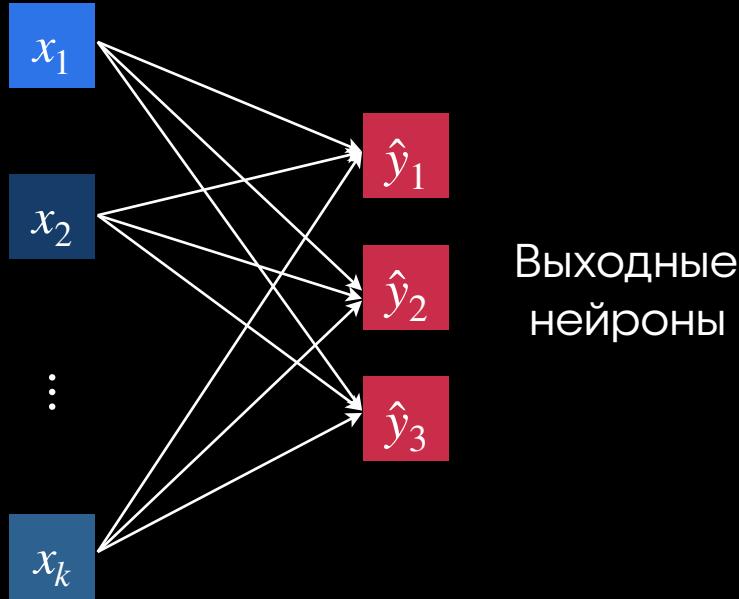
$$\hat{y} = f(\mathbf{w}^T \mathbf{x})$$

Если классов больше двух?



Если классов больше двух?

Входные
нейроны

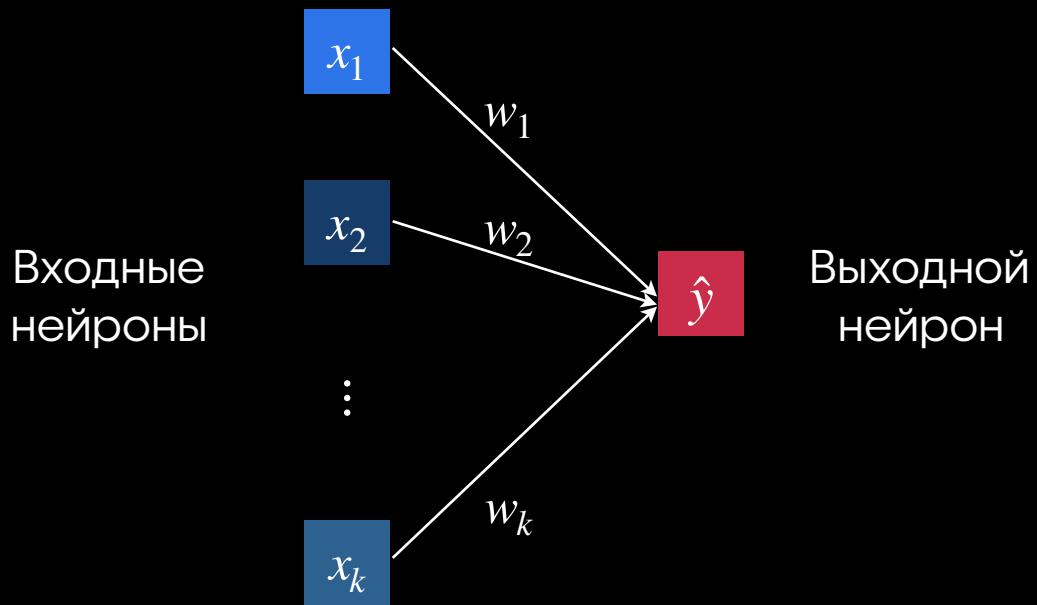


$$a_j = \sum_{i=1}^k w_{ji}x_i \quad \dots \quad \text{Пре-активация (logits)}$$

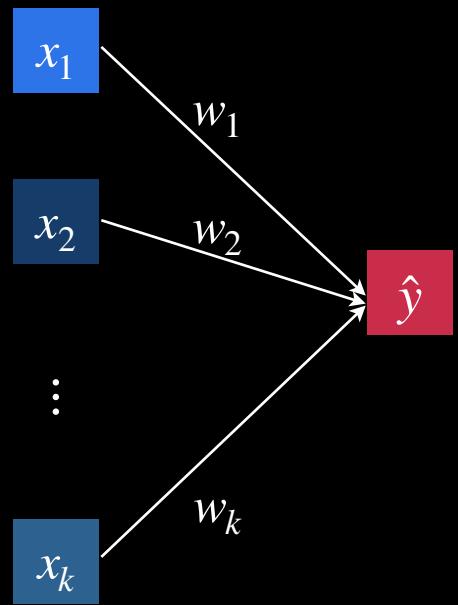
$$\hat{y}_j = \text{softmax}(\mathbf{a})_j \quad \dots \quad \text{Активация}$$

$$\text{softmax}(\mathbf{a})_j = \frac{\exp(a_j)}{\sum_k \exp(a_k)}$$

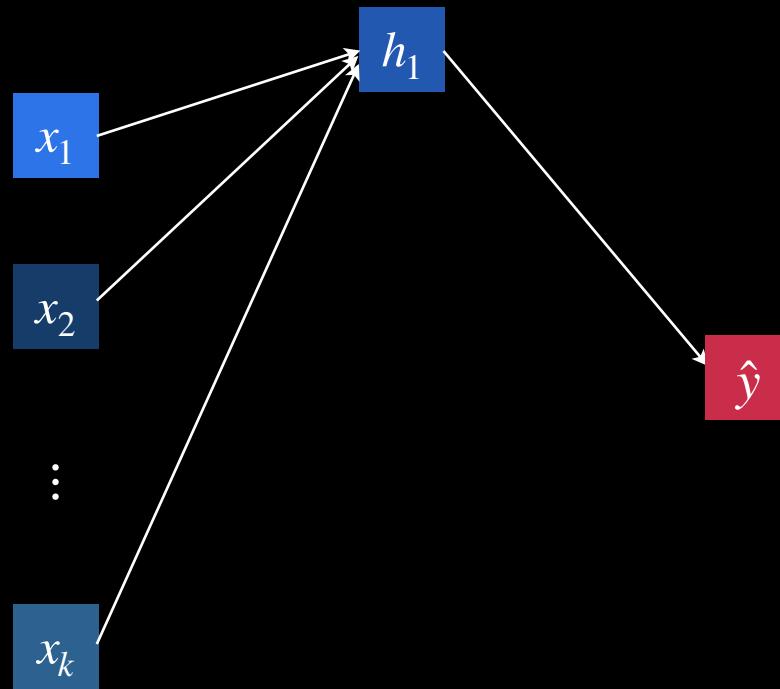
Однослойная нейронная сеть



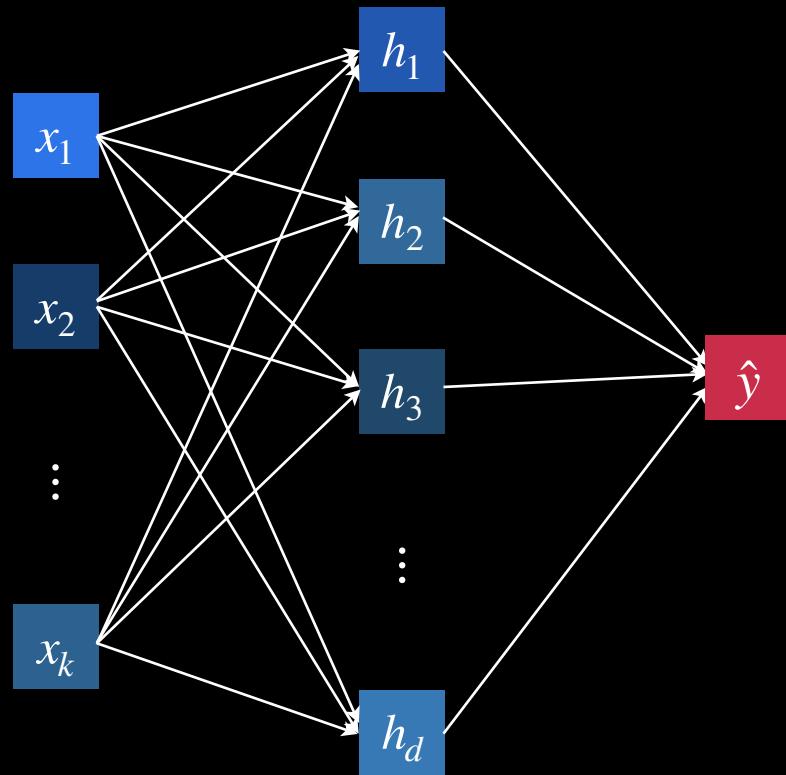
Однослойная нейронная сеть



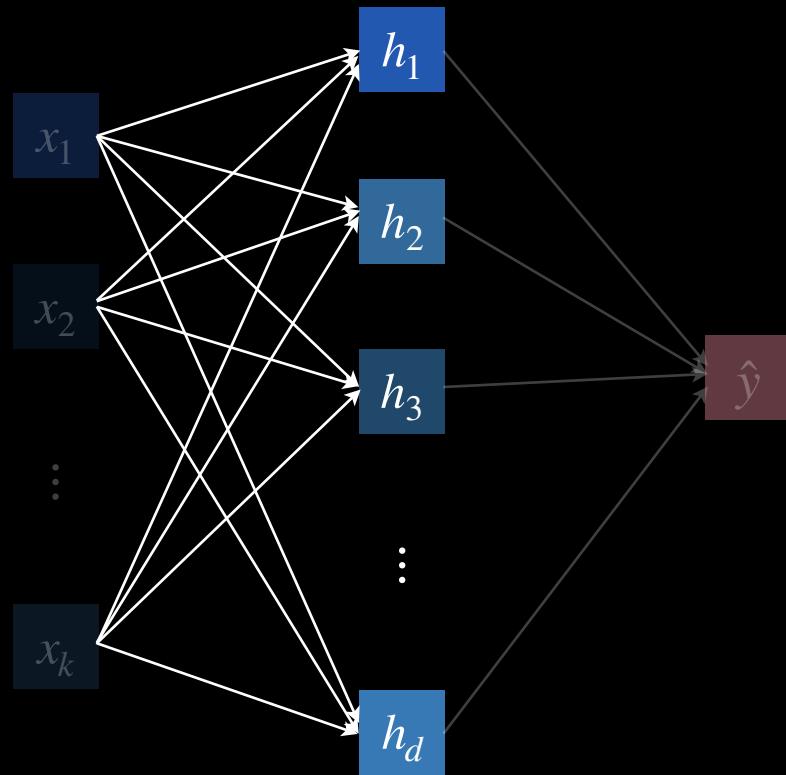
Добавление скрытого слоя



Добавление скрытого слоя

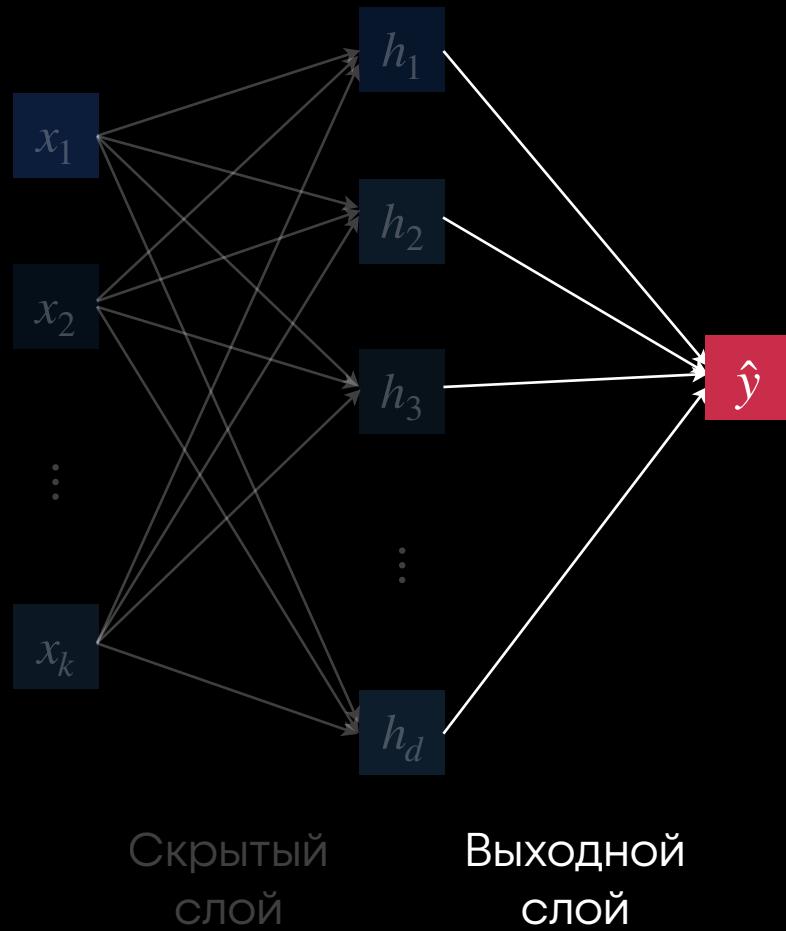


Добавление скрытого слоя

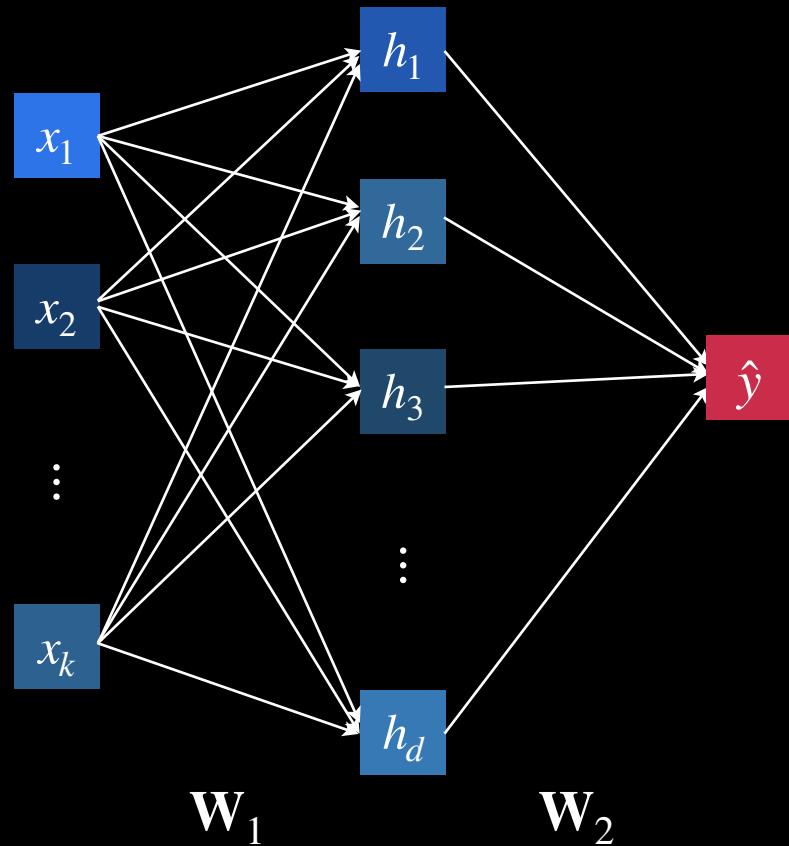


Скрытый
слой

Добавление скрытого слоя



Добавление скрытого слоя



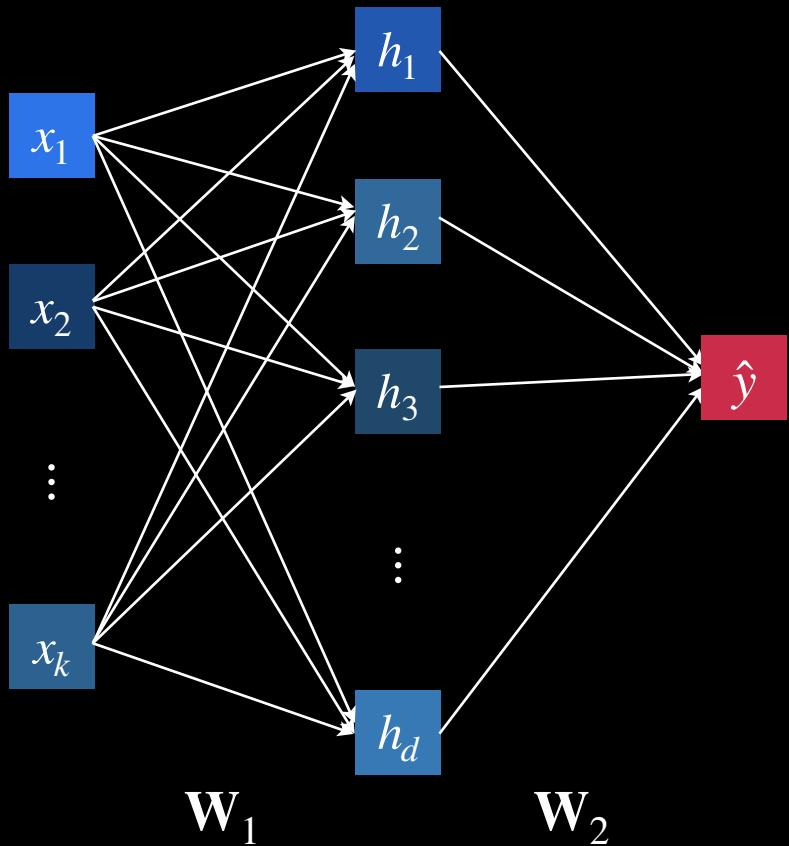
$$h_j = f_h \left(\sum_{i=1}^k w_{ji}^{(1)} x_i \right) \quad \text{Скрытые активации}$$
$$\hat{y}_j = f_o \left(\sum_{i=1}^d w_{ji}^{(2)} h_i \right) \quad \text{Выход сети}$$

В матричной форме:

$$\mathbf{h} = f_h (\mathbf{W}_1 \mathbf{x})$$

$$\hat{\mathbf{y}} = f_o (\mathbf{W}_2 \mathbf{h})$$

Добавление скрытого слоя



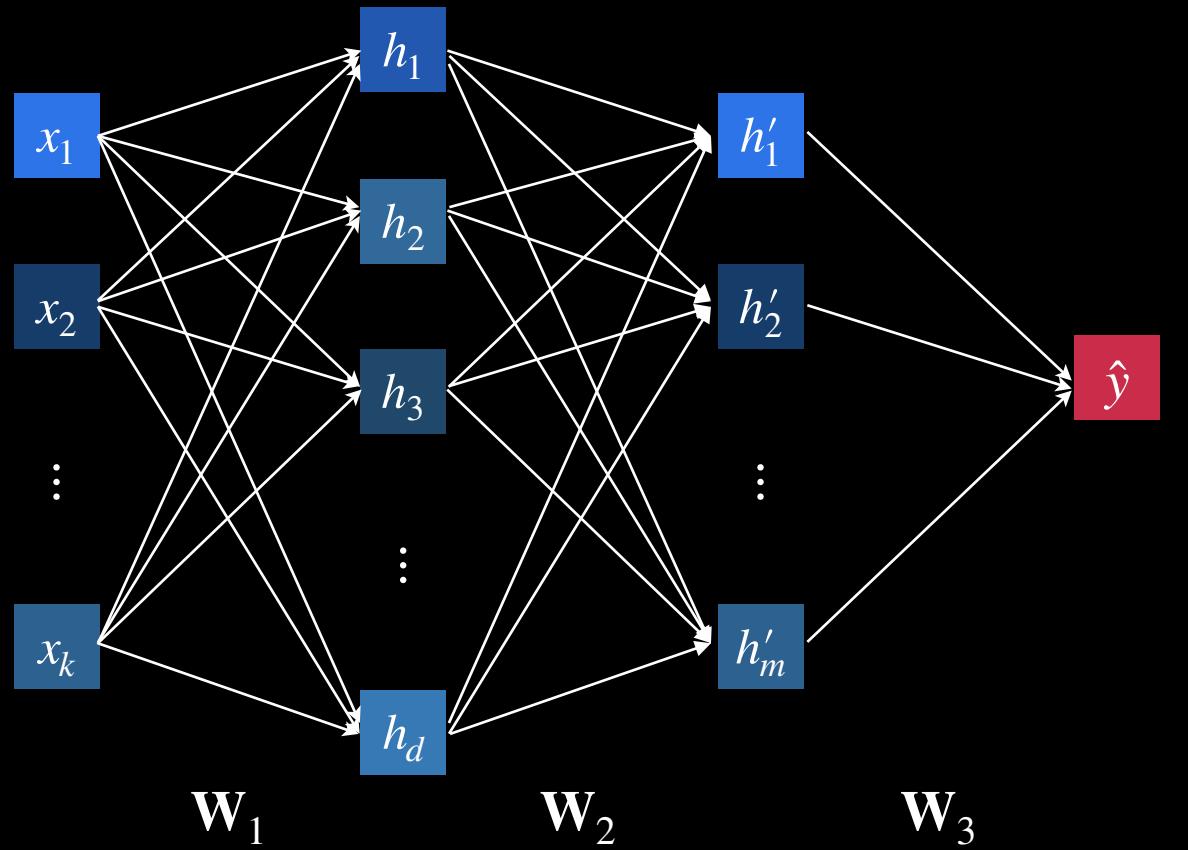
В матричной форме:

$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

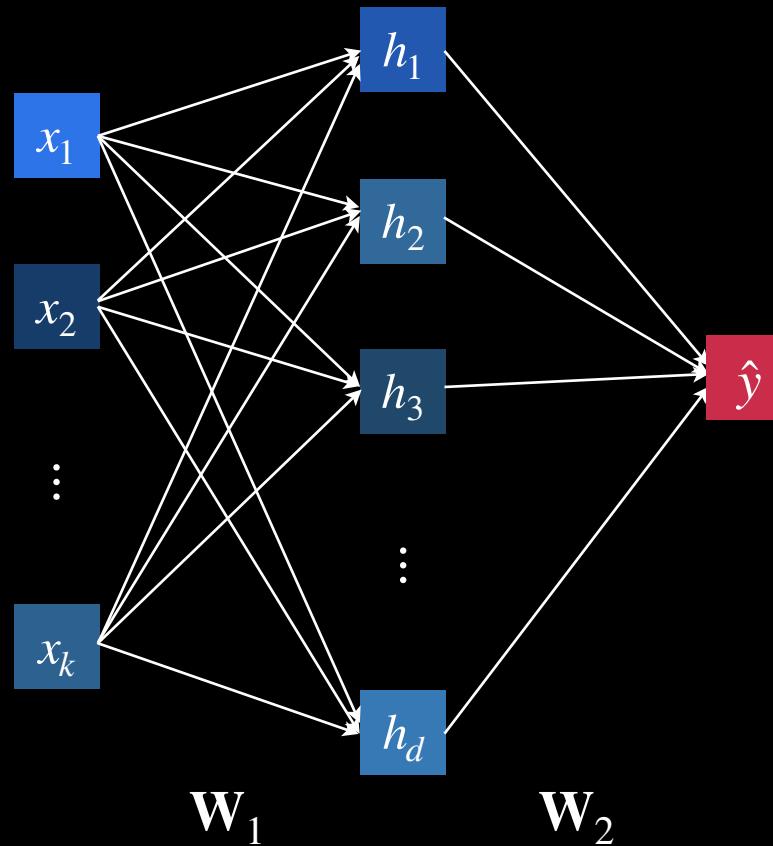
$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

Для некоторых нелинейных
функций активации f_h
(например, для сигмоиды)
такая сеть —
универсальный аппроксиматор

Добавление скрытого слоя



Добавление скрытого слоя



В матричной форме:

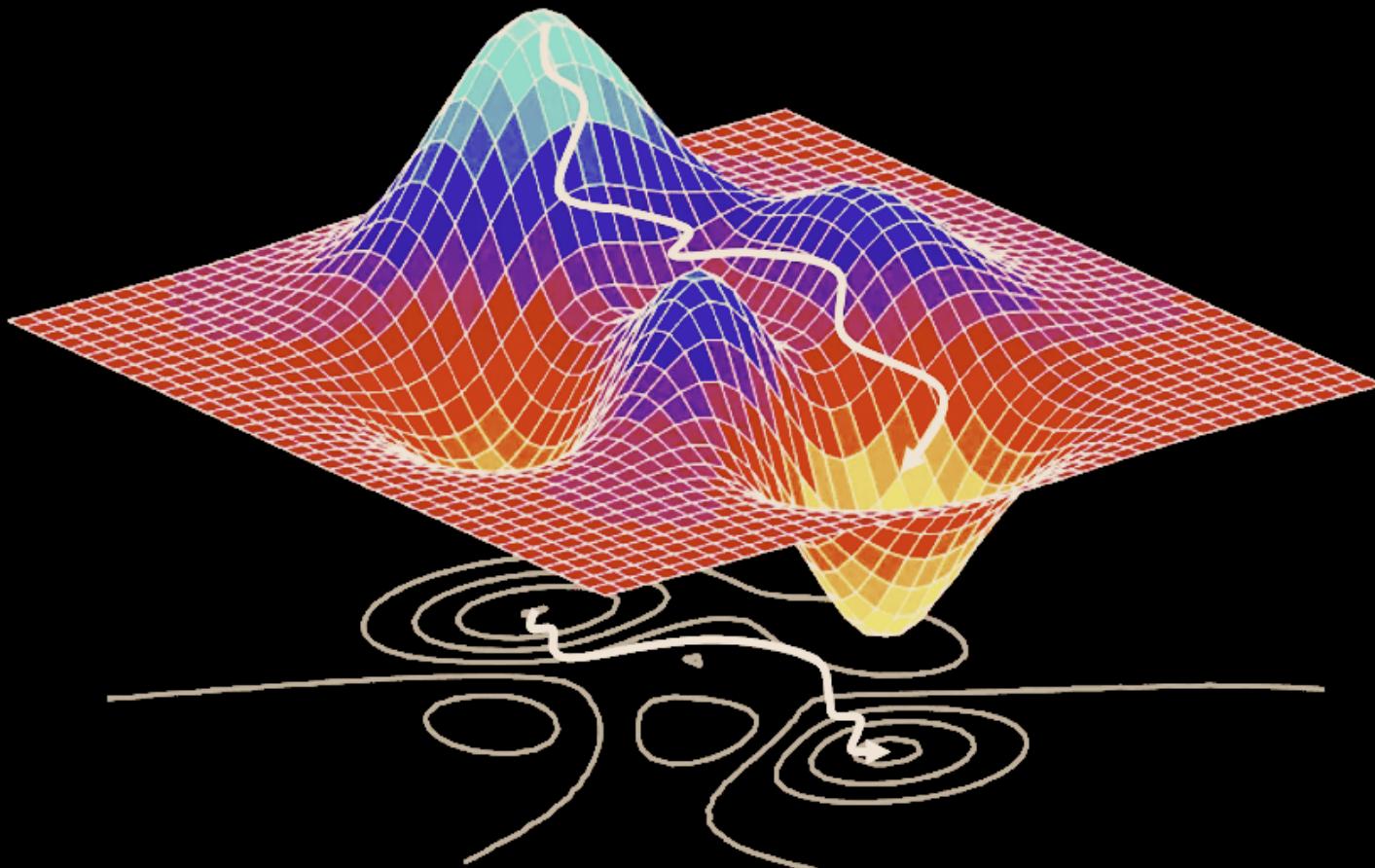
$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

Как такое учить?

Производная и градиент

Производная и градиент



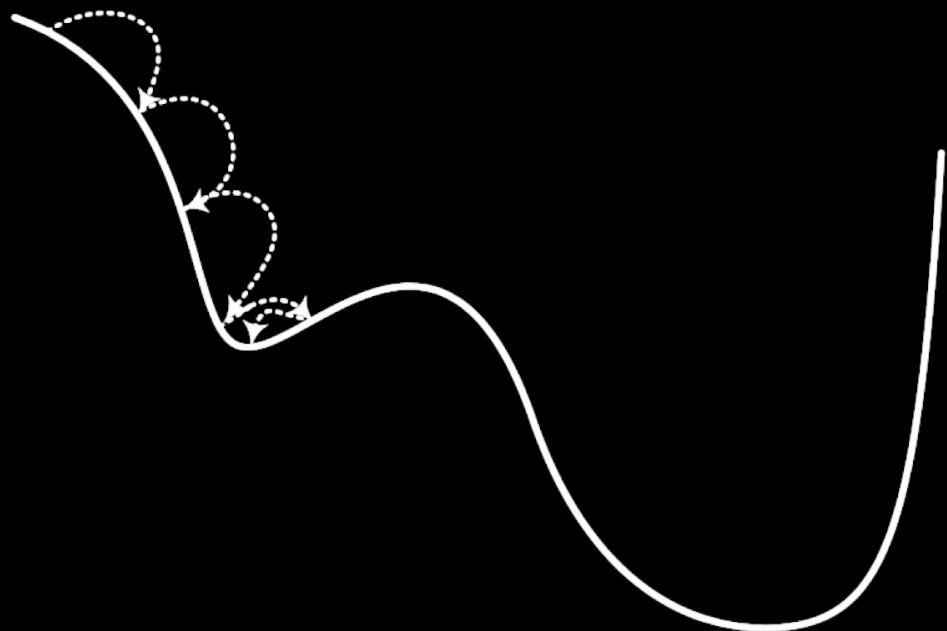
$E(\mathbf{W})$ — функция, которую хотим минимизировать

Градиентный спуск:

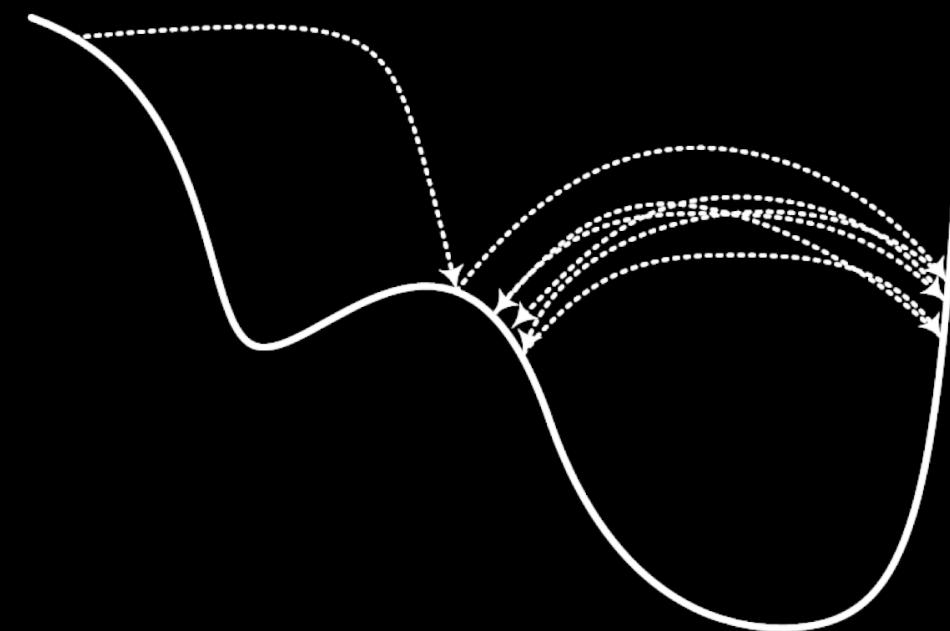
$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \nabla E \quad \text{Градиент}$$

Размер шага

Выбор шага оптимизации

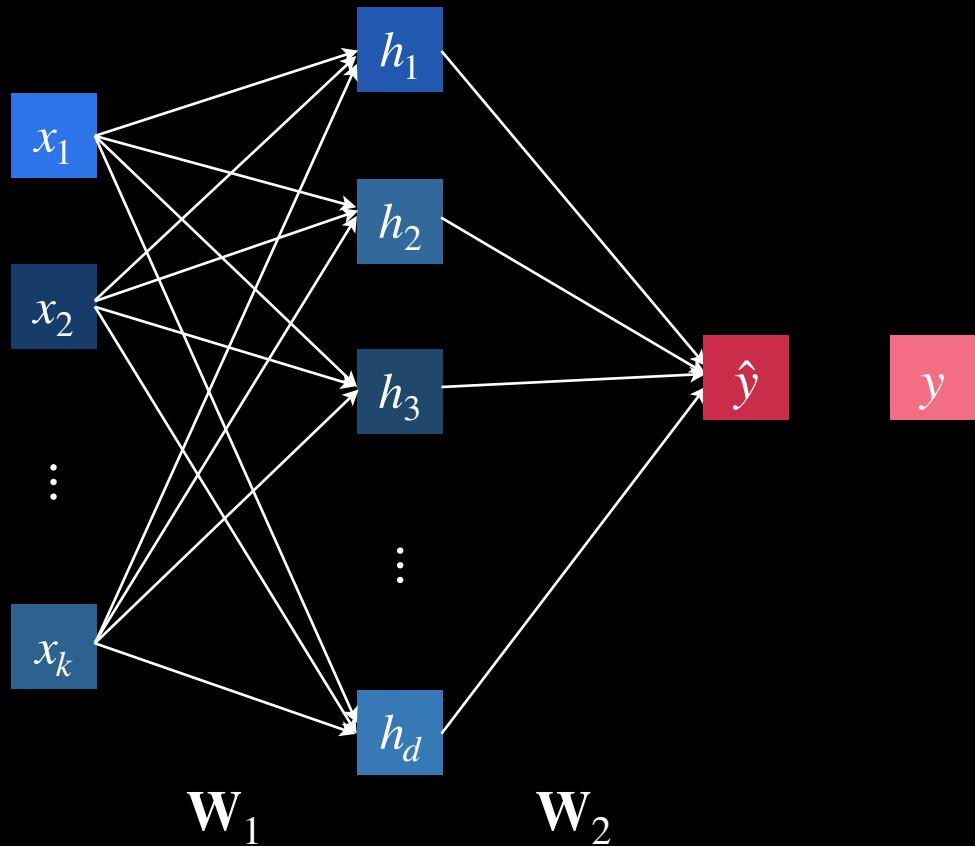


(а)



(б)

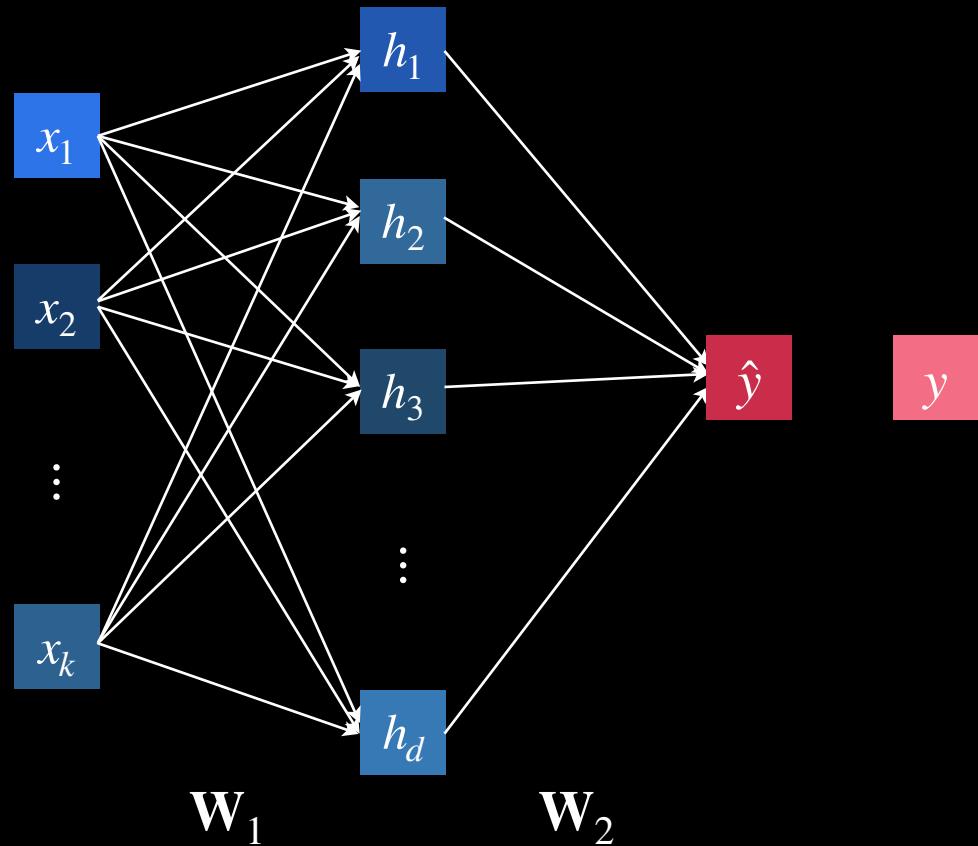
Обучение нейронных сетей



$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

Обучение нейронных сетей



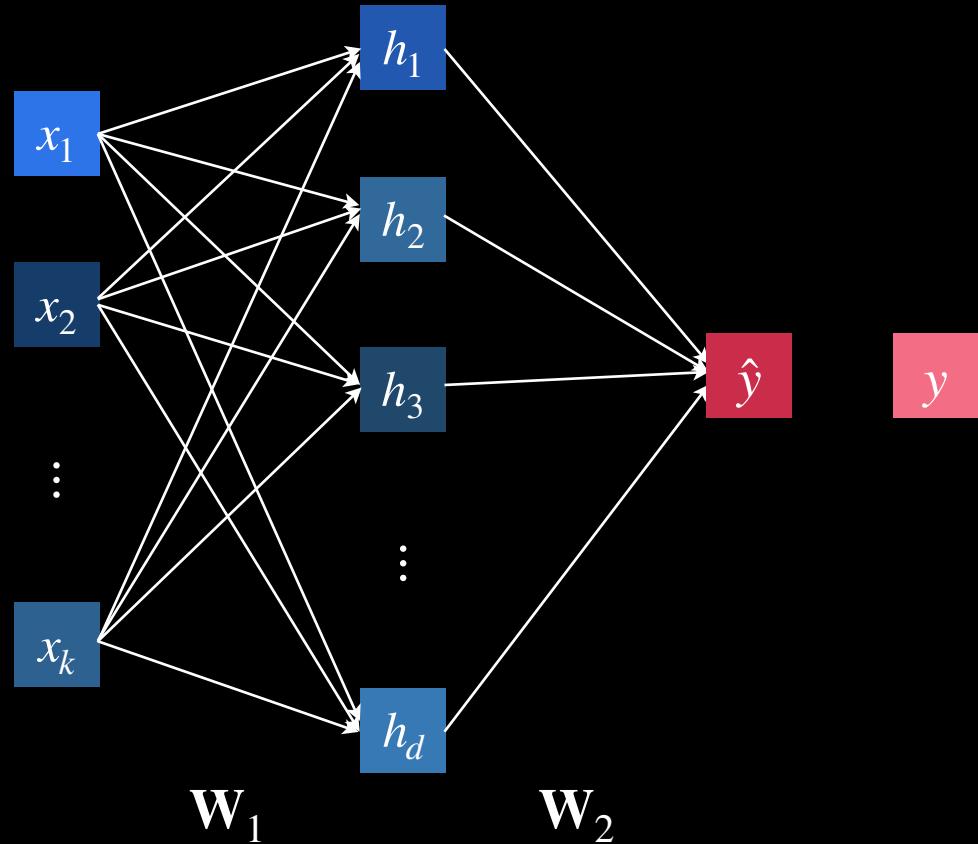
$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

$$E(\hat{y}, y)$$

Функция ошибки

Обучение нейронных сетей



$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

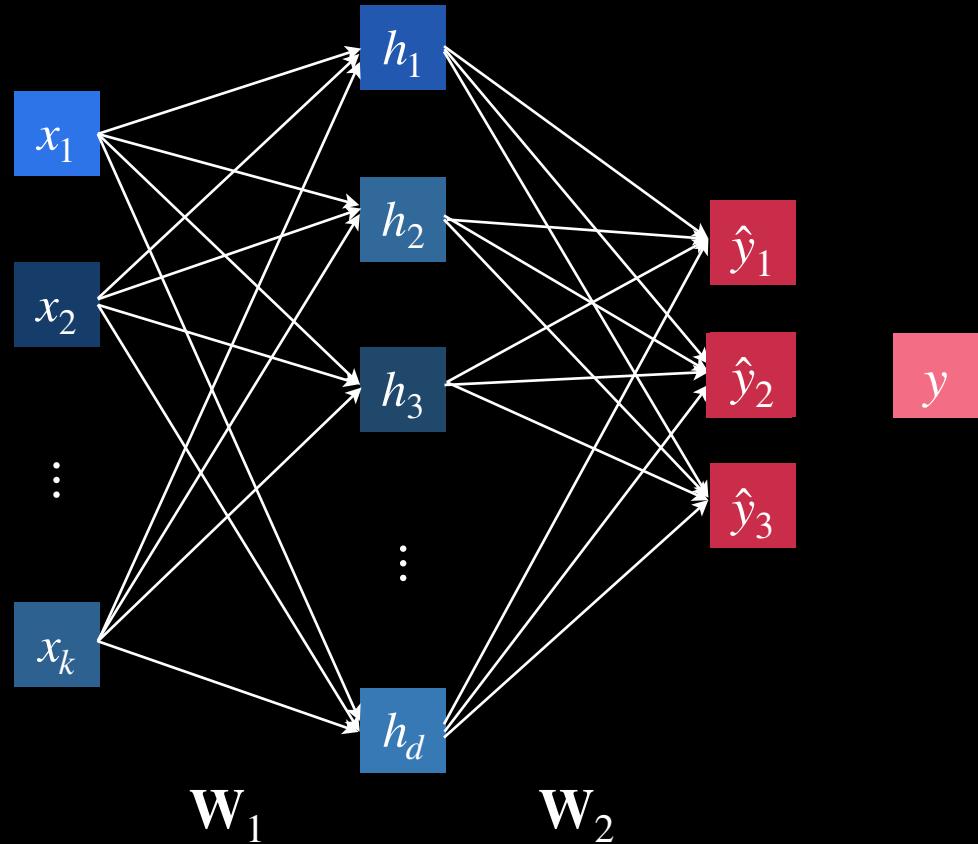
$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

$E(\hat{y}, y)$ Функция ошибки

Для задачи регрессии:

$$E(\hat{y}, y) = (\hat{y} - y)^2$$

Обучение нейронных сетей



$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

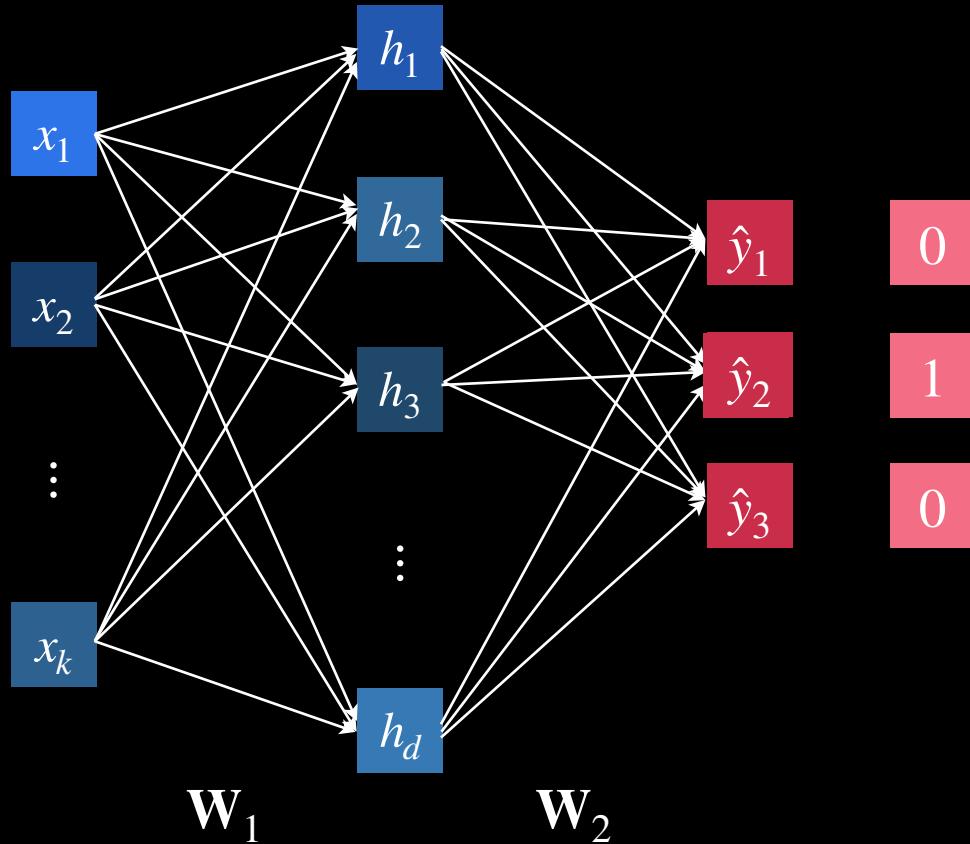
$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

$$E(\hat{\mathbf{y}}, \mathbf{y})$$

Функция ошибки

Для задачи классификации:

Обучение нейронных сетей



$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

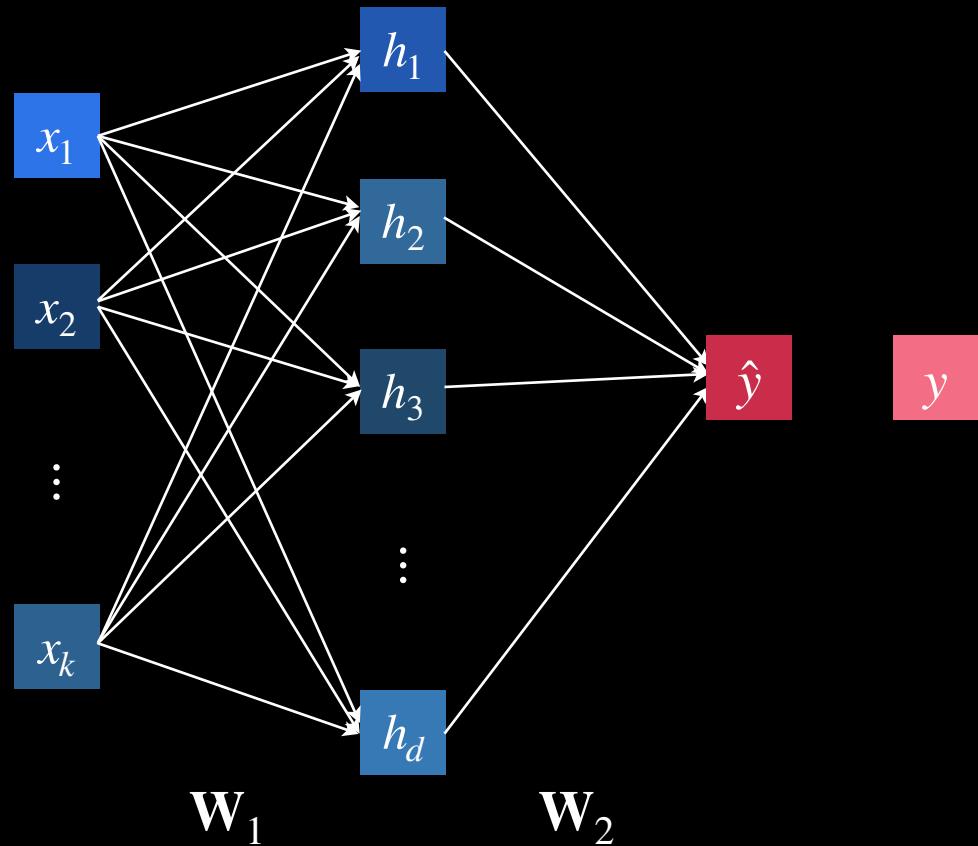
$E(\hat{y}, y)$ Функция ошибки

Для задачи классификации:

$$E(\hat{y}, y) = - \sum_i y_i \ln \hat{y}_i = -\ln(\hat{y}_y)$$

Перекрёстная энтропия
(cross-entropy)

Обучение нейронных сетей



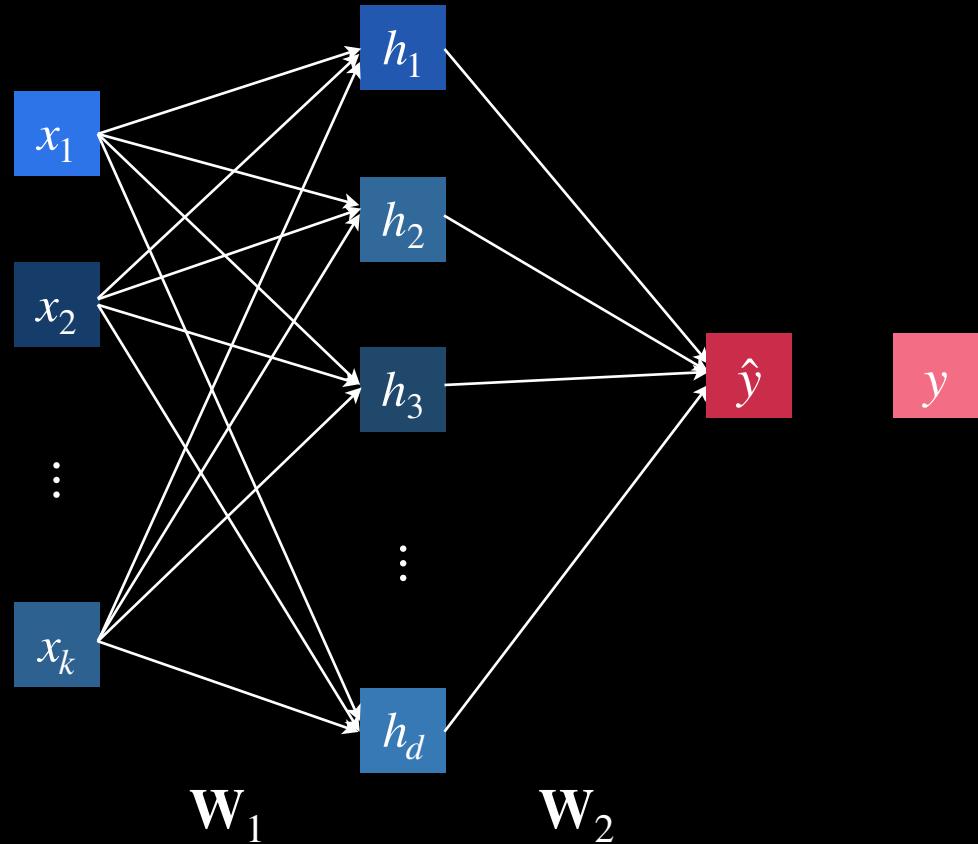
$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

$$E(\hat{y}, y)$$

Функция ошибки

Обучение нейронных сетей



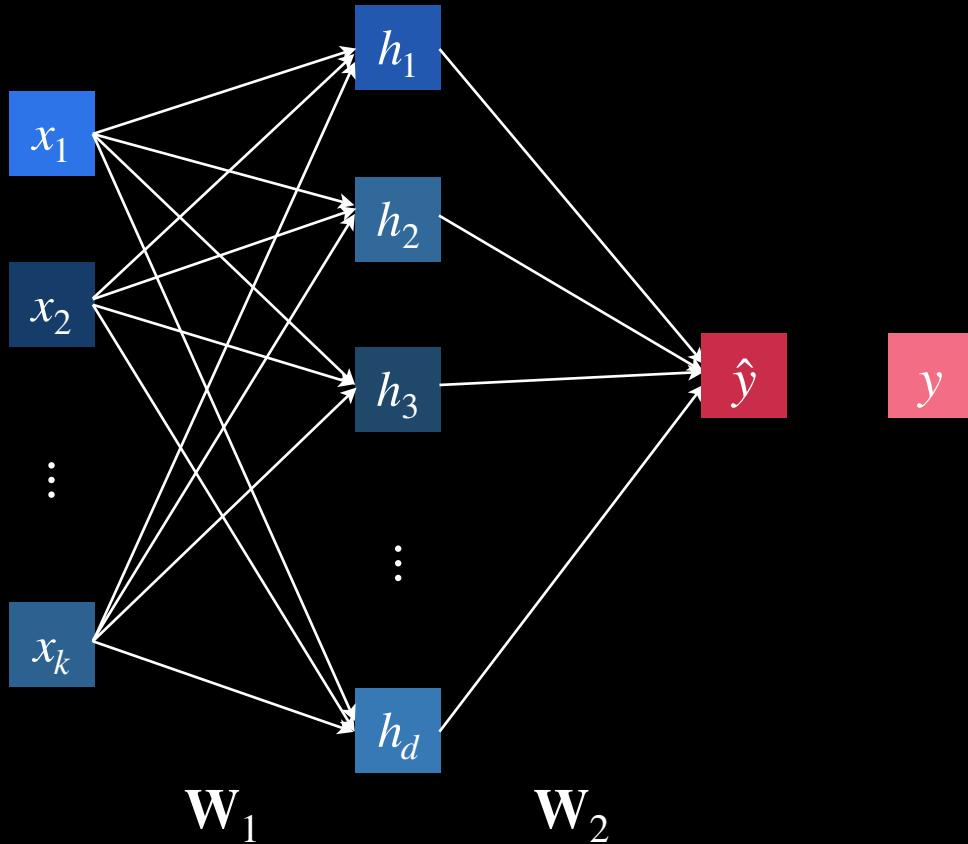
$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

$E(\hat{y}, y)$ Функция ошибки

$$E(\hat{y}, y) \longrightarrow \min_{\mathbf{W}}$$

Обучение нейронных сетей



$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

$E(\hat{y}, y)$ Функция ошибки

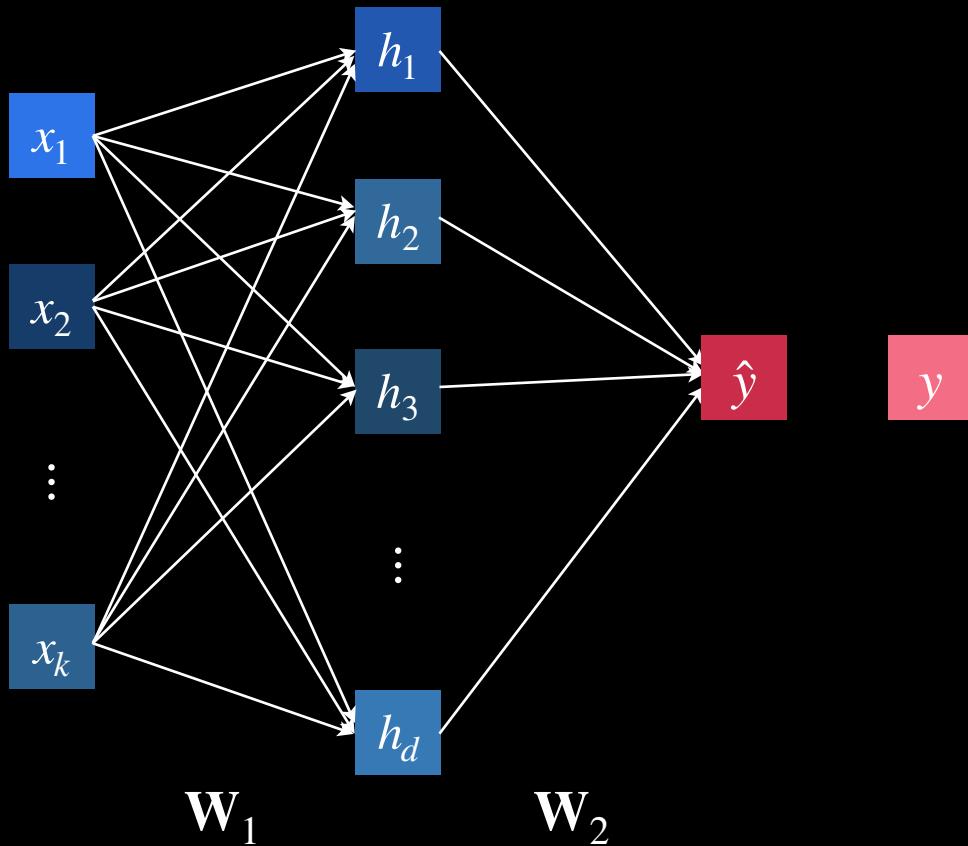
$$E(\hat{y}, y) \longrightarrow \min_{\mathbf{W}}$$

Трюк:

Функции активации — дифференцируемые

Функция ошибки — тоже

Обучение нейронных сетей



$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

$E(\hat{y}, y)$ Функция ошибки

$$E(\hat{y}, y) \longrightarrow \min_{\mathbf{W}}$$

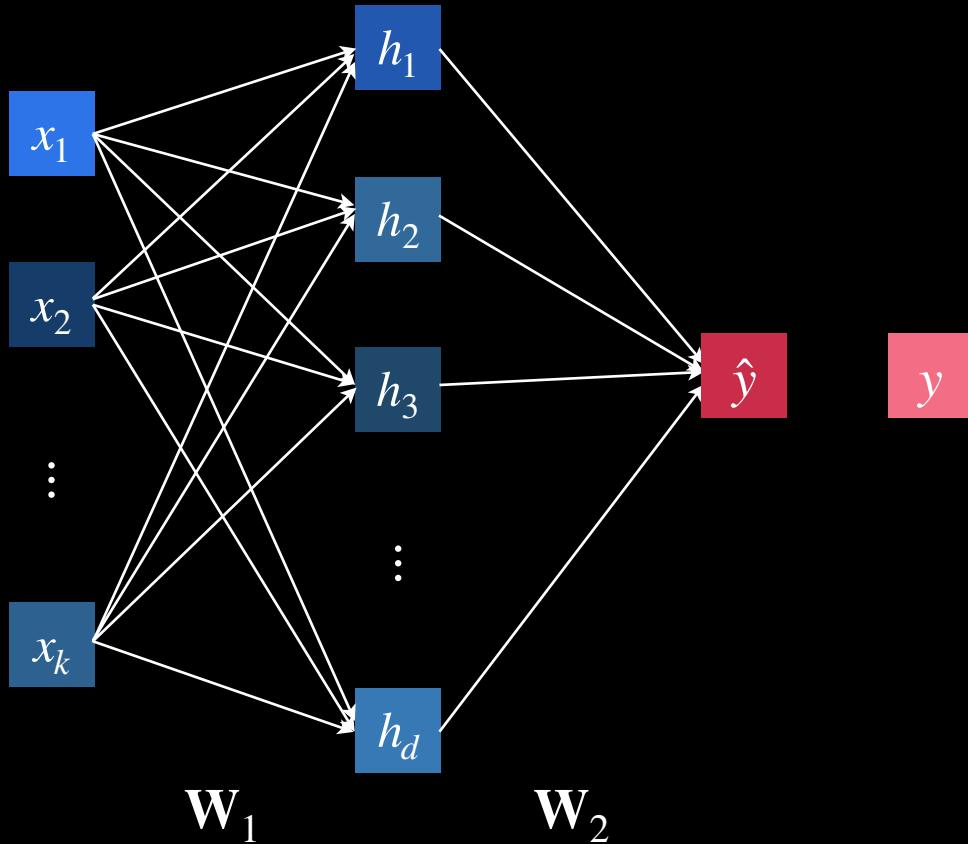
Трюк:

Функции активации — дифференцируемые

Функция ошибки — тоже

Значит можно посчитать производную ошибки по каждому из параметров!

Обучение нейронных сетей



$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

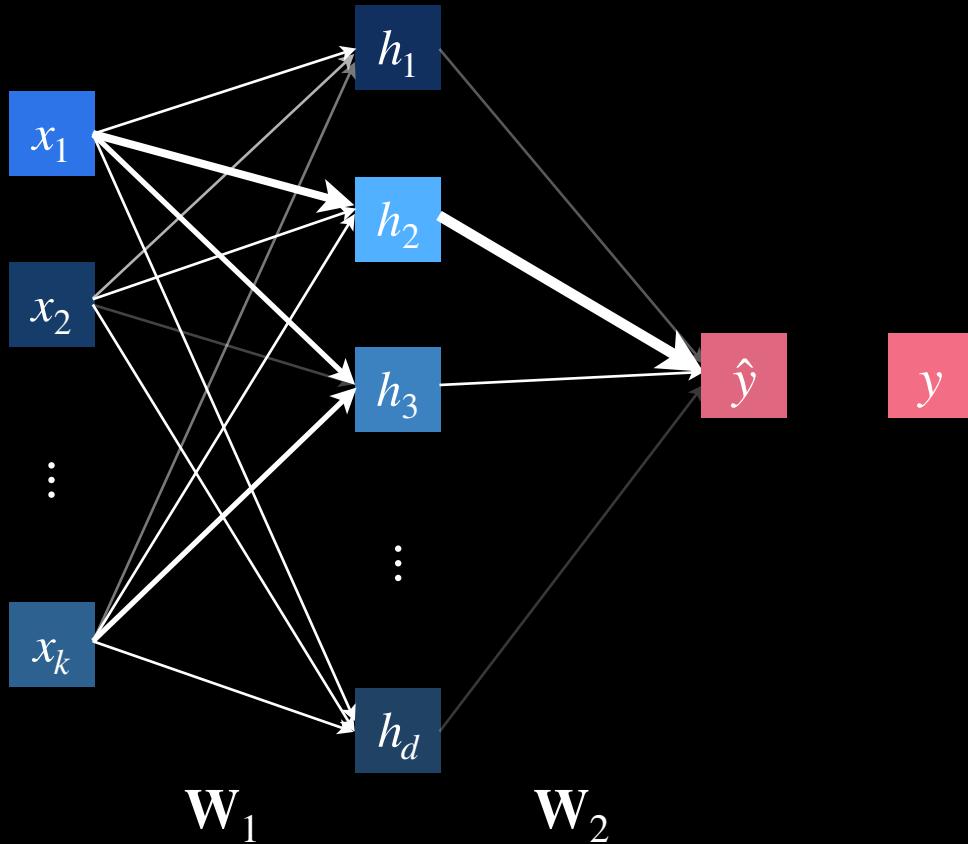
$E(\hat{y}, y)$ Функция ошибки

Градиентный спуск:

$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \nabla E$ Градиент

..... Размер шага

Обучение нейронных сетей



$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

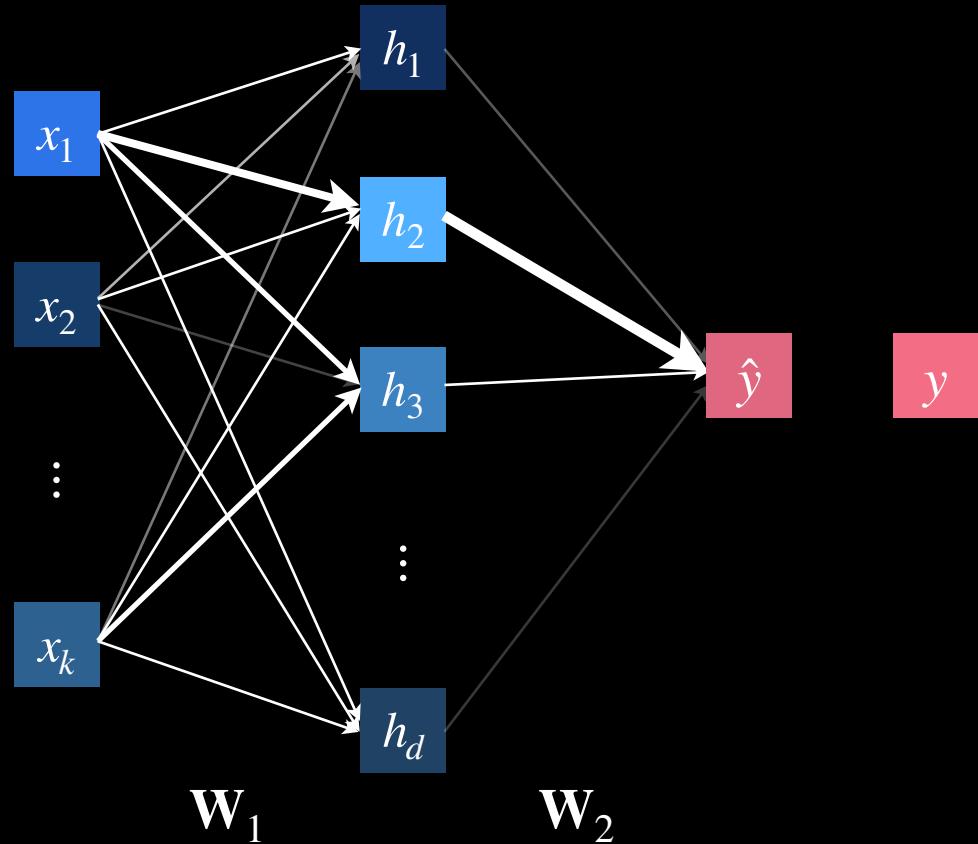
$E(\hat{y}, y)$ Функция ошибки

Градиентный спуск:

$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \nabla E$ Градиент

..... Размер шага

Обучение нейронных сетей



$$\mathbf{h} = f_h(\mathbf{W}_1 \mathbf{x})$$

$$\hat{\mathbf{y}} = f_o(\mathbf{W}_2 \mathbf{h})$$

$$E(\hat{y}, y)$$

Функция ошибки

Градиентный спуск:

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \nabla E$$

Градиент

Размер шага

Дифференцируемое программирование

```
import torch
from torch import nn, Tensor
from torch.optim import Optimizer

def training_step(
    features: Tensor, # входные признаки
    target: Tensor, # правильные ответы
    model: nn.Module, # модель с обучаемыми параметрами
    loss_fn: tuple[Tensor, Tensor] -> Tensor, # функция ошибки
    optimizer: Optimizer, # оптимизатор для обновления параметров
):
    prediction = model.forward(features) # прямой проход: вычисление предсказаний
    loss = loss_fn(prediction, target) # расчёт ошибки
    loss.backward() # обратный проход: вычисление градиентов
    optimizer.step() # обновление параметров модели
```