

Глубинное обучение

Лекция 4

Регуляризация. Dropout. Batch Normalization.

Автокодировщики.

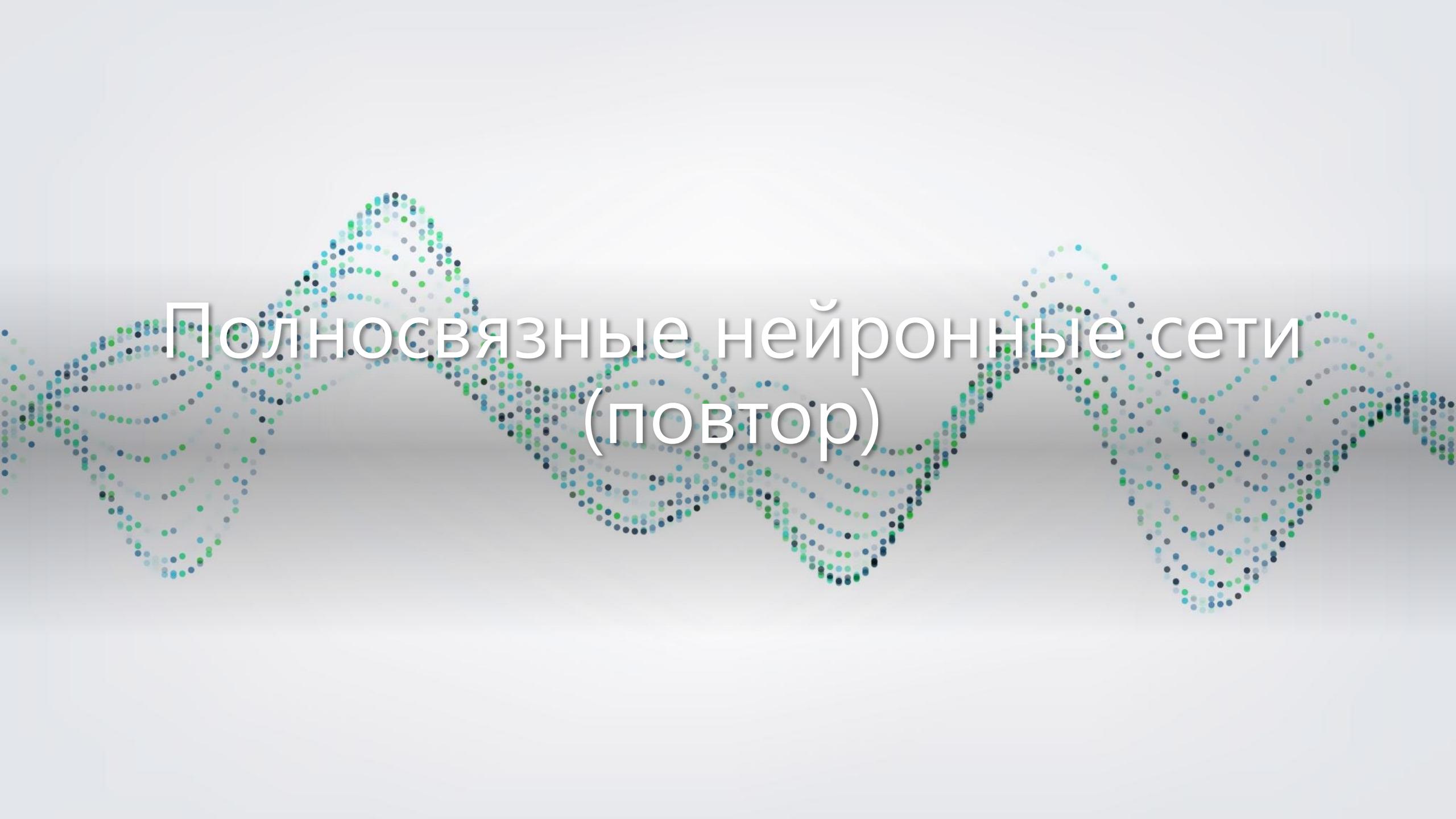
Михаил Гущин

mhushchyn@hse.ru

НИУ ВШЭ, 2025



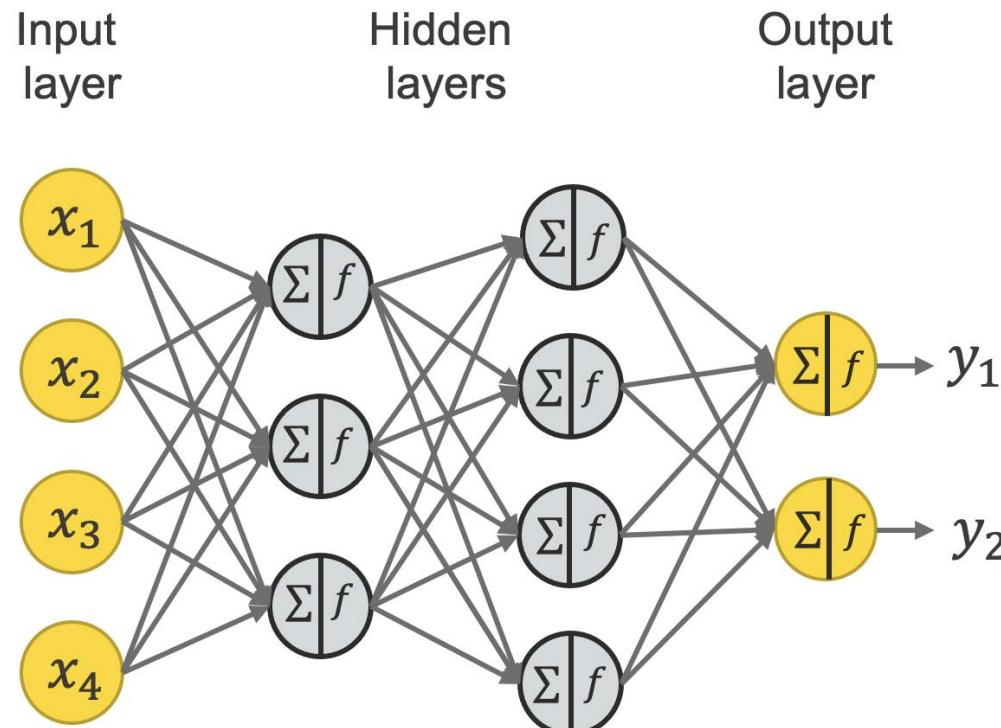
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ



Полносвязные нейронные сети (повтор)

Нейронная сеть

- ▶ Пусть дан набор наблюдений $\{x_i, y_i\}_{i=1}^n$, где $x_i \in R^d$, $y_i \in R^q$
- ▶ Построим нейронную сеть



Нейронная сеть в матричной форме

$$H^{(1)} = f_1(XW^{(1)} + b^{(1)})$$

$$H^{(2)} = f_2(H^{(1)}W^{(2)} + b^{(2)})$$

$$O = f_3(H^{(2)}W^{(3)} + b^{(3)})$$

Размеры матриц:

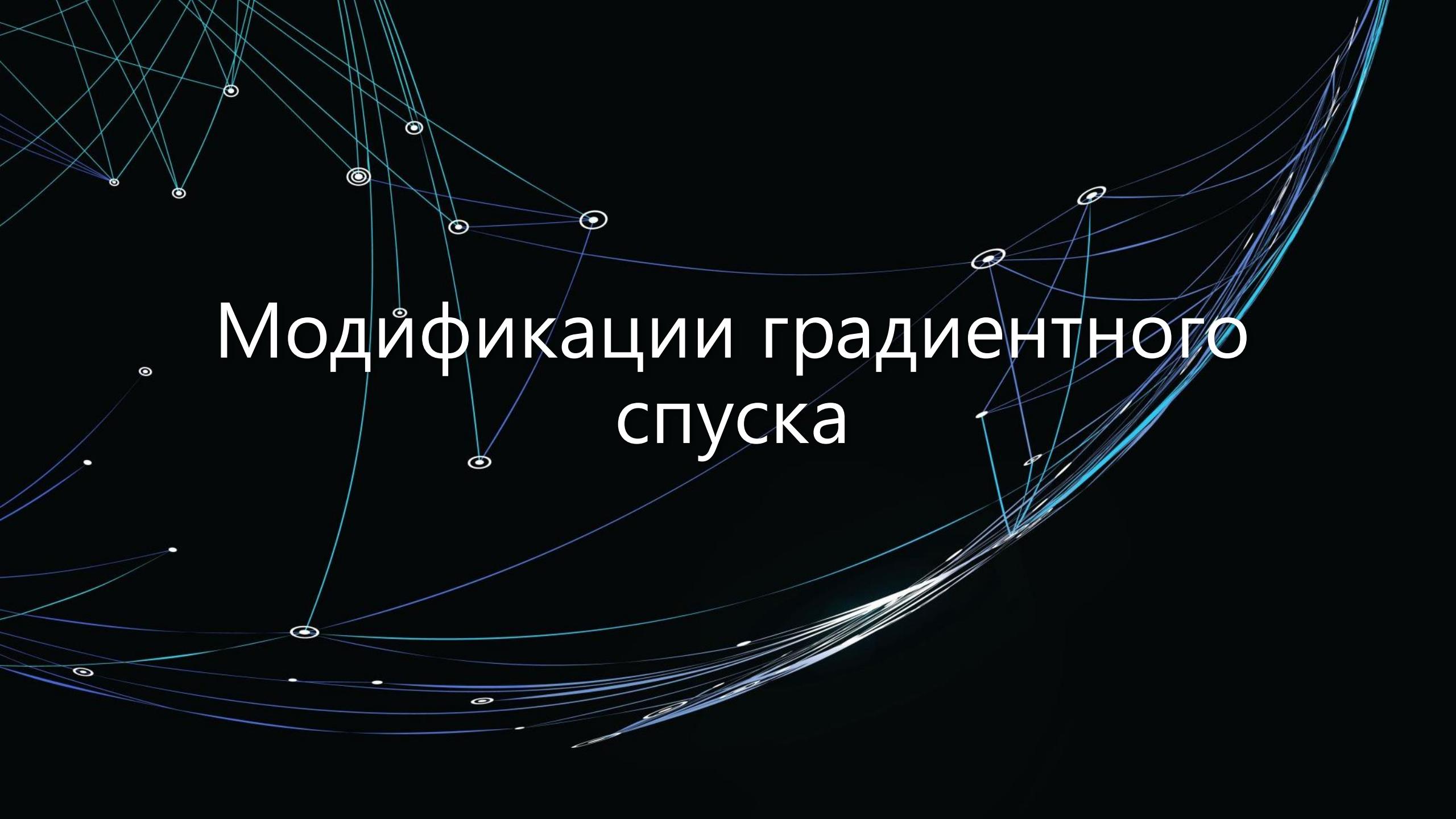
- ▶ $X \in R^{n \times d}$, $W^{(1)} \in R^{d \times h}$, $b^{(1)} \in R^{1 \times h}$, h - число нейронов в первом слое
- ▶ $H^{(1)} \in R^{n \times h}$, $W^{(2)} \in R^{h \times m}$, $b^{(2)} \in R^{1 \times m}$, m - число нейронов во втором слое
- ▶ $H^{(2)} \in R^{n \times m}$, $W^{(3)} \in R^{m \times q}$, $b^{(3)} \in R^{1 \times q}$, q - число выходов сети
- ▶ $O \in R^{n \times q}$ - матрица прогнозов сети

Градиентный спуск

- ▶ Нейронные сети обучаем с помощью градиентного спуска

$$w = w - \alpha \frac{\partial L}{\partial w}$$

- ▶ Как посчитать градиент функции потерь по нужному весу сети?
- ▶ Про это поговорим на следующей лекции



Модификации градиентного спуска

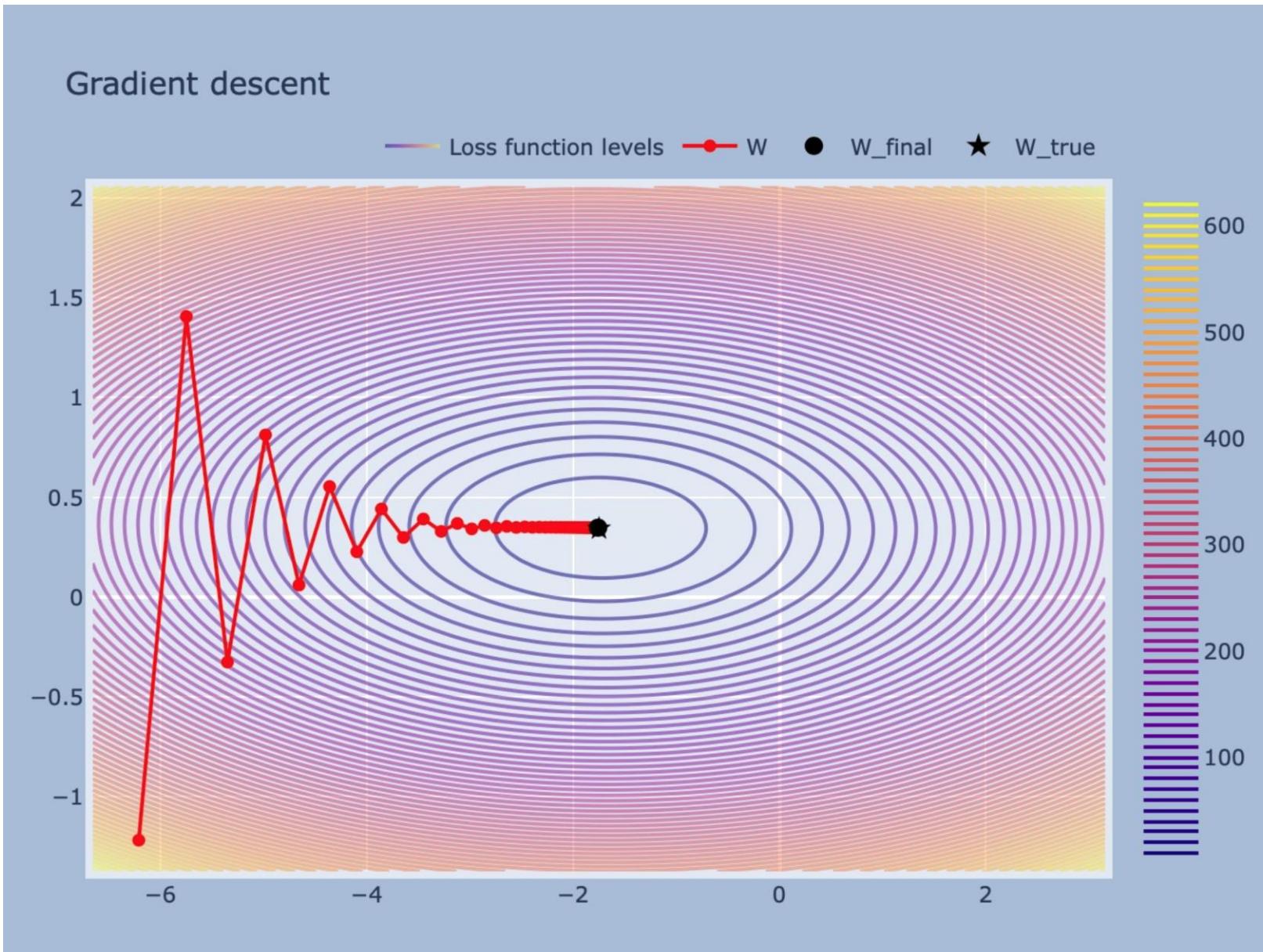
Проблемы градиентного спуска

- ▶ Рассмотрим функцию для оптимизации:

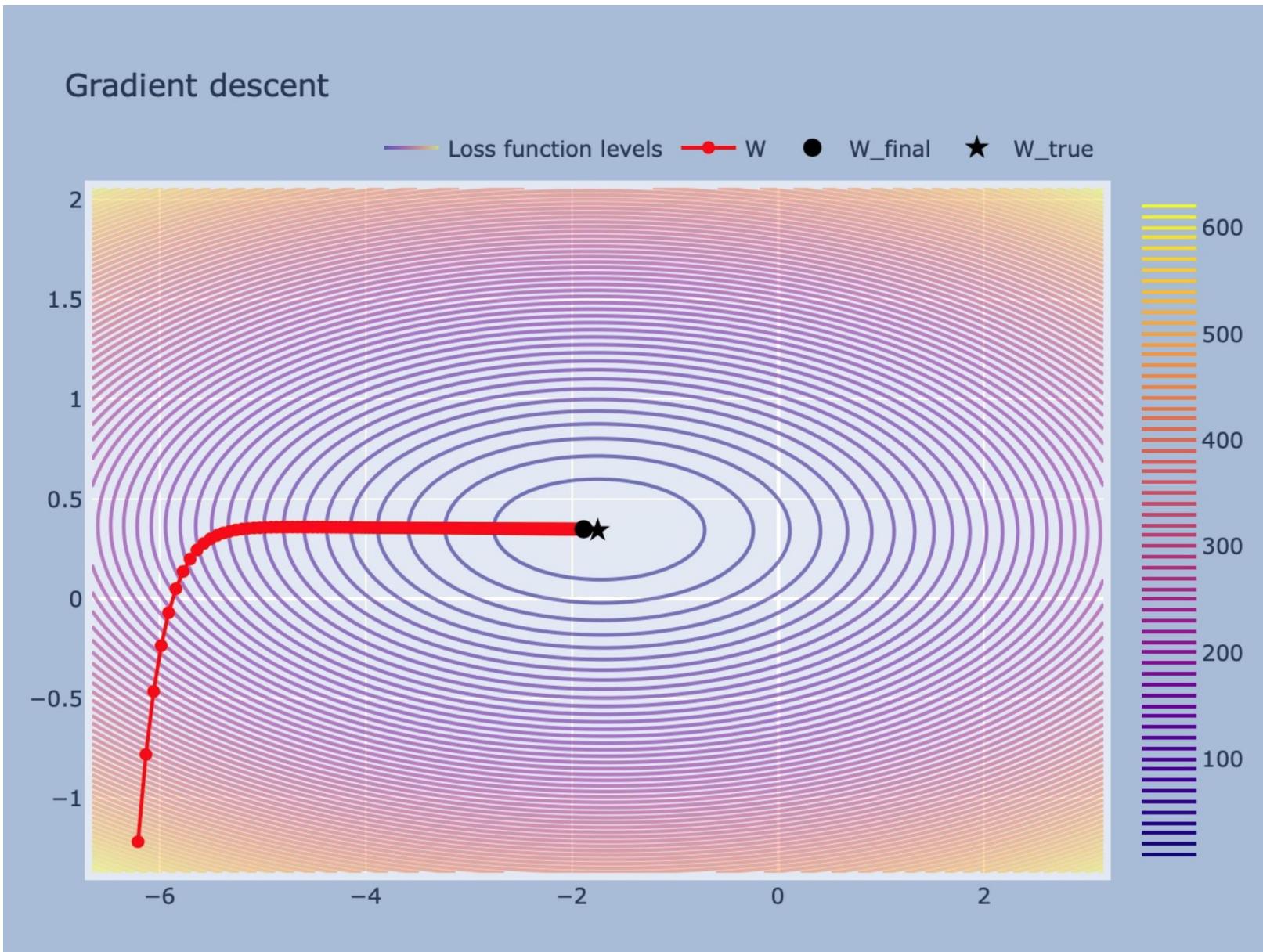
$$L(w) = 0.1w_1^2 + 2w_2^2$$

- ▶ Видим, что вклад w_2 сильно больше, чем для w_1
- ▶ Применим градиентный спуск

Проблемы градиентного спуска



Проблемы градиентного спуска



Проблемы градиентного спуска

- ▶ Если линии уровня функции вытянуты, то градиентный спуск требует аккуратного выбора длины шага
- ▶ Если шаг слишком большой, то градиентный спуск не сойдется
- ▶ Если шаг слишком маленький, то градиентный спуск будет сходиться слишком долго

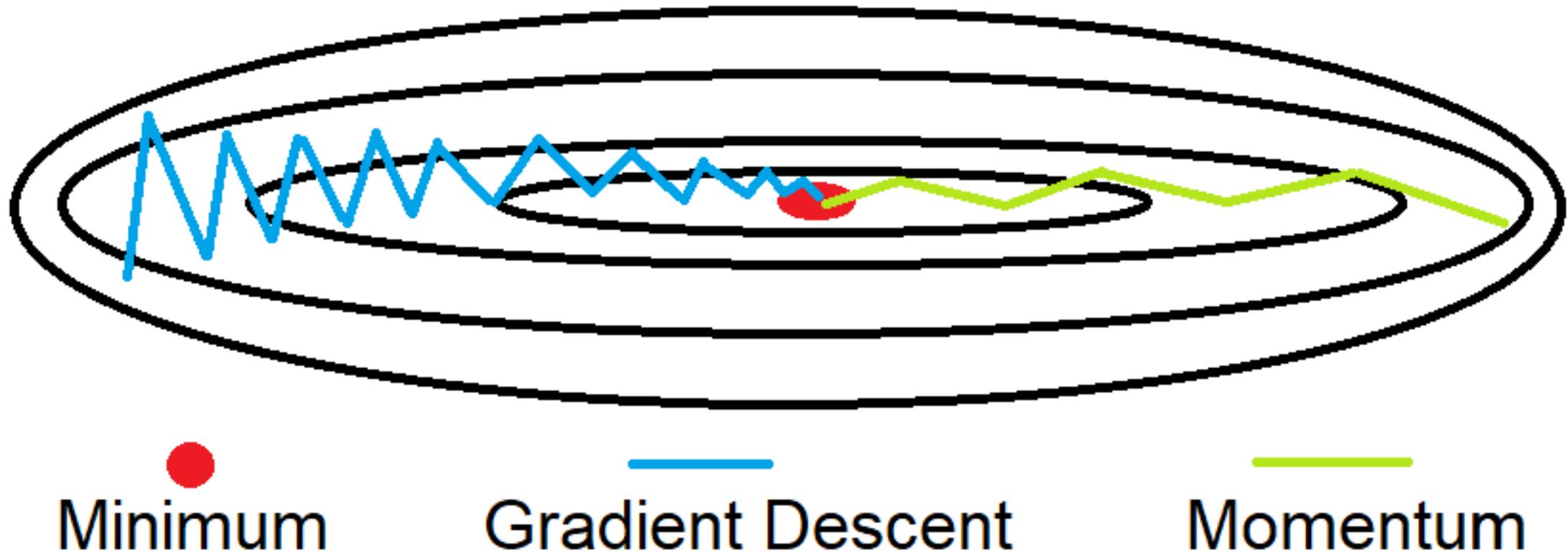
Momentum

$$h_t = \beta h_{t-1} + \nabla L(w^{(t-1)})$$

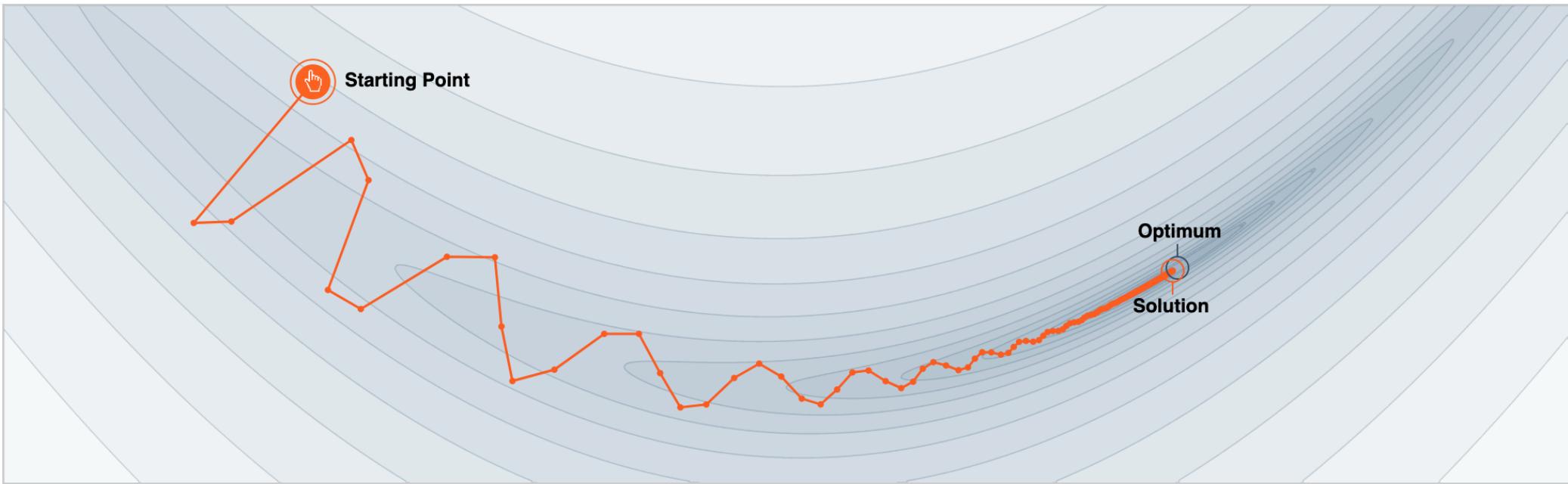
$$w^{(t)} = w^{(t-1)} - \eta h_t$$

- ▶ h_t – инерция, усредненное направление движения
- ▶ β – гиперпараметр затухания

Пример momentum



Приимер momentum



Step-size $\alpha = 0.02$



Momentum $\beta = 0.83$



We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?

<https://distill.pub/2017/momentum>

AdaGrad

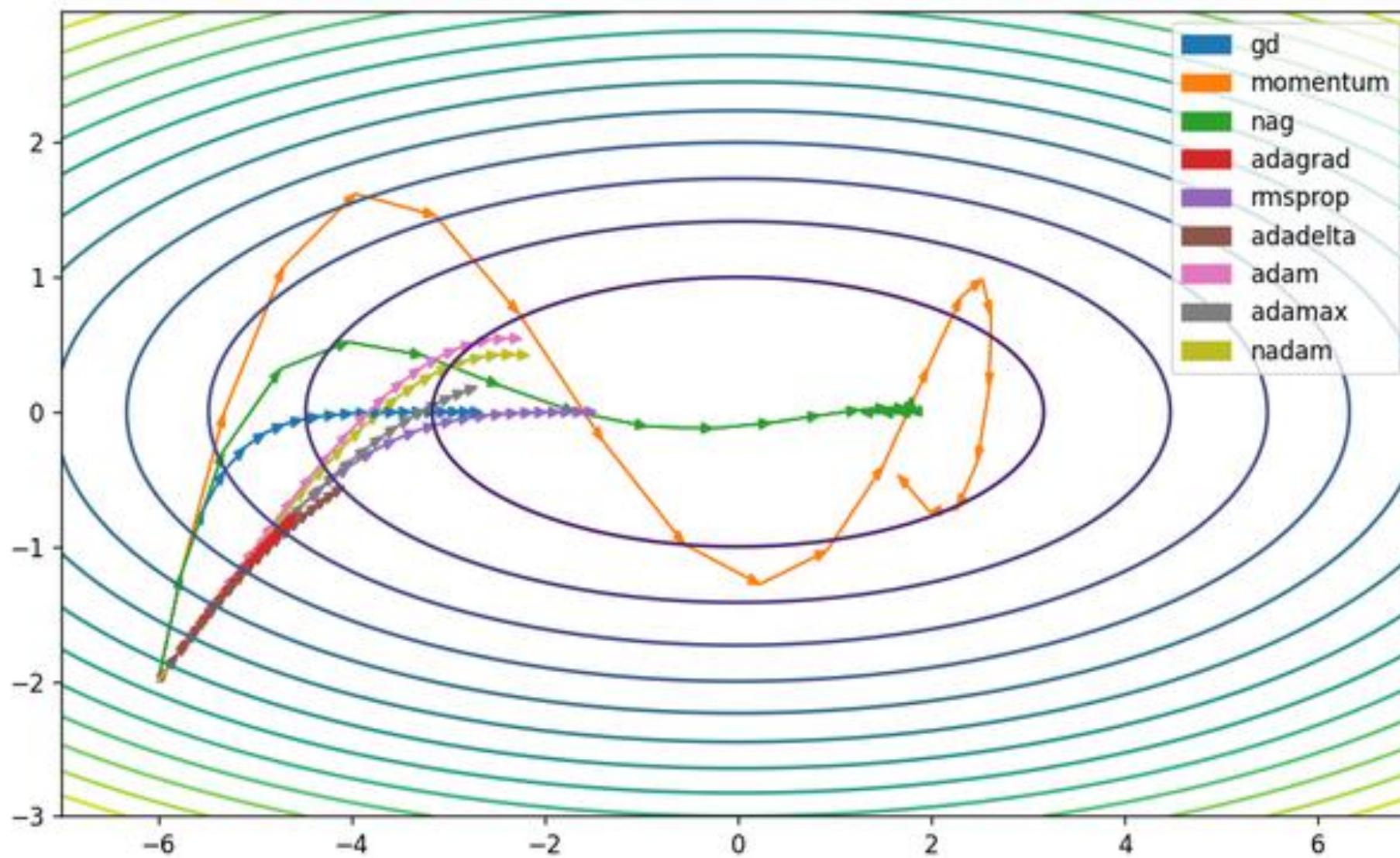
$$g_{t-1} = \nabla L(w^{(t-1)})$$

$$s_t = s_{t-1} + (g_{t-1})^2$$

$$w^{(t)} = w^{(t-1)} - \frac{\eta}{\sqrt{s_t + \epsilon}} \odot g_{t-1}$$

- ▶ \odot - поэлементное умножение
- ▶ По каждому параметру (весу сети) своя скорость спуска. Это полезно, когда у признаков разный масштаб (1, 100, 1000)

Пример



RMSprop

$$g_{t-1} = \nabla L(w^{(t-1)})$$

$$s_t = \gamma s_{t-1} + (1 - \gamma)(g_{t-1})^2$$

$$w^{(t)} = w^{(t-1)} - \frac{\eta}{\sqrt{s_t + \epsilon}} \odot g_{t-1}$$

- ▶ $\gamma=0.9$
- ▶ Скорость спуска сильнее зависит от недавних шагов

Adam

$$g_{t-1} = \nabla L(w^{(t-1)})$$

$$h_t = \beta_1 h_{t-1} + (1 - \beta_1) g_{t-1}$$

$$\hat{h}_t = \frac{h_t}{1 - \beta_1^t}$$

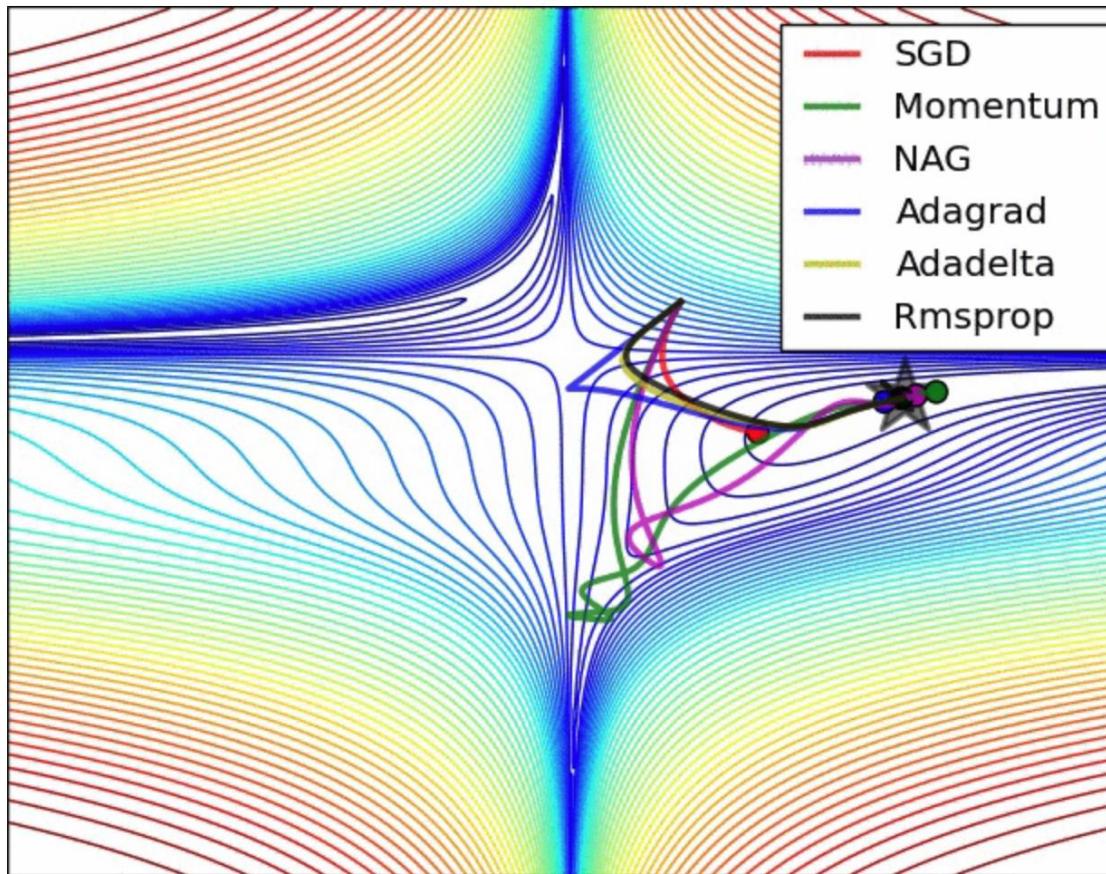
$$s_t = \beta_2 s_{t-1} + (1 - \beta_2) (g_{t-1})^2$$

$$\hat{s}_t = \frac{s_t}{1 - \beta_2^t}$$

$$w^{(t)} = w^{(t-1)} - \frac{\eta}{\sqrt{\hat{s}_t + \epsilon}} \odot \hat{h}_t$$

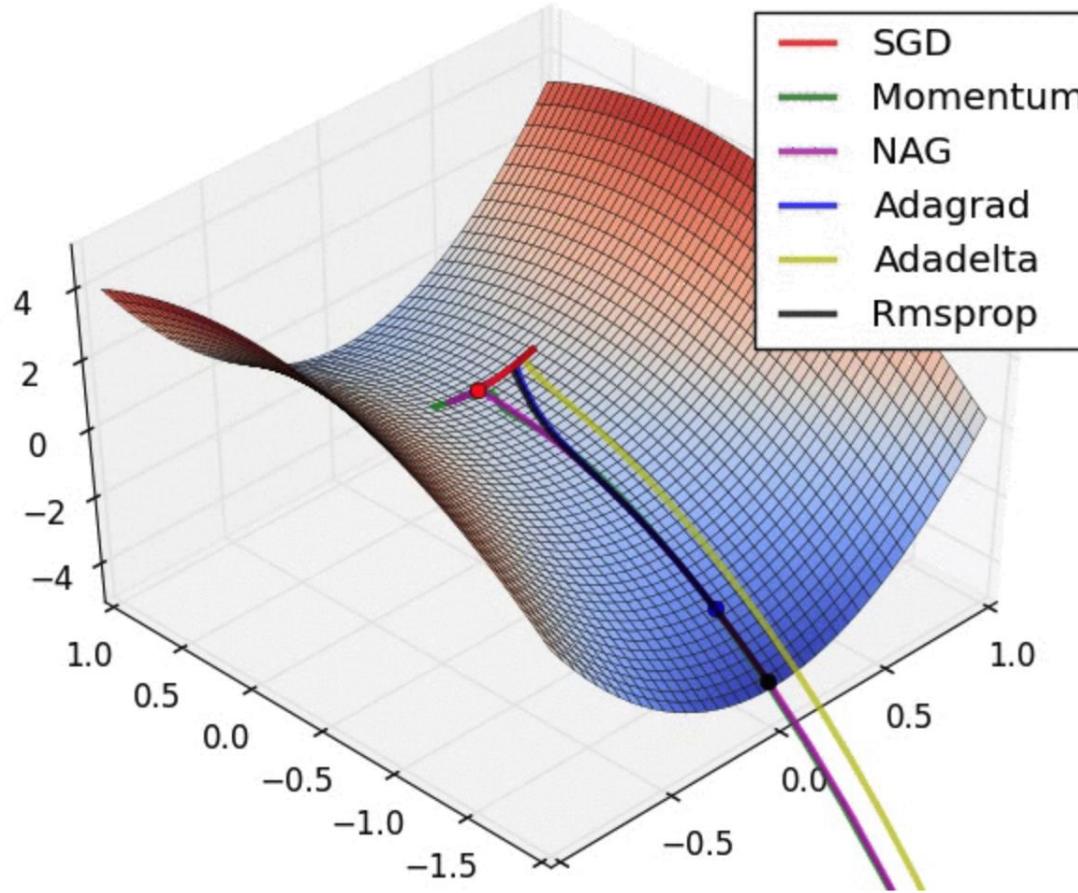
- ▶ $\beta_1 = 0.9$
- ▶ $\beta_2 = 0.999$

Пример



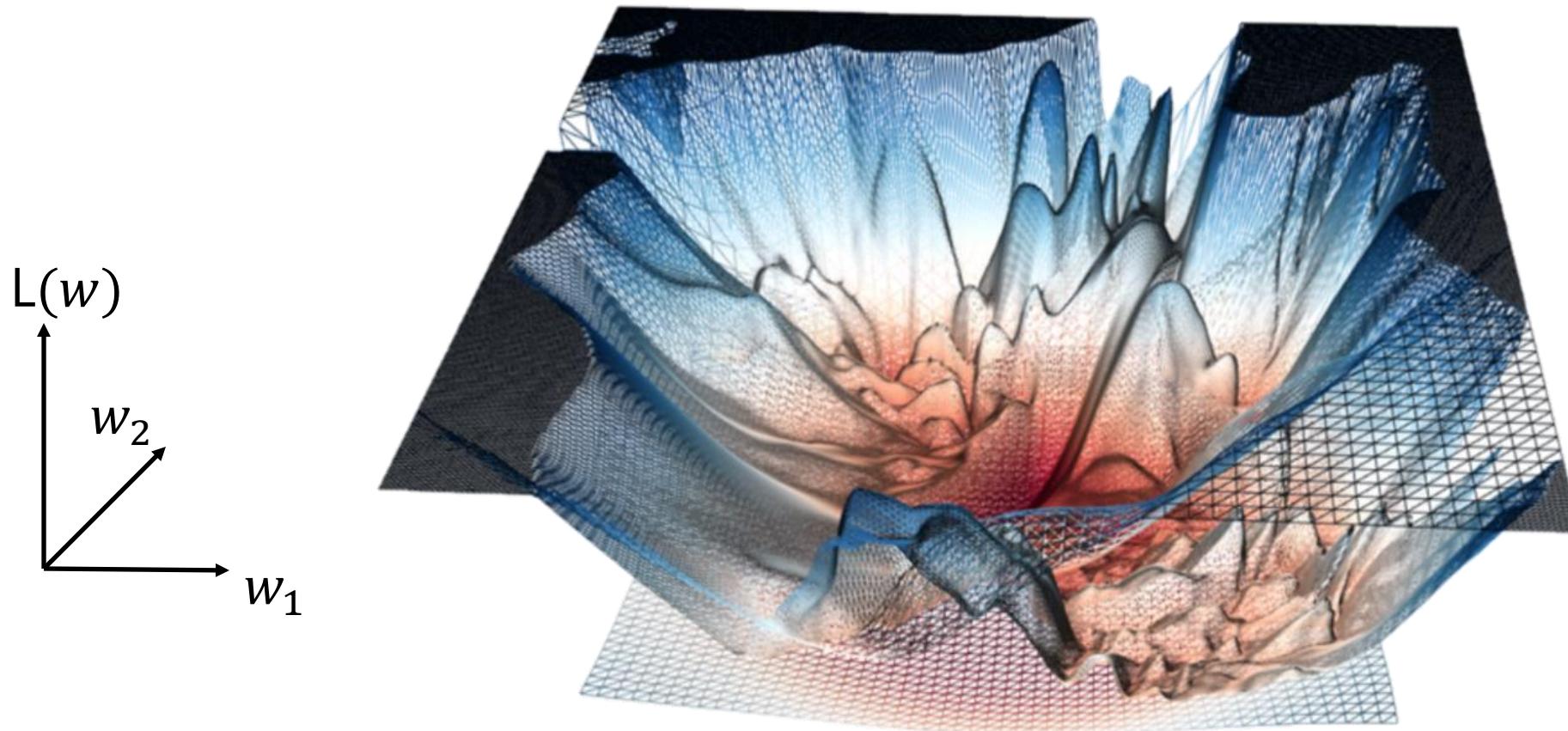
<https://api.wandb.ai/files/lavanyashukla/images/projects/36993/d07b8a79.gif>

Пример

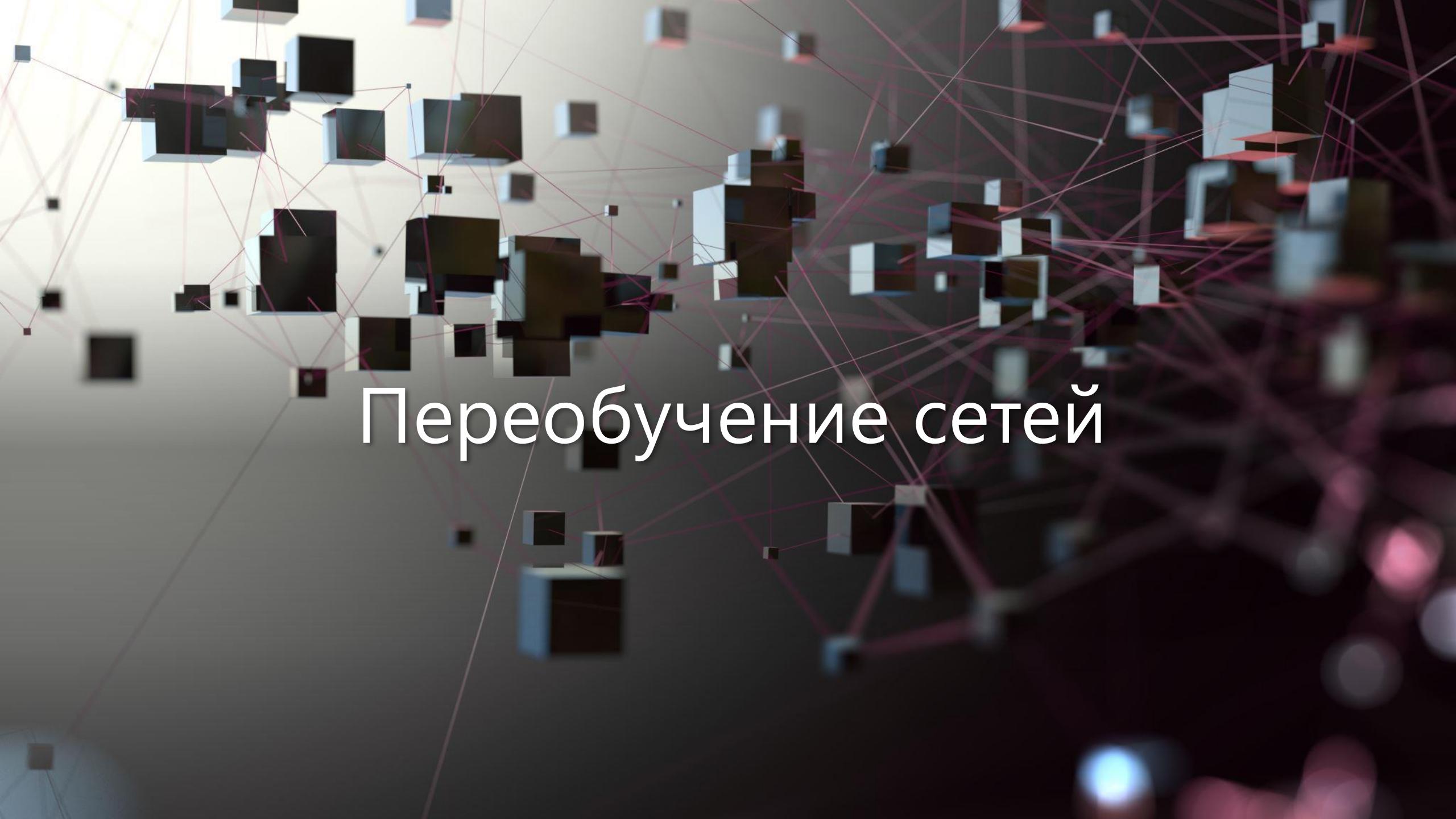


<https://api.wandb.ai/files/lavanyashukla/images/projects/36993/2a6f771c.gif>

Примеры функций потерь нейронных сетей



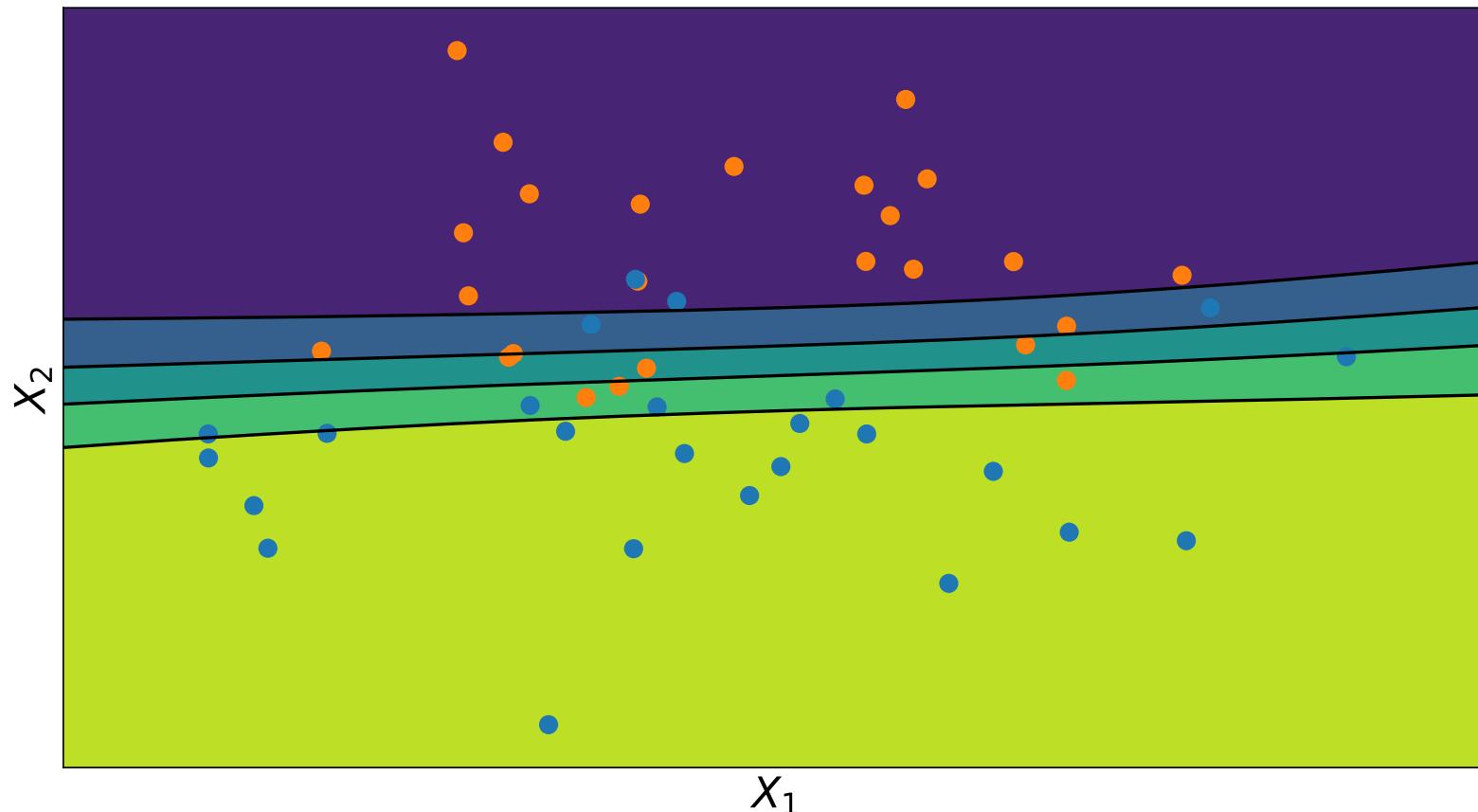
- ▶ Demo: <https://www.telesens.co/loss-landscape-viz/viewer.html>
- ▶ Функции потерь как искусство: <https://losslandscape.com/>



Переобучение сетей

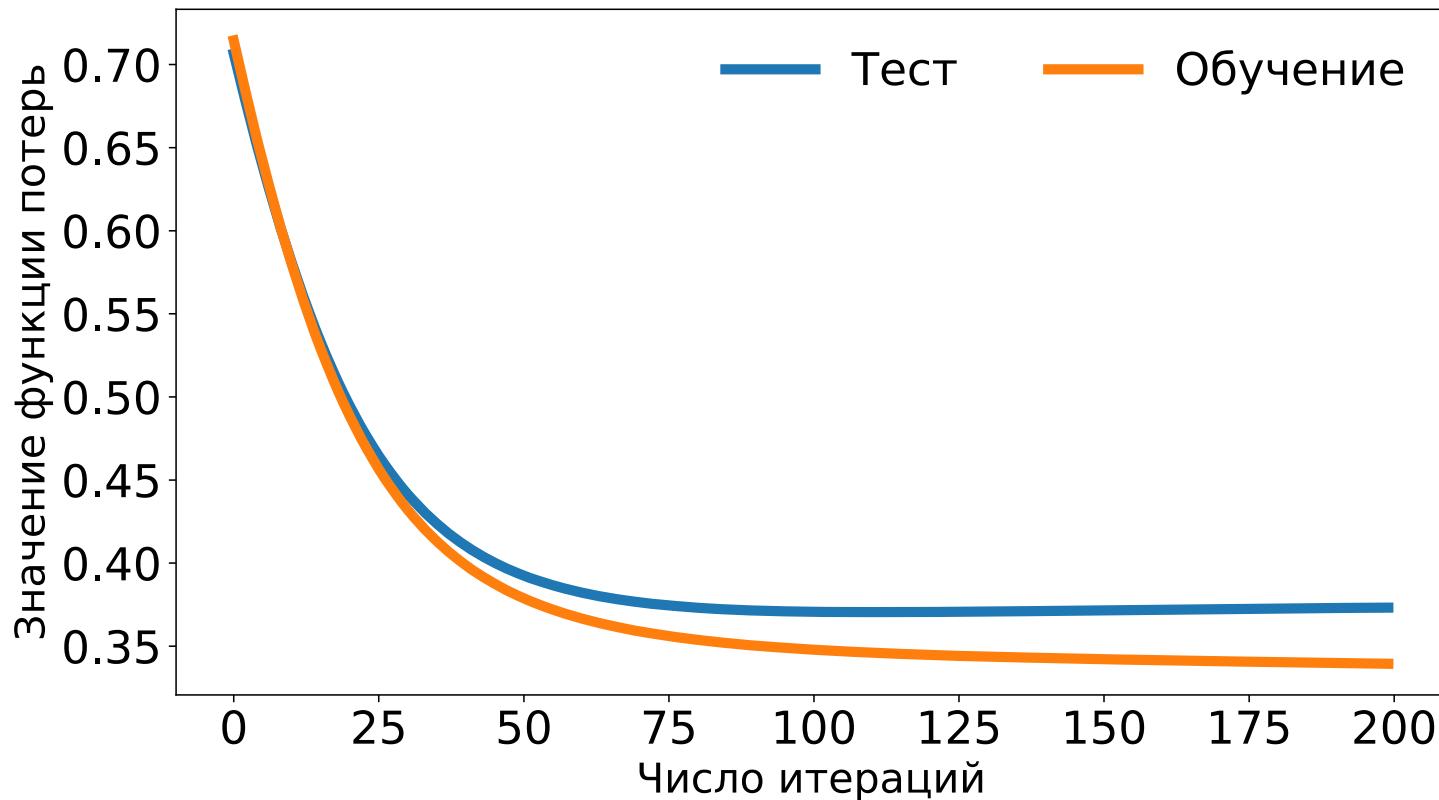
Оптимальное обучение сети

- ▶ Рассмотрим задачу бинарной классификации



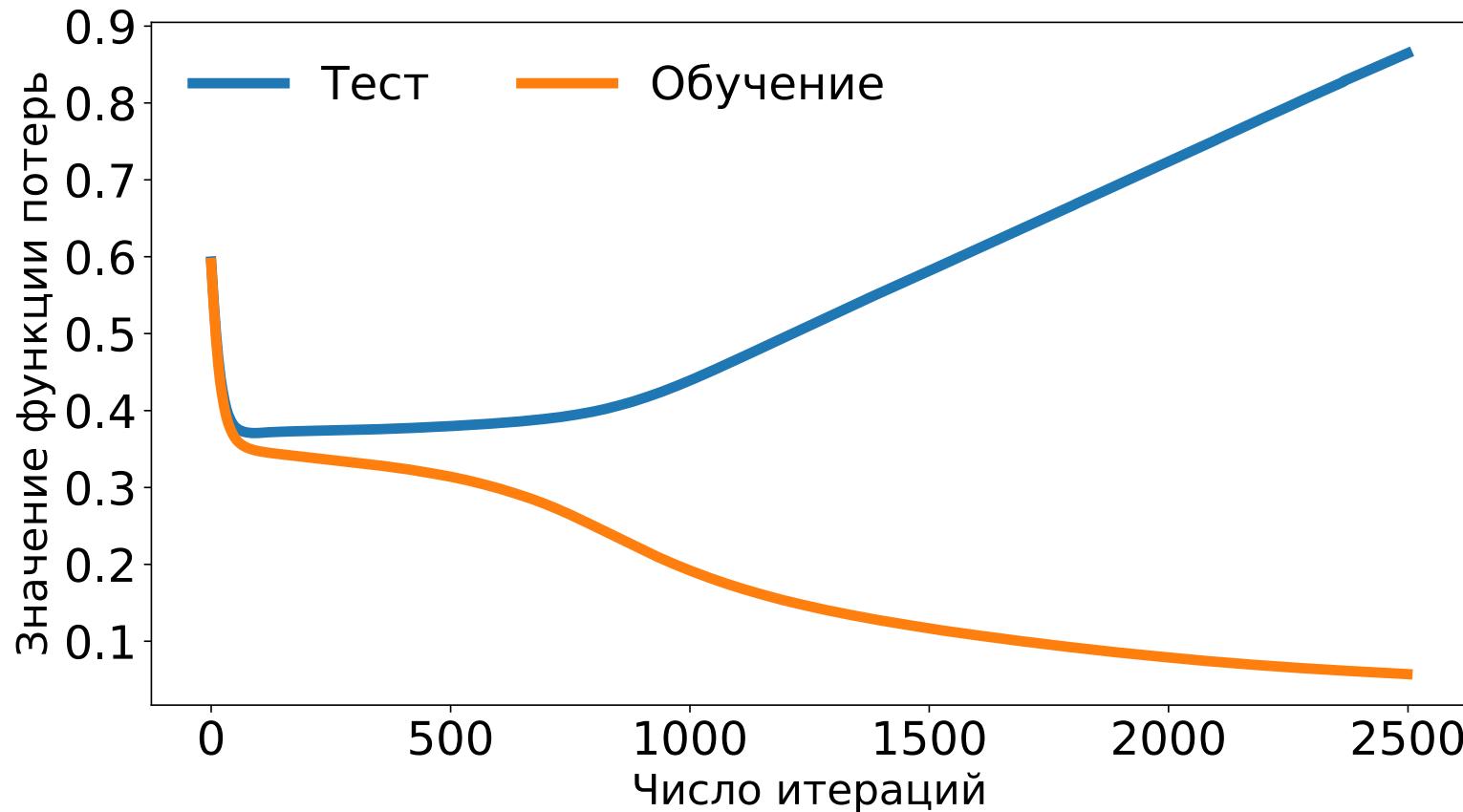
Оптимальное обучение сети

- ▶ При оптимальном решении значения функции потерь на обучающей и тестовой выборках должны быть близкими и малыми



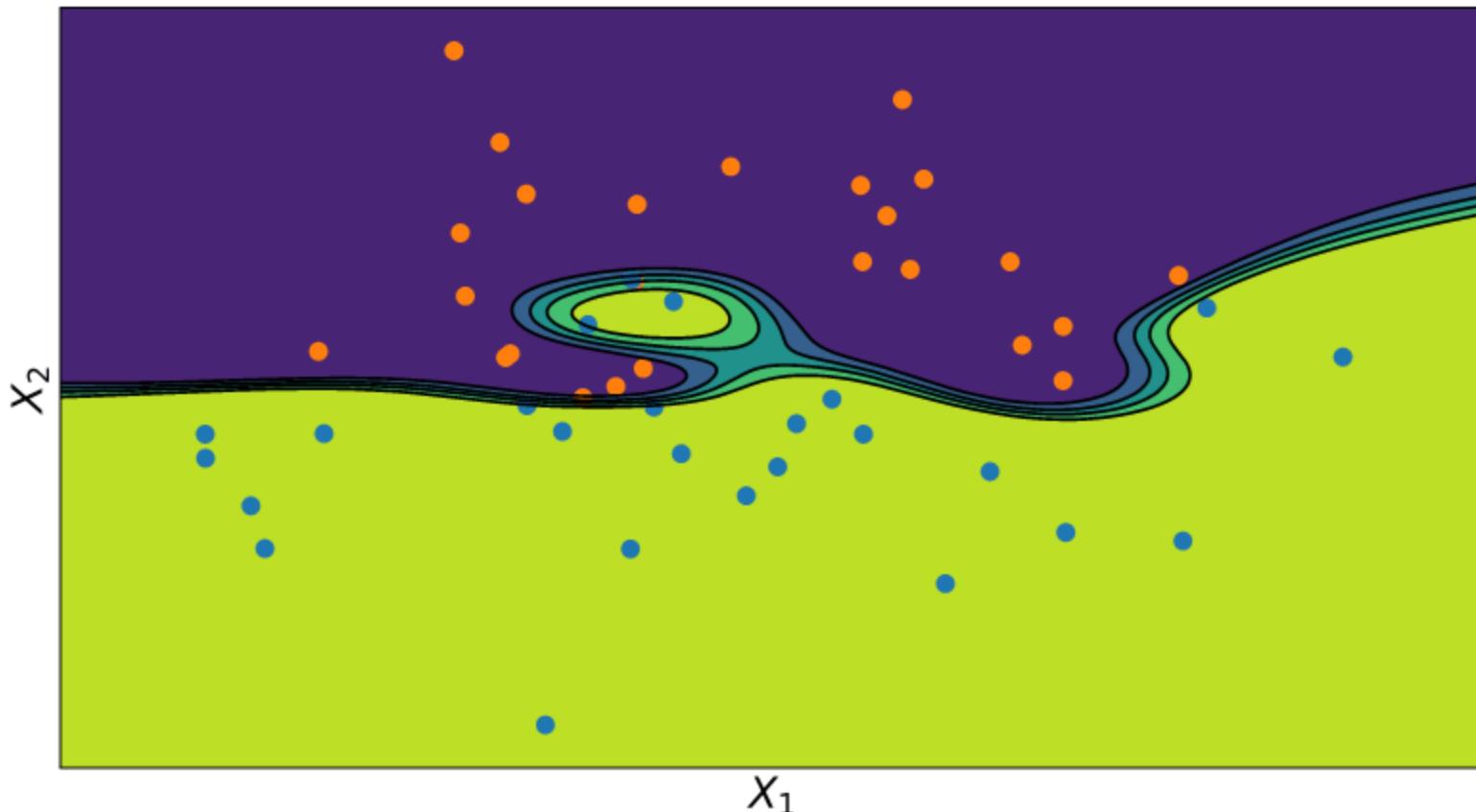
Переобучение

- ▶ При **переобучении** сети значение функции потерь на обучении сильно меньше, чем на teste

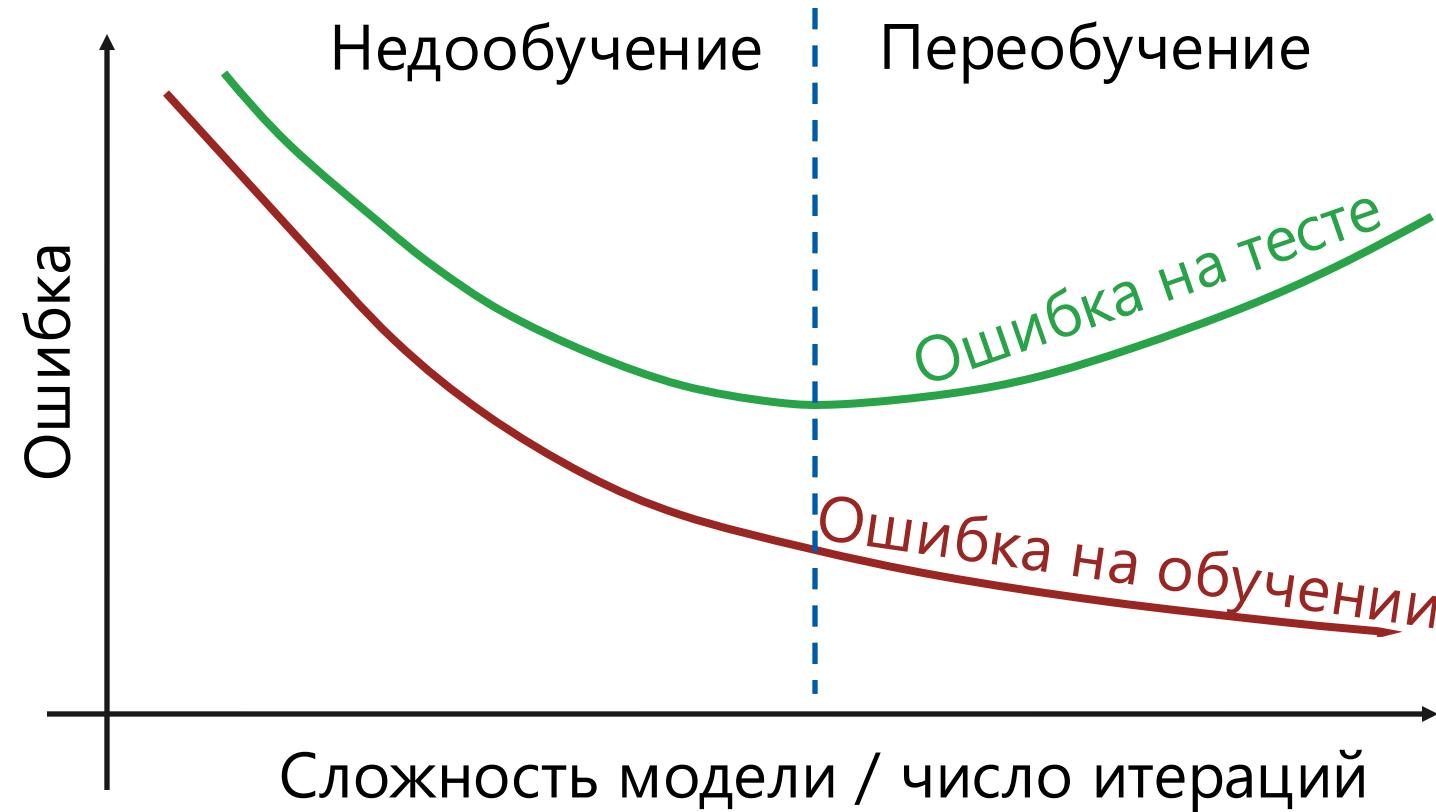


Переобучение

- ▶ Нейронная сеть выучивает особенности обучающей выборки, которые не характерны для тестовой выборки.



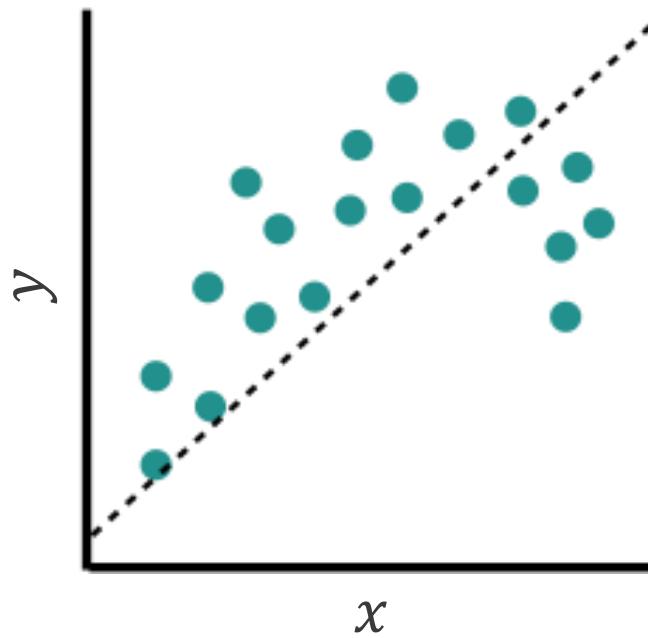
Кривая обучения



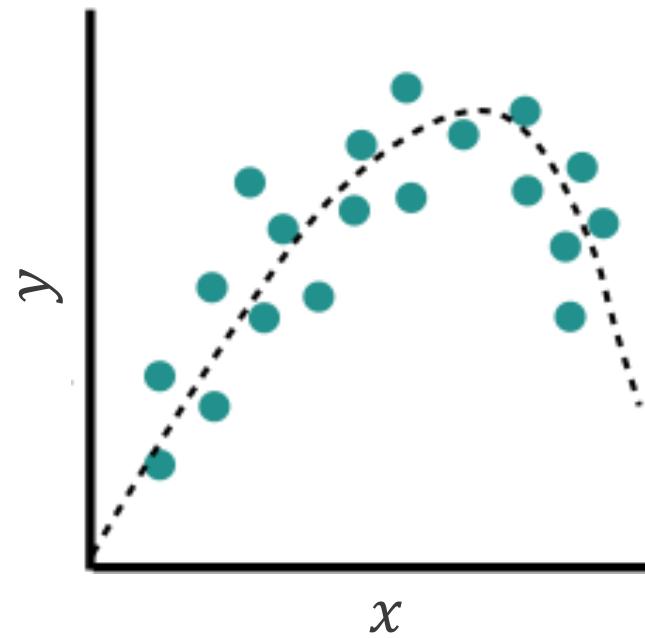
* Сложность модели = количество параметров для обучения

Недообучение и переобучение

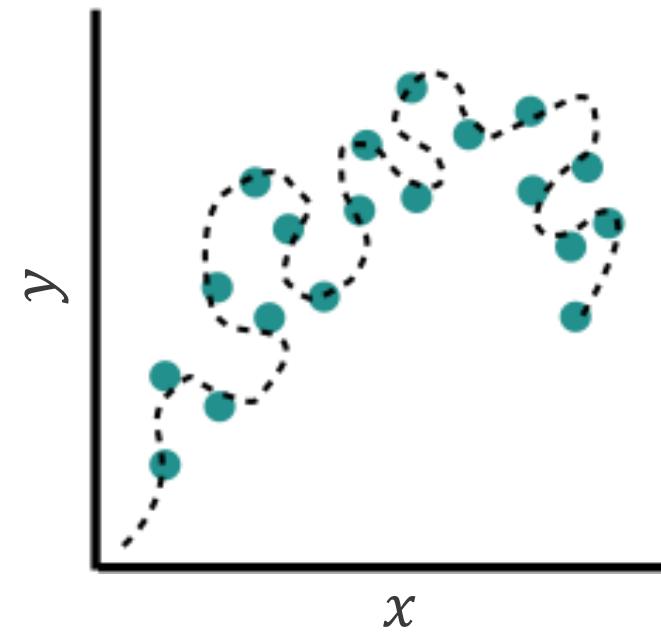
Недообучение



Оптимум

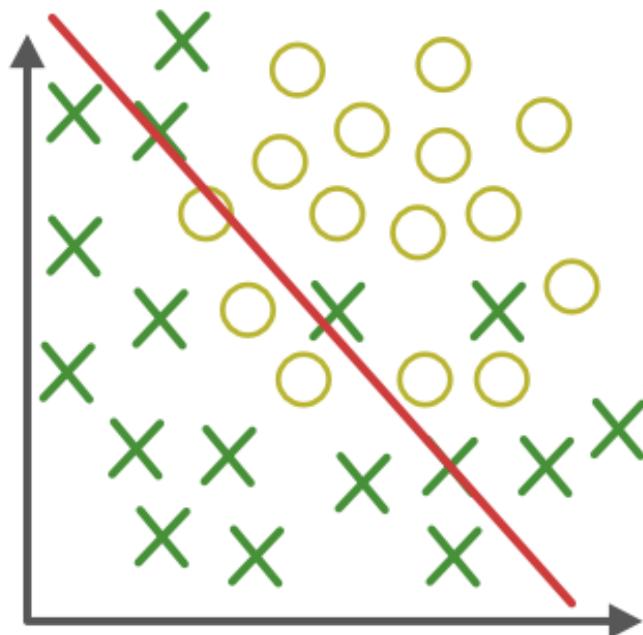


Переобучение

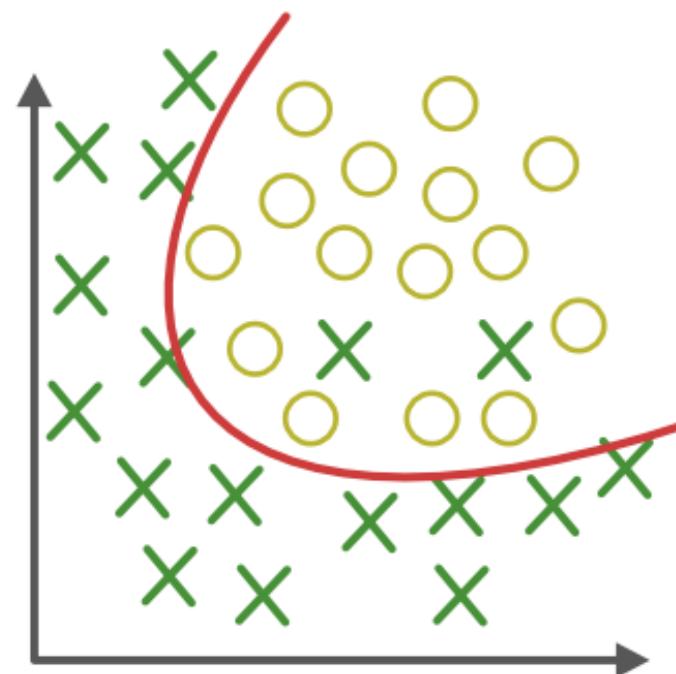


Недообучение и переобучение

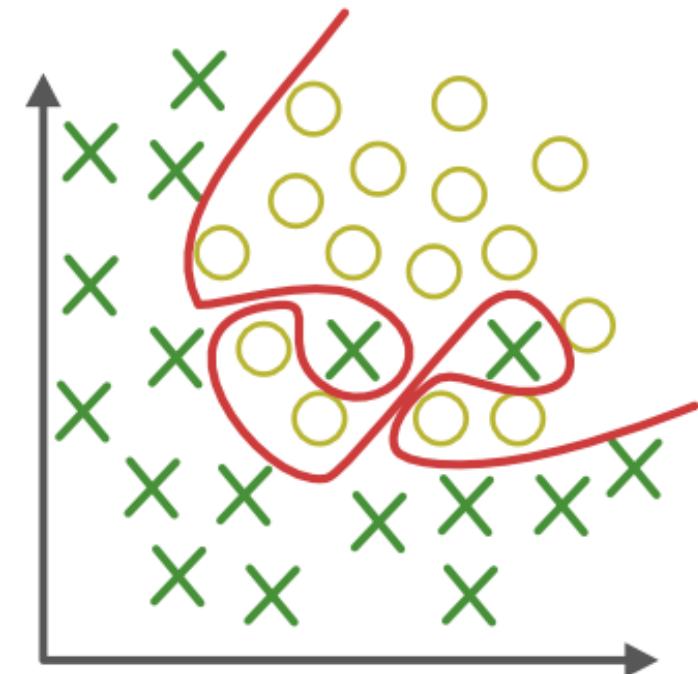
Недообучение



Оптимум

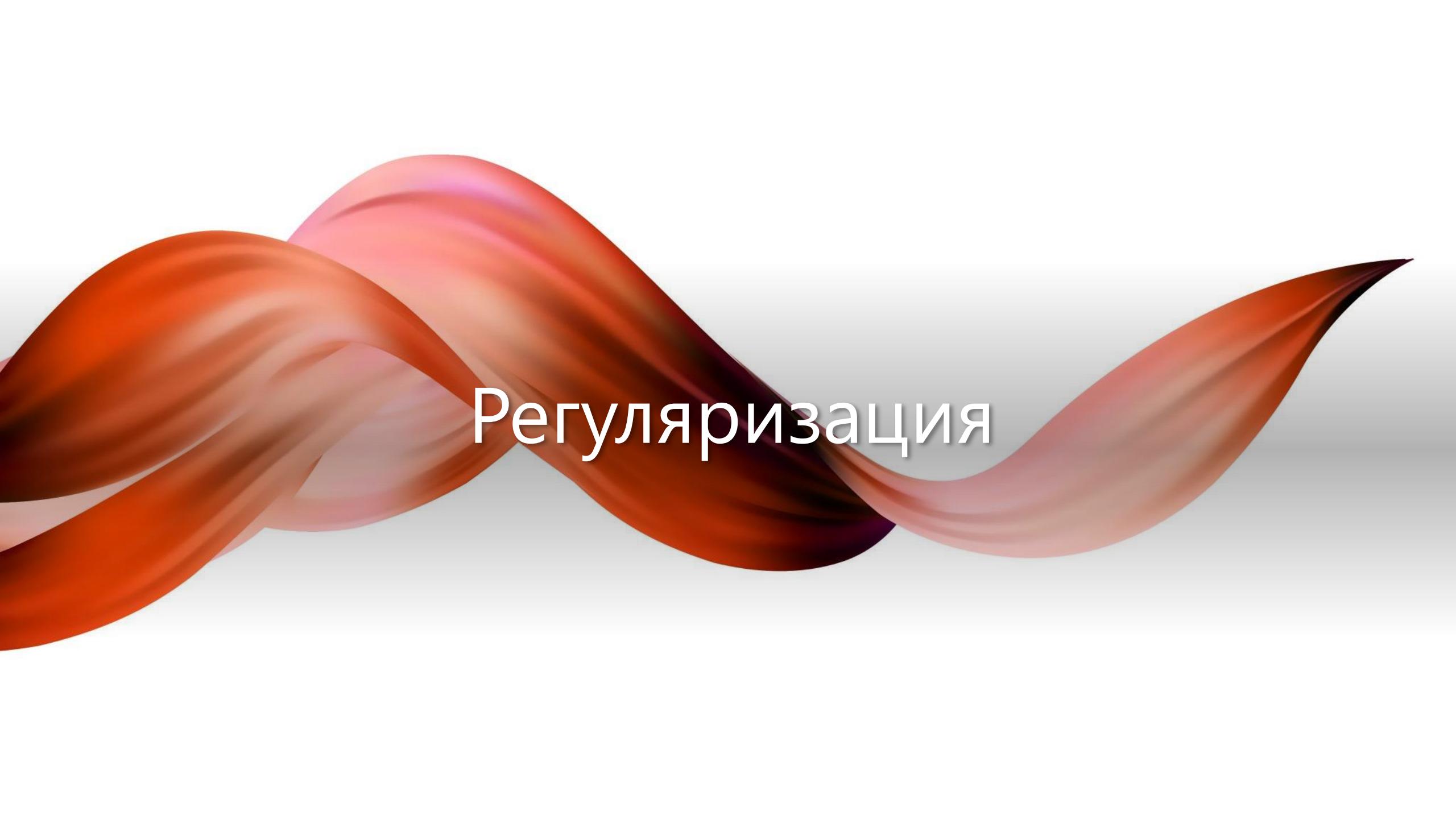


Переобучение



Переобучение

- ▶ Ранняя остановка обучения позволяет избежать переобучения сети
- ▶ Какие есть еще способы?

The background features two thick, flowing ribbons in shades of orange, red, and pink against a white gradient background. One ribbon curves from the bottom left towards the top right, while the other follows a similar path from the bottom left but ends in a sharp point on the right side.

Регуляризация

Мотивация

- ▶ Регуляризация позволяет избежать переобучения сети
- ▶ Улучшает качество прогноза модели

Обучение нейрона

Рассмотрим для примера один нейрон:

$$\hat{y} = g\left(\sum_{i=1}^n w_i x_i\right)$$

И некоторую функцию потерь:

$$L = L(\hat{y}, y) \rightarrow \min_w$$

где \hat{y} — прогноз нейрона;

$y \in \{0, 1\}$ — истинные метки классов

Обучение нейрона

- ▶ Веса нейронов могут становиться большими, что приводит к чувствительности к малым изменениям входа и переобучению
- ▶ Будем штрафовать сеть за большие веса
- ▶ Добавим этот штраф в функцию потерь

Регуляризация

L_1 — регуляризация или Lasso:

$$L' = L(\hat{y}, y) + \beta_1 \sum_i |w_i|$$

- ▶ β_1 - гиперпараметр.

Регуляризация

L_2 — регуляризация или Ridge:

$$L' = L(\hat{y}, y) + \beta_2 \sum_i w_i^2$$

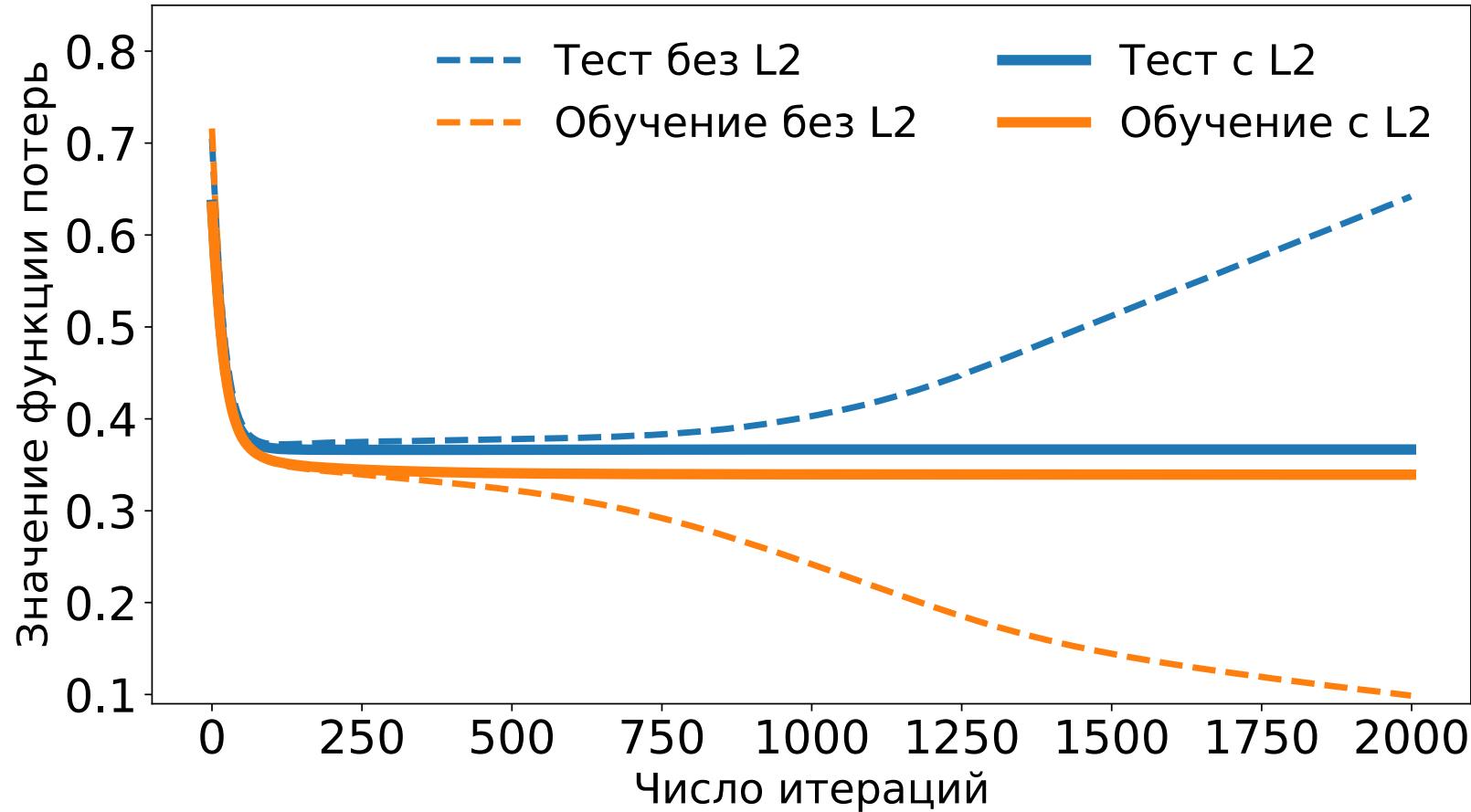
- ▶ Реализован во многих оптимизаторах нейронных сетей
- ▶ Параметр оптимизатора `weight_decay` = $\frac{\beta_2}{2}$

Регуляризация

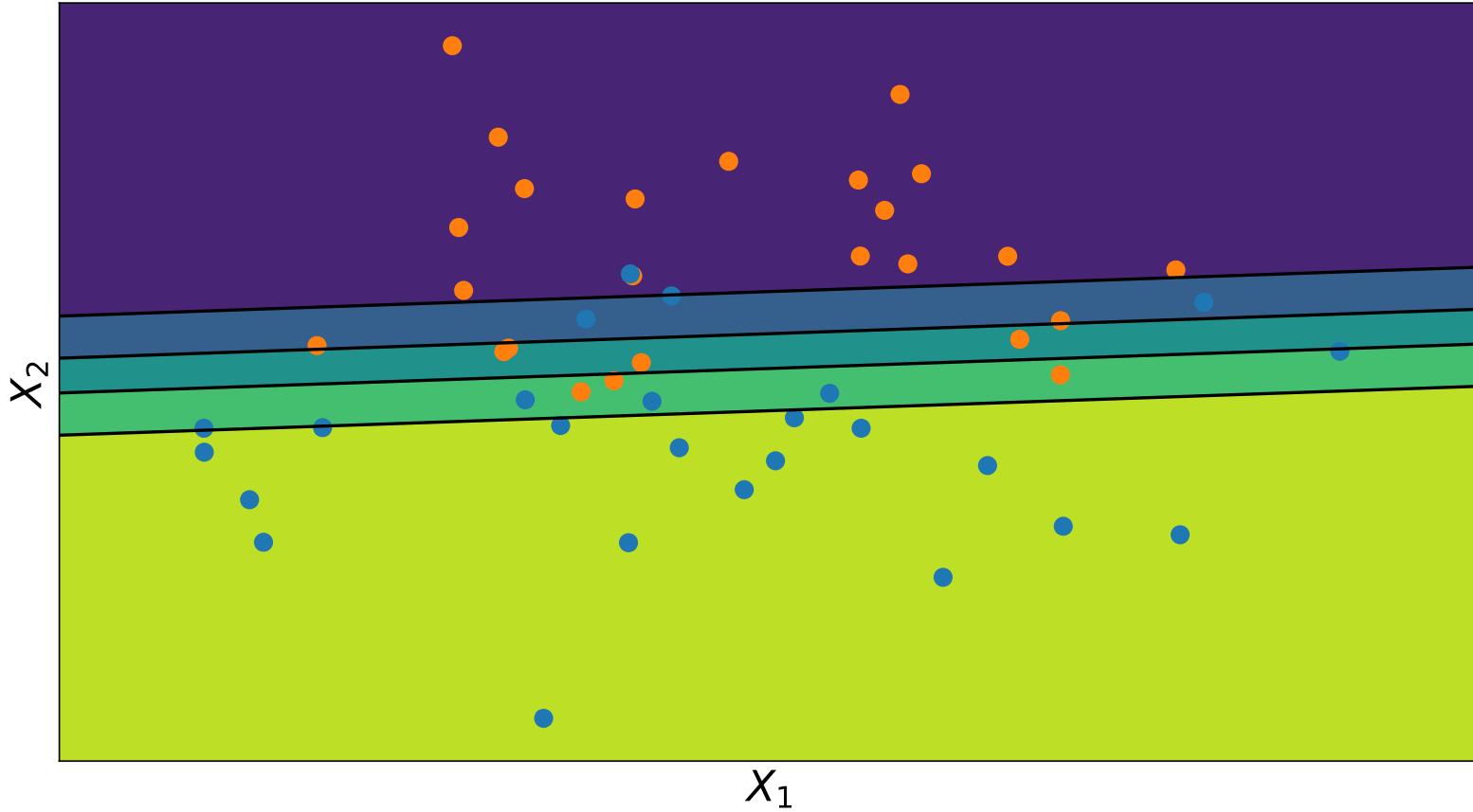
Elastic Net регуляризация:

$$L' = L(\hat{y}, y) + \beta_1 \sum_i |w_i| + \beta_2 \sum_i w_i^2$$

Пример

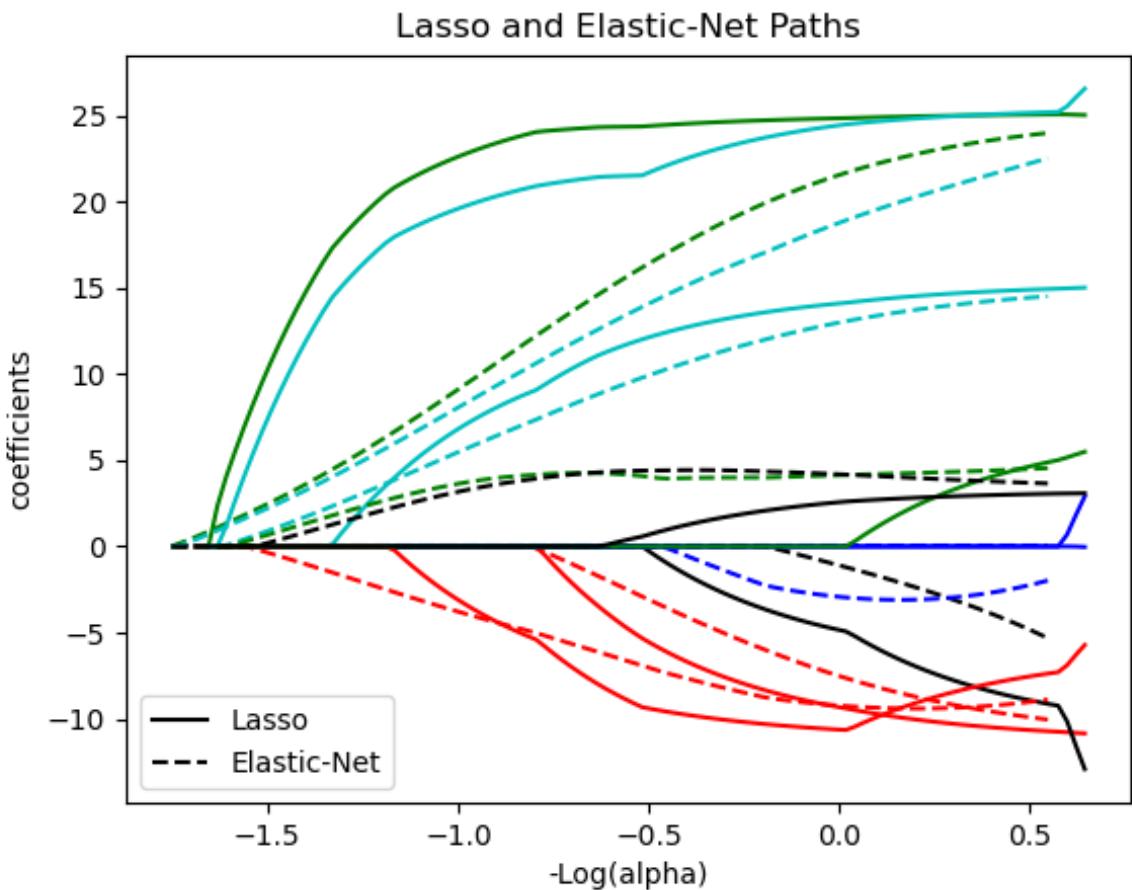


Пример



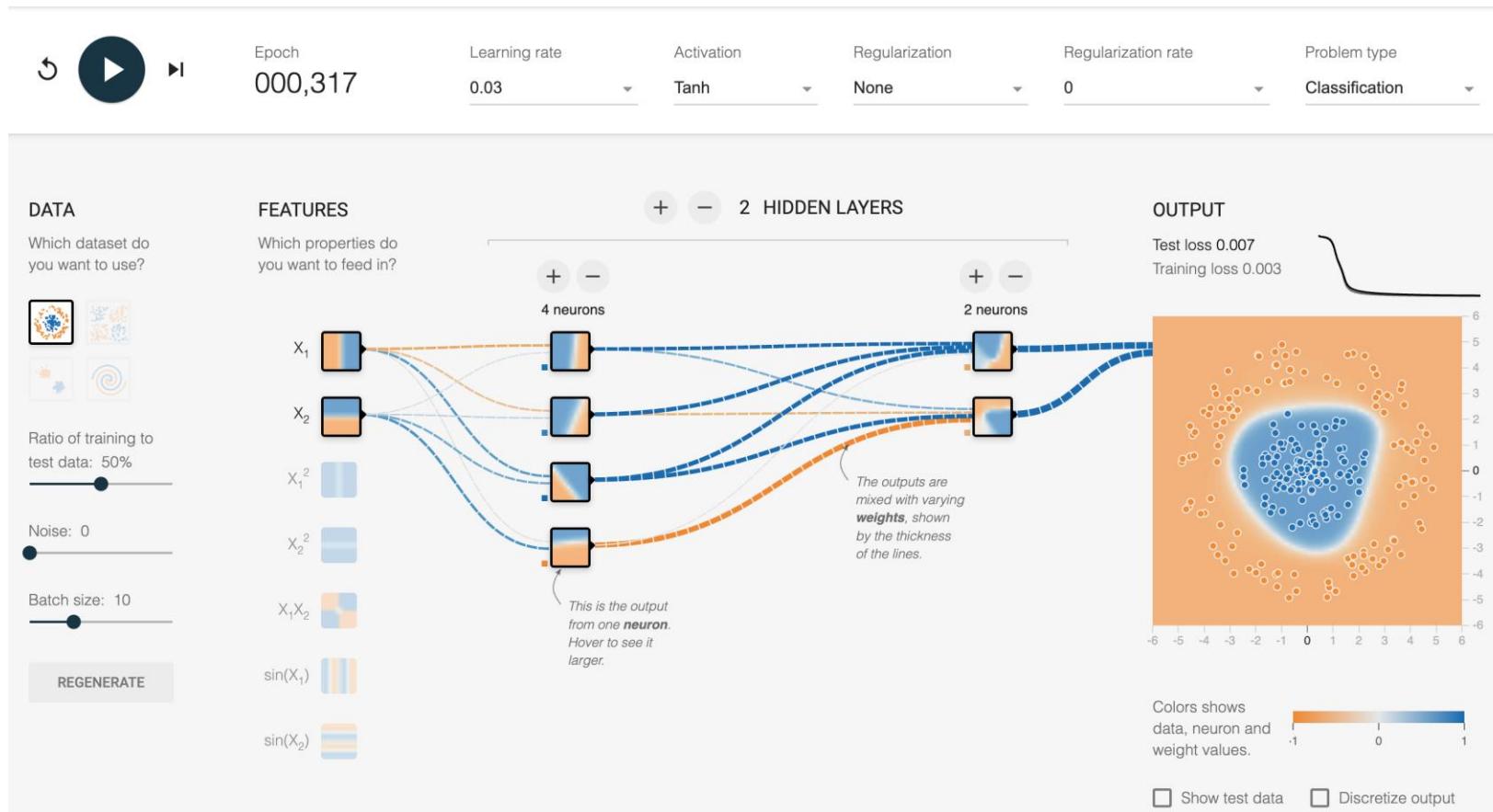
Свойства регуляризации

- ▶ L_2 регуляризация стремится уменьшить веса модели
- ▶ L_1 позволяет проводить **отбор признаков**
- ▶ **L_1 обнуляет веса** для наименее информативных признаков



<https://scikit-learn.org>

Демо с регуляризацией



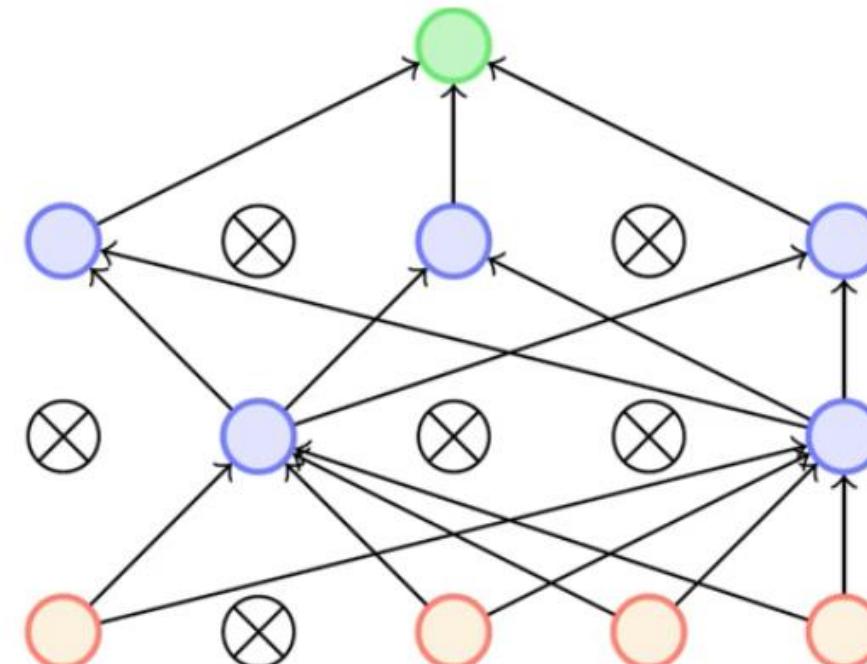
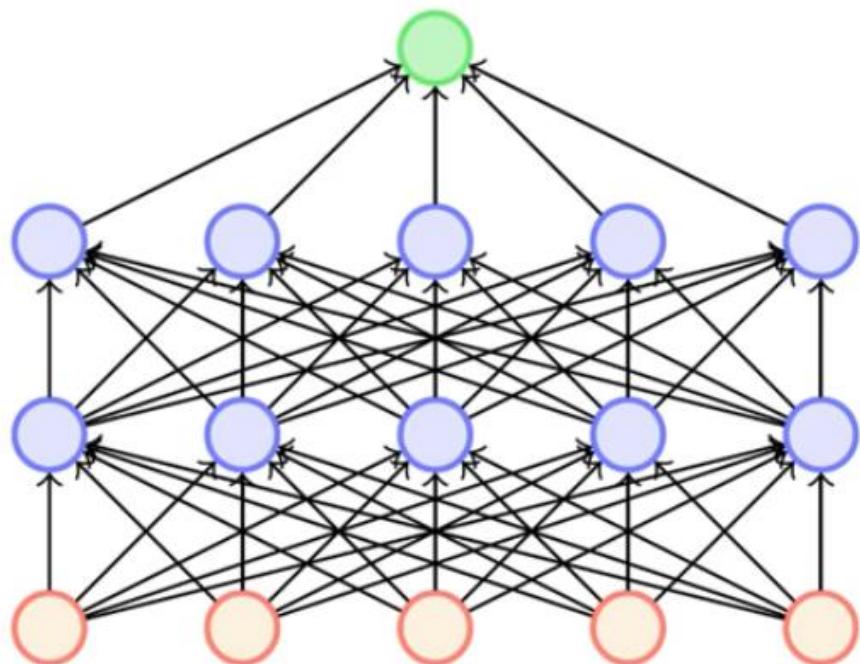
<https://playground.tensorflow.org>

The background of the slide features a complex, glowing network of neurons. These neurons are depicted as thin, translucent tubes in shades of orange, yellow, and white, branching out from small, glowing circular cell bodies. The overall effect is one of a dense, interconnected biological system, possibly representing a brain or a neural network.

Отключение нейронов (dropout)

Отключение нейронов (dropout)

Идея: отключение нейронов с некоторой вероятностью r на каждой итерации обучения сети



Отключение нейронов (dropout)

Обучение сети:

На каждой итерации отключаем каждый нейрон с вероятностью p :

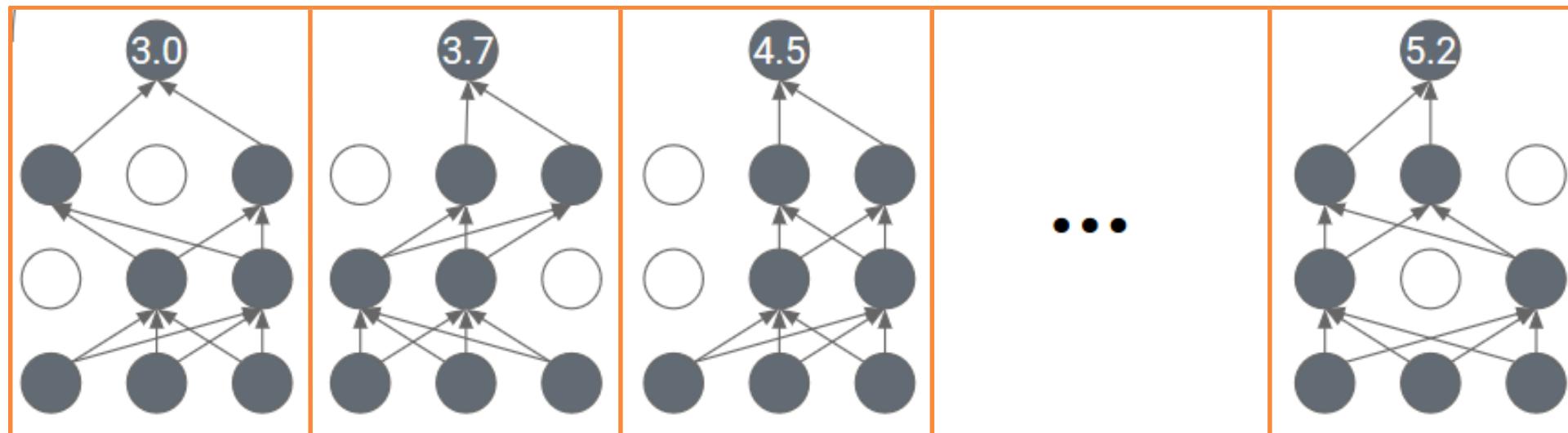
$$a = \xi g \left(\sum_{i=1}^n w_i x_i \right)$$

$$P(\xi = 0) = p$$

$$P(\xi = 1) = 1 - p$$

Пример обучения

На каждой итерации обучения получаем разные архитектуры связей нейронов:



Отключение нейронов (dropout)

Тестирование сети:

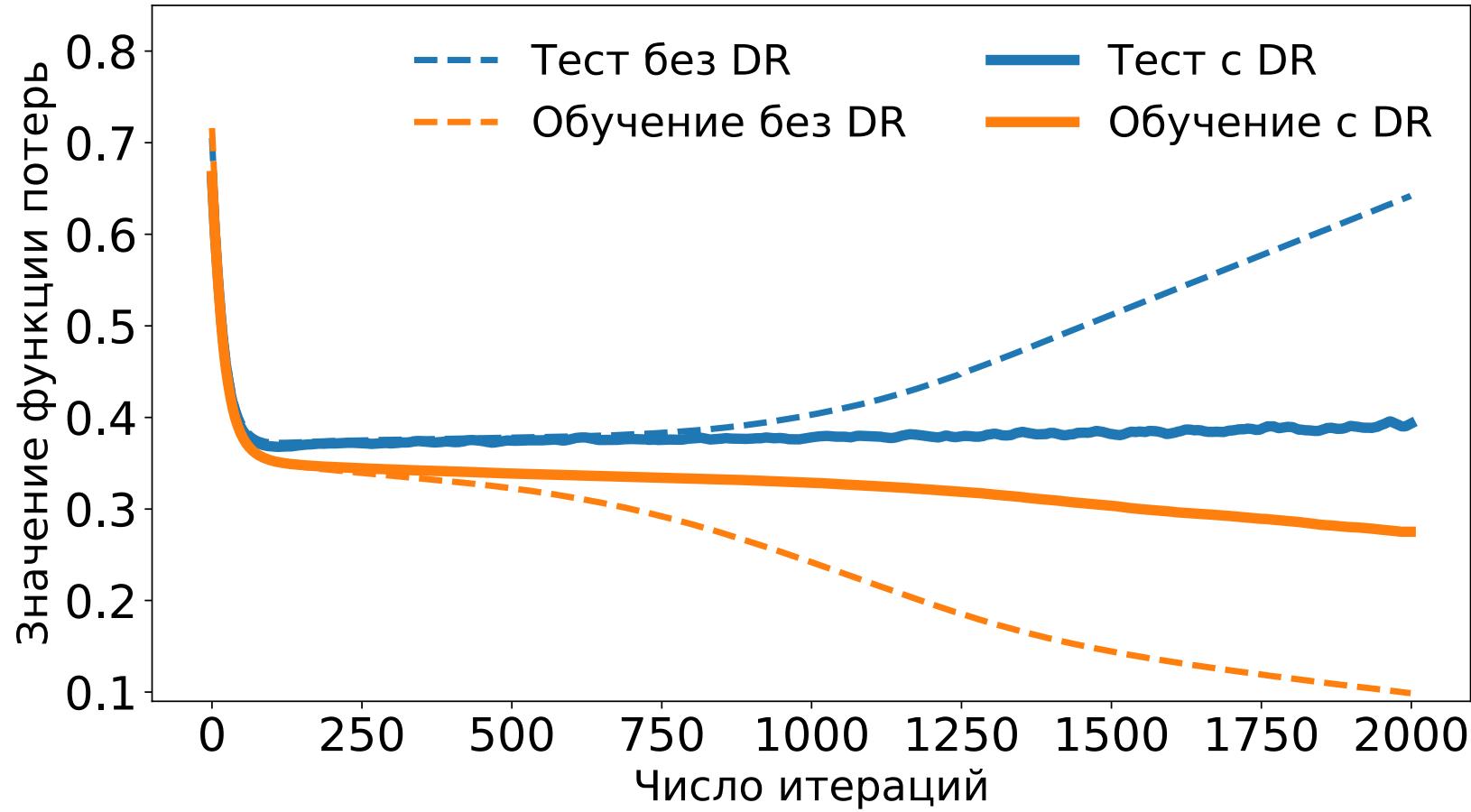
Включаем все нейроны сети, но с поправкой:

$$a = (\mathbf{1} - p)g\left(\sum_{i=1}^n w_i x_i\right)$$

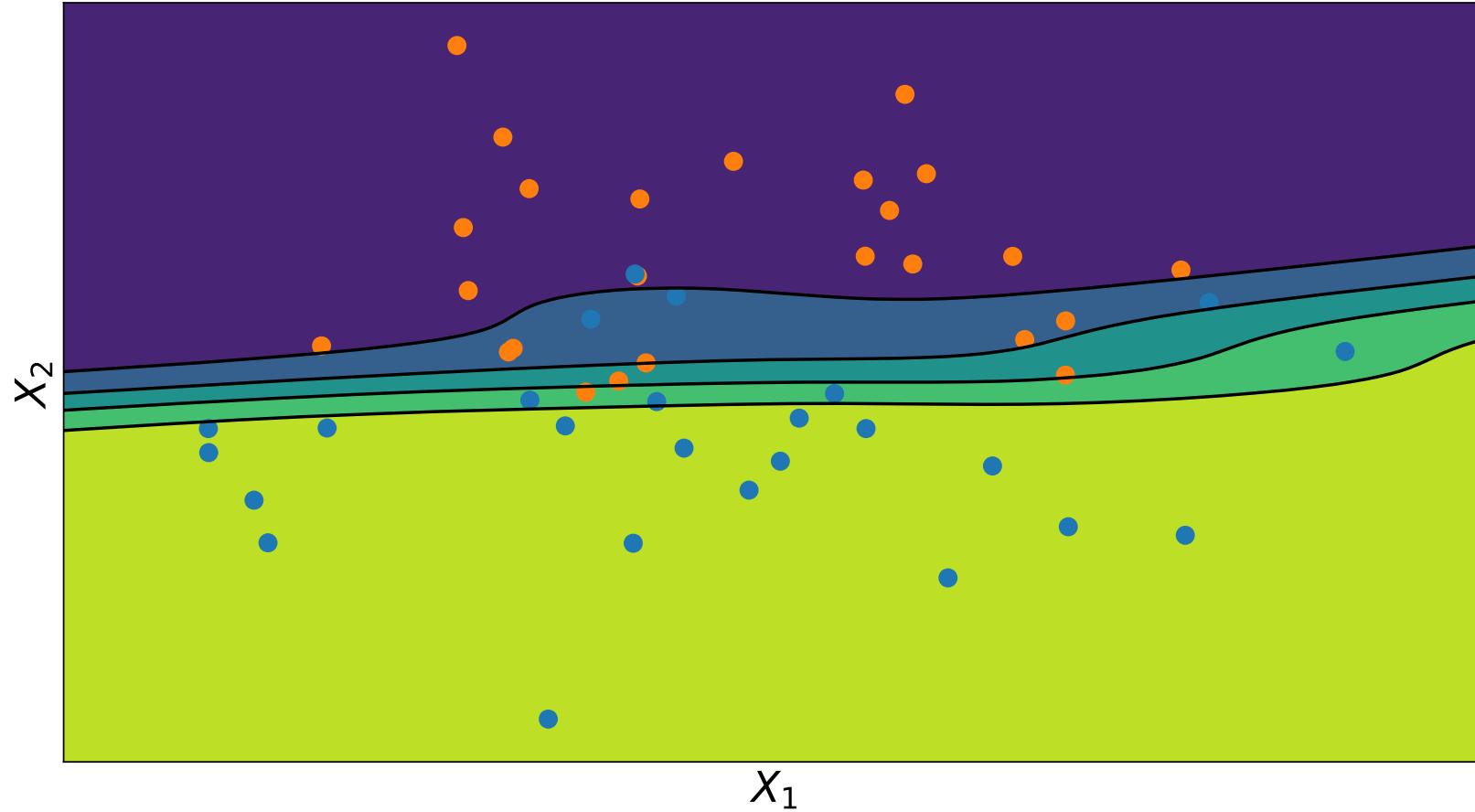
Интерпретация

- ▶ Уменьшение переобучения: разные части сети учат одну и ту же задачу
- ▶ Регуляризация: сеть учится быть устойчивой к потере rN нейронов
- ▶ Ансамбль: усреднение прогнозов сетей с разными связями нейронов

Пример



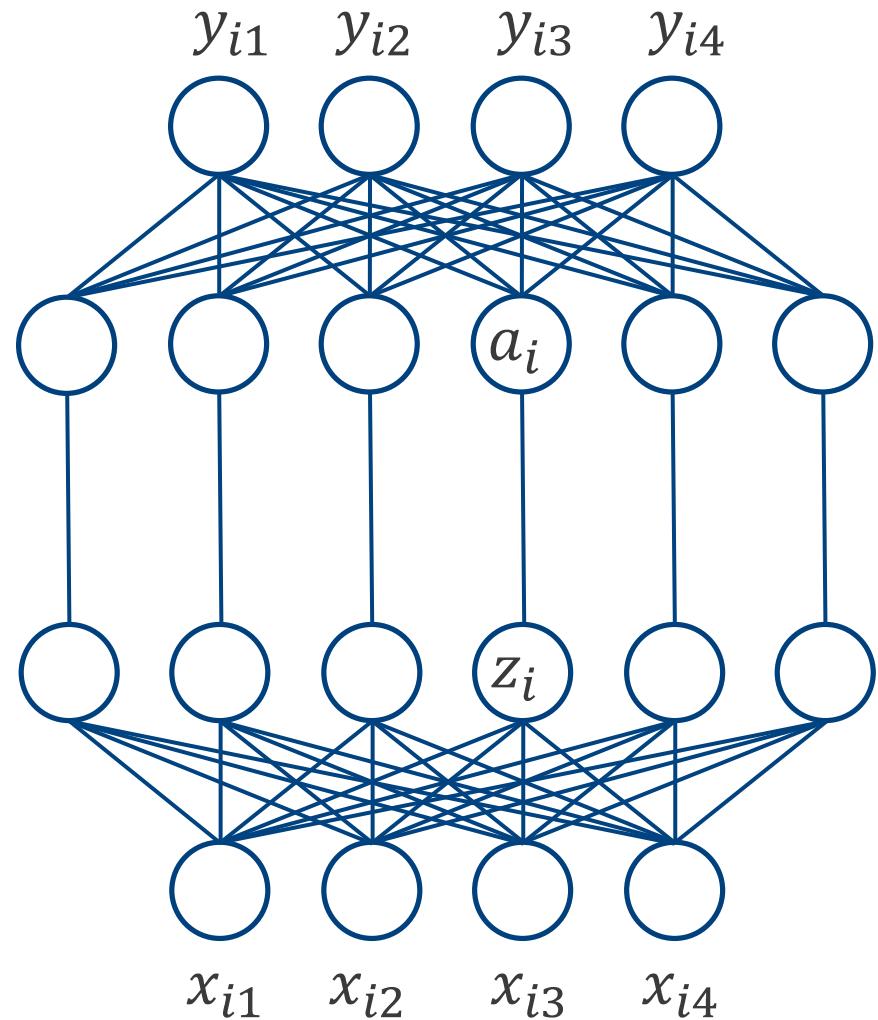
Пример





Пакетная нормализация (batch normalization)

Пакетная нормализация (batch normalization)



$$a_i = g(z_i)$$

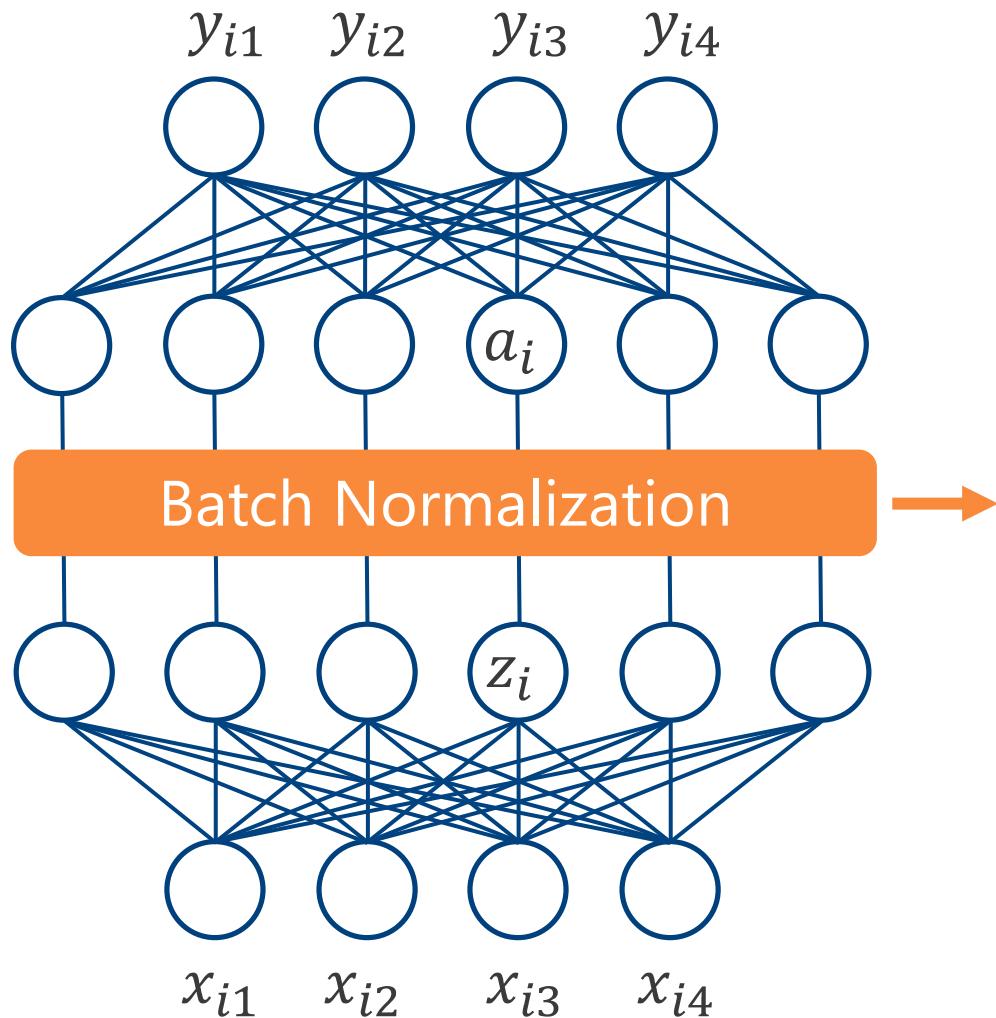
Пакетная нормализация (batch normalization)

Для каждого нейрона вычислим среднее и дисперсию его выхода по пакету данных (batch):

$$\mu = \frac{1}{n_{batch}} \sum_i z_i$$

$$\sigma^2 = \frac{1}{n_{batch}} \sum_i (z_i - \mu)^2$$

Пакетная нормализация (batch normalization)

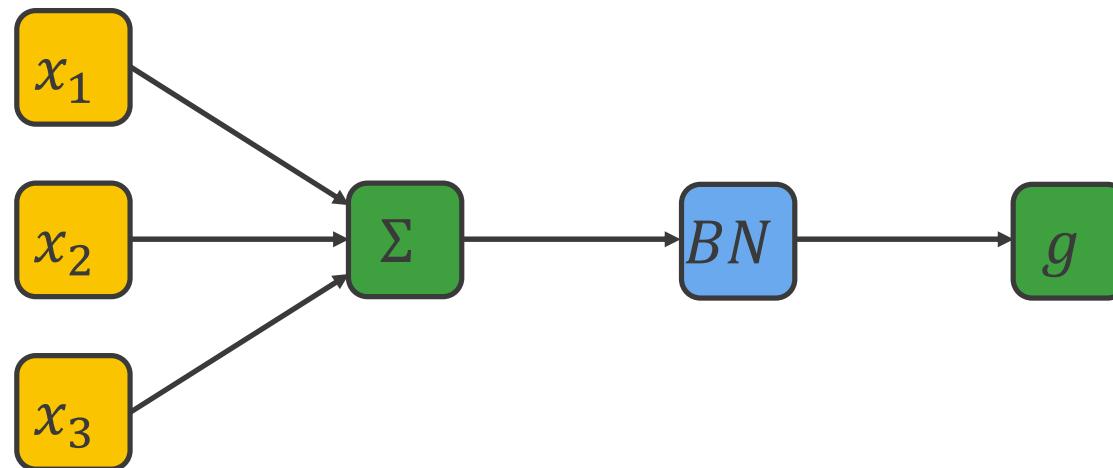


$$\tilde{z}_i = \gamma \frac{z_i - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

γ, β — обучаемые параметры
 ϵ — константа

Вычислительный граф

$$a = g \left(BN \left(\sum_i w_i x_i \right) \right)$$



Пакетная нормализация (batch normalization)

Batch = ($N_{\text{samples in batch}}$, N_{features})

Число объектов в пакете
Нормализация тут!

Число нейронов
в слое

Применение

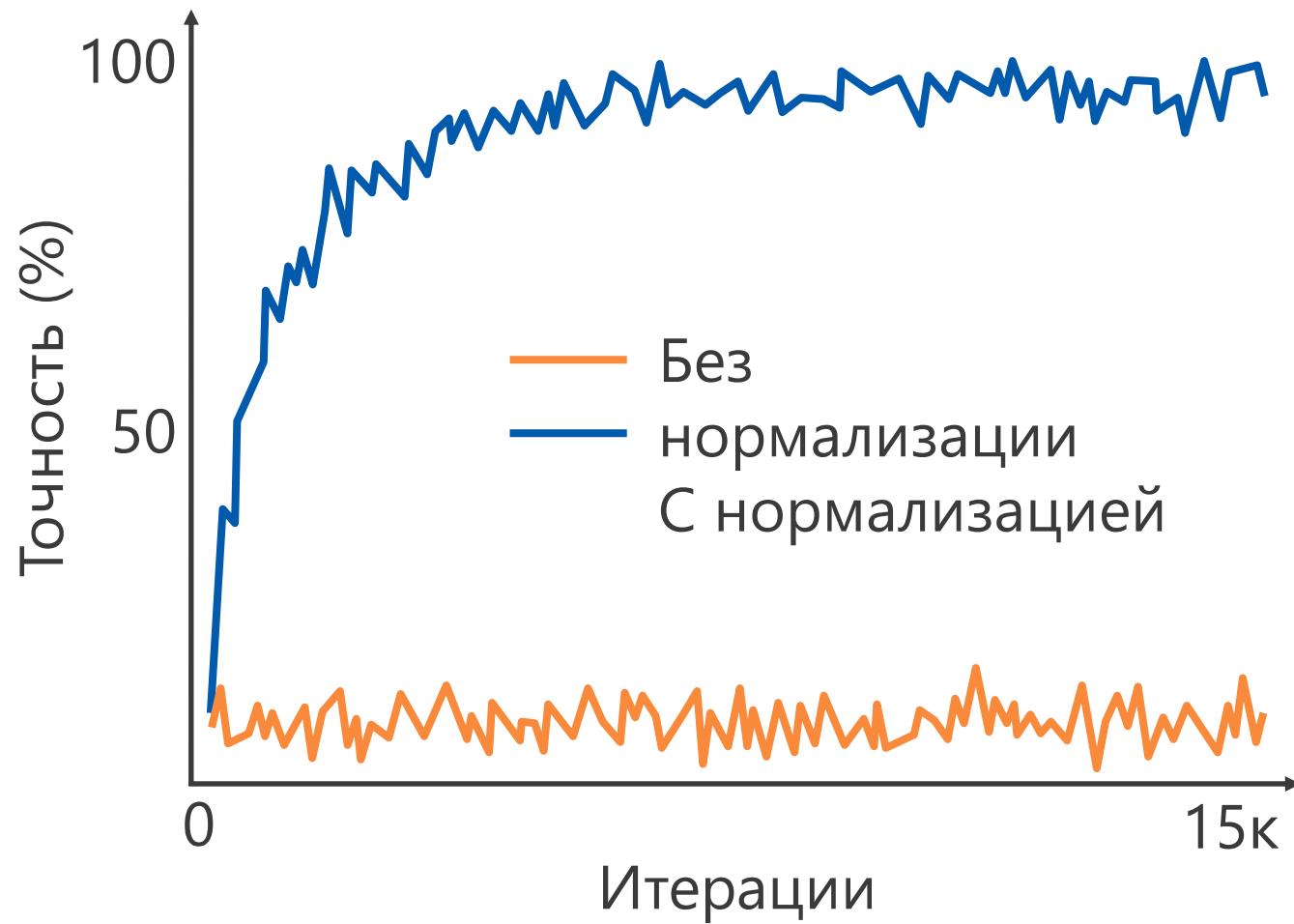
Обучение сети:

- ▶ Считаем μ, σ для каждого пакета данных (batch)

Тестирование сети:

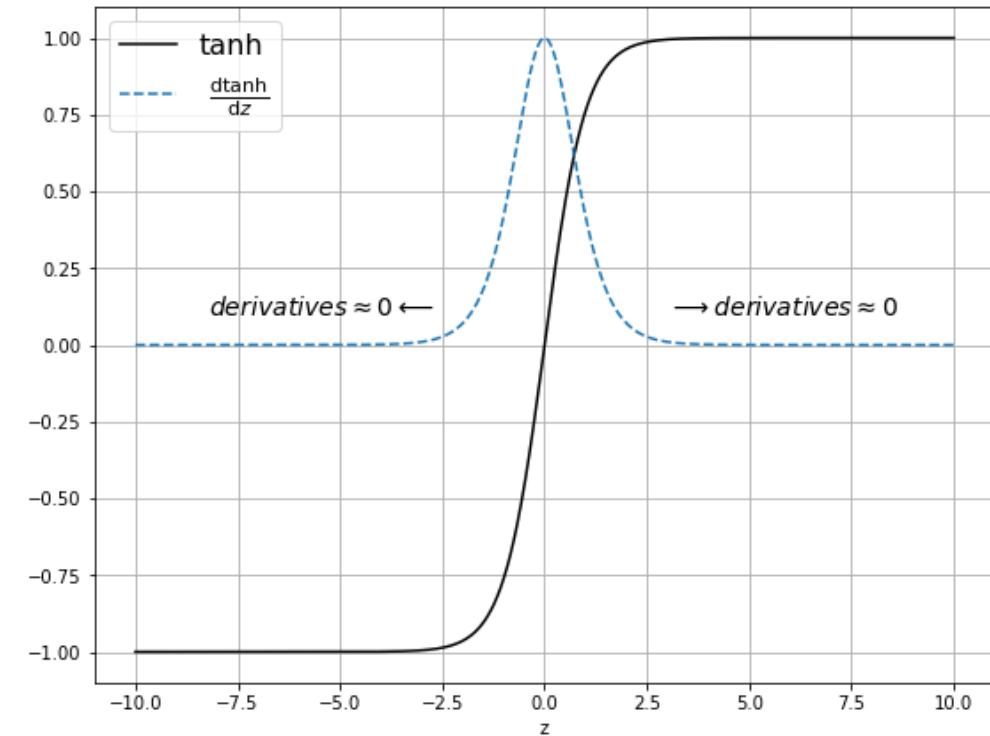
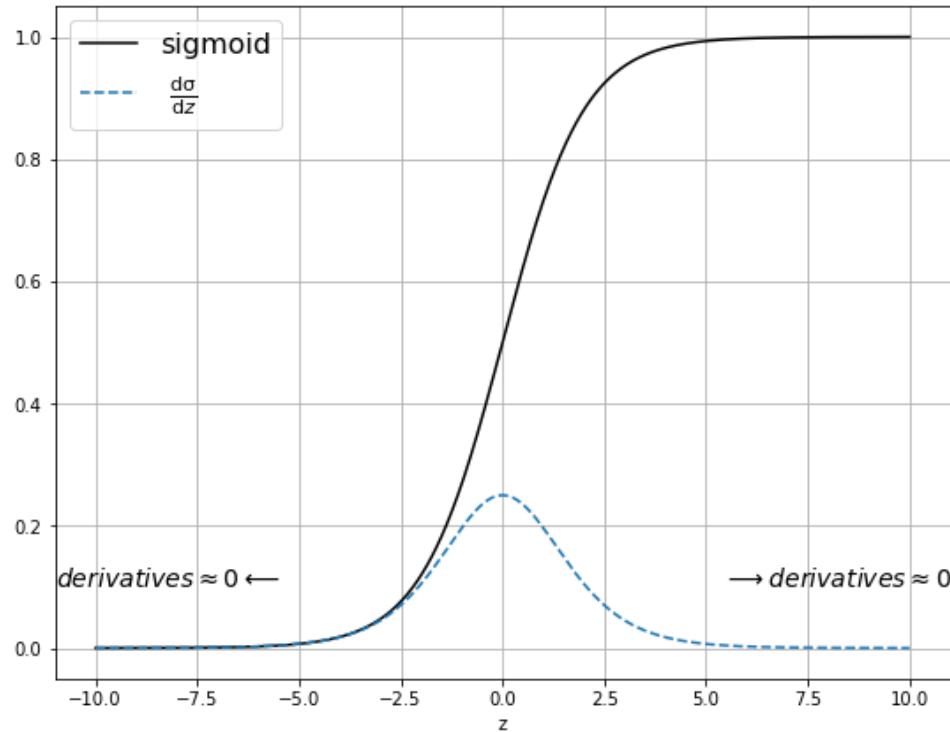
- ▶ Считаем μ, σ на всей выборке данных

Пример



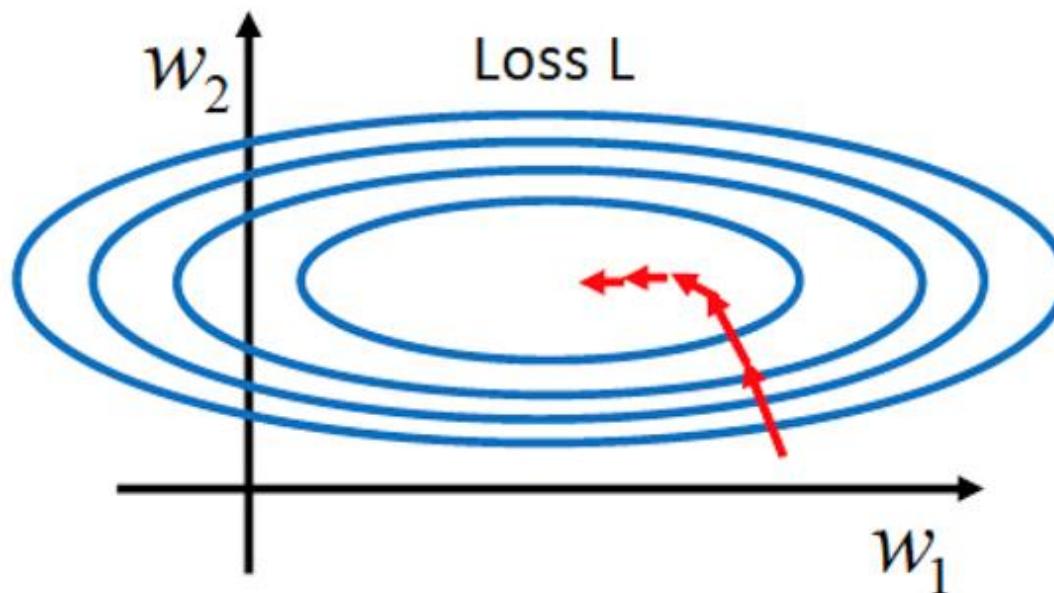
Почему это работает?

Без нормировки выходы нейронов могут попасть в область малых градиентов. В результате обучение сети замедлится.

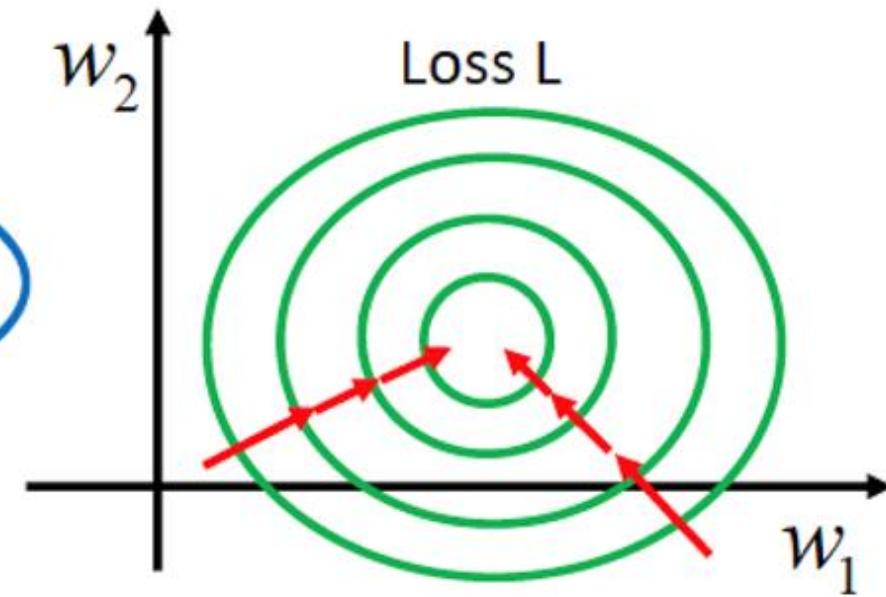


Почему это работает?

Без нормализации

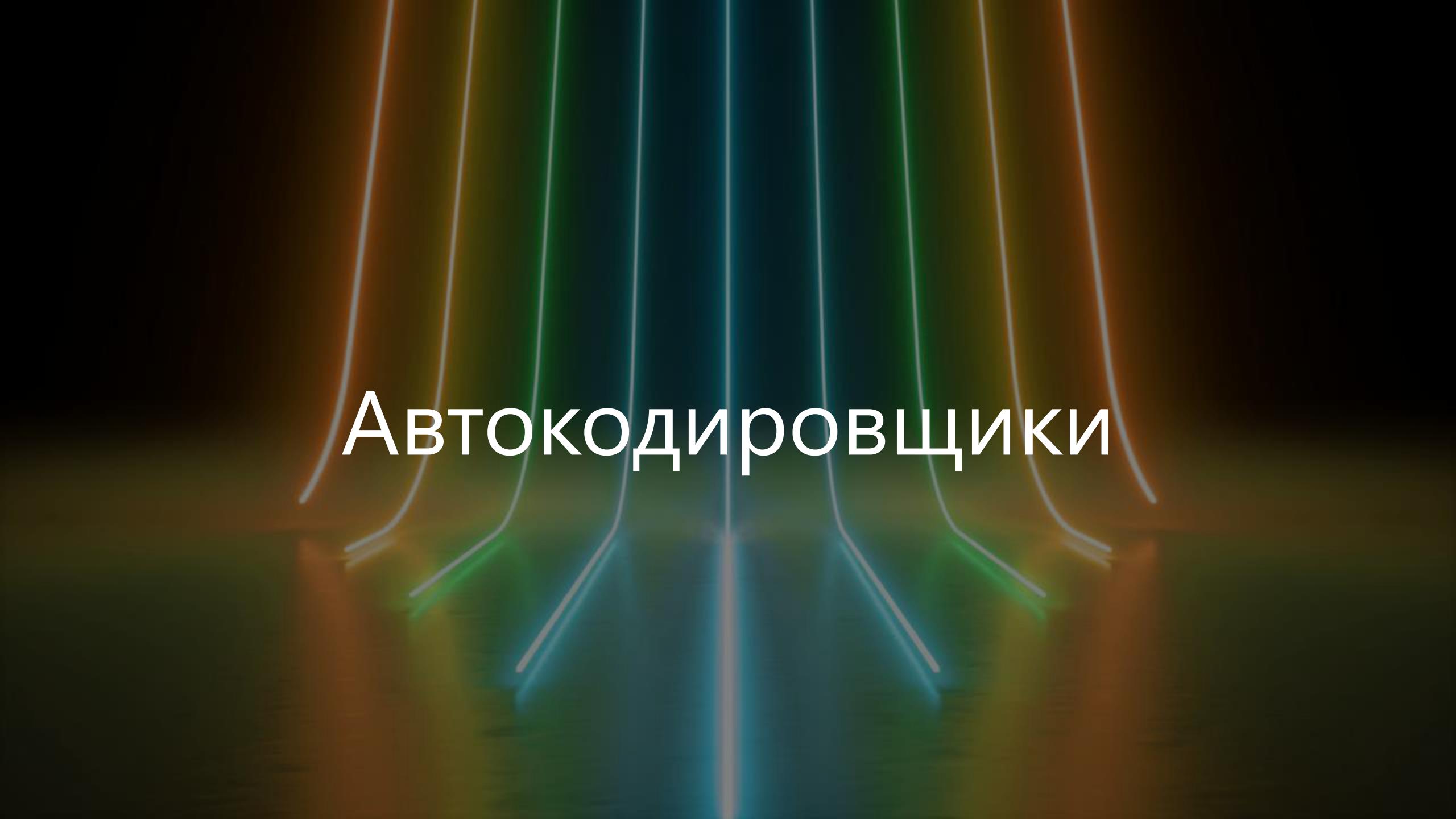


С нормализацией



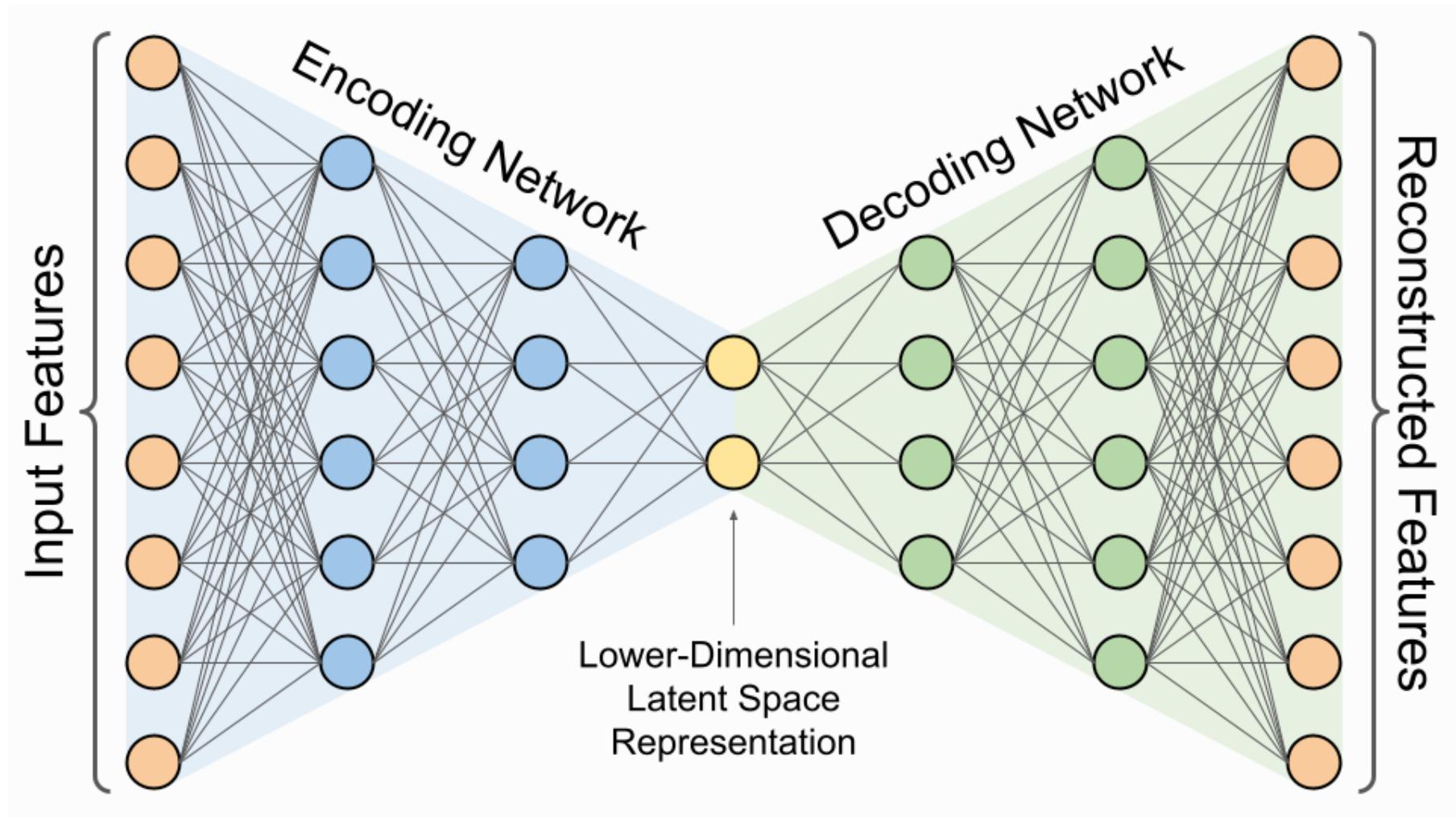
Мотивация

- ▶ Предотвращение сдвига ковариации выходов нейронов
- ▶ Регуляризация
- ▶ Ускорение обучения
- ▶ Повышение качества модели
- ▶ Сильно зависит от размера пакета

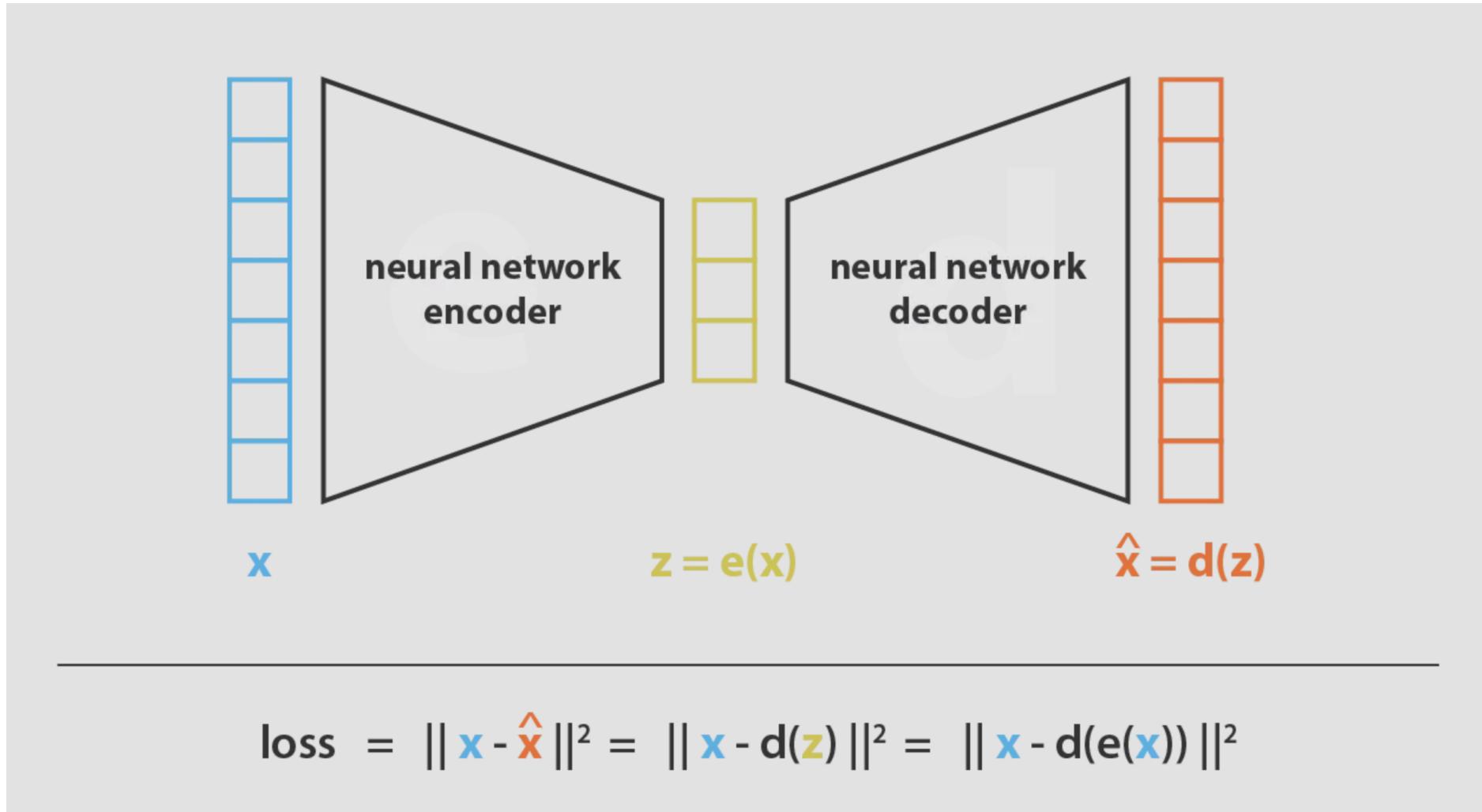


Автокодировщики

Архитектура автокодировщика



ФУНКЦИЯ ПОТЕРЬ

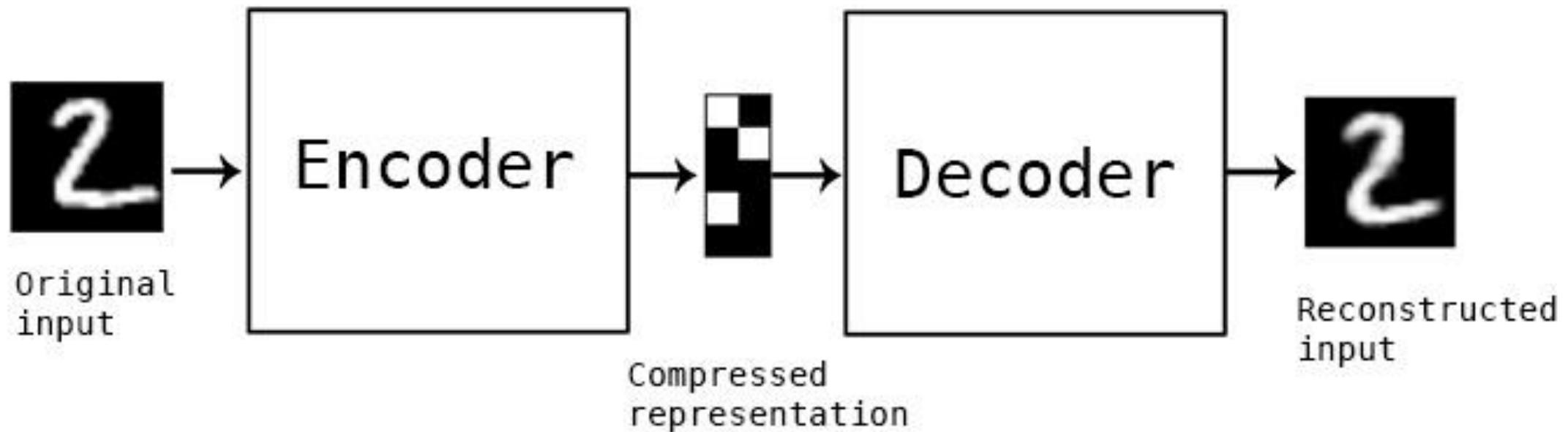


Источник: <https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>

Приложения

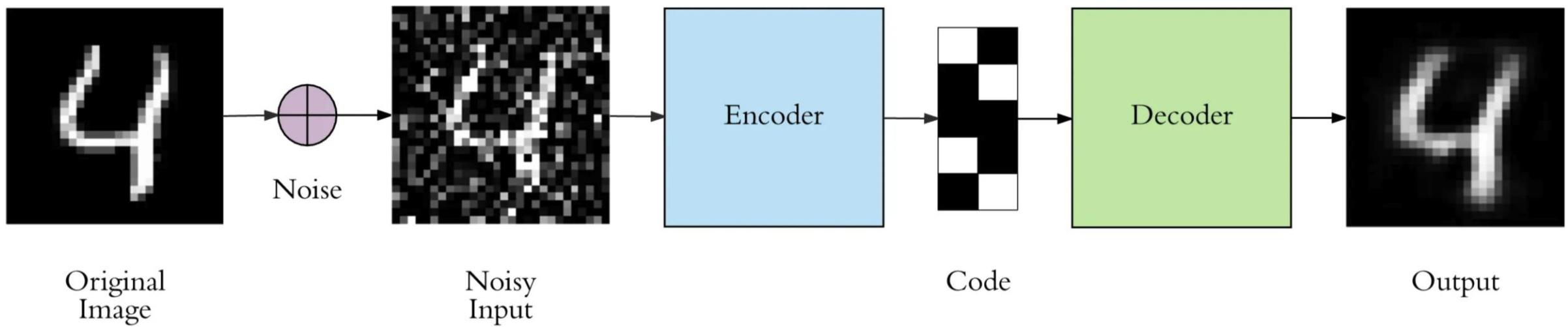
- ▶ Сжатие данных
- ▶ Уменьшение размерности данных
- ▶ Уменьшение зашумленности изображений
- ▶ Генерация новых данных

Сжатие и уменьшение размерности

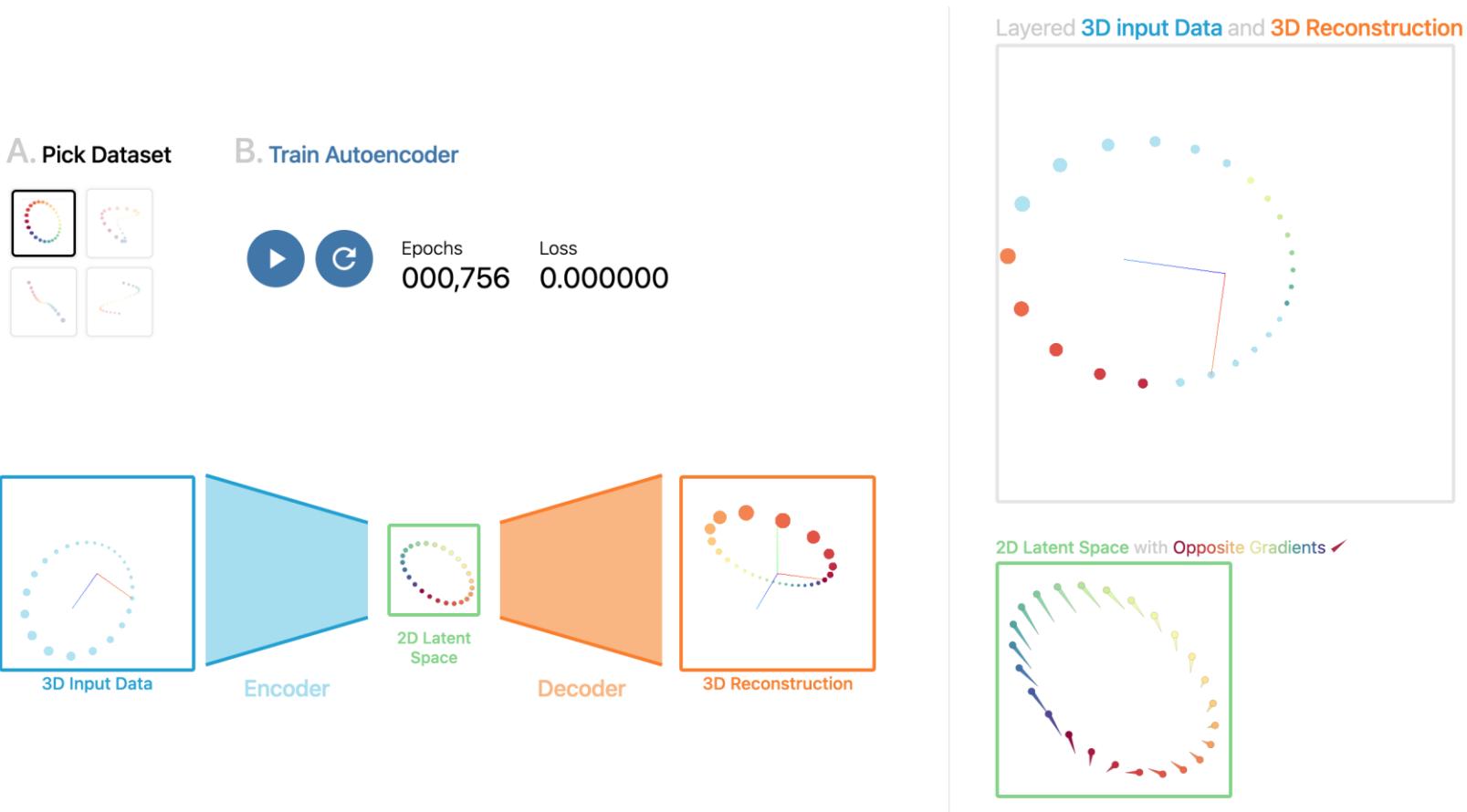


Источник: <https://blog.keras.io/building-autoencoders-in-keras.html>

Уменьшение шума (denoising)



Демо



Демо: <https://introduction-to-autoencoders.vercel.app>

Поставьте свою оценку лекции

