

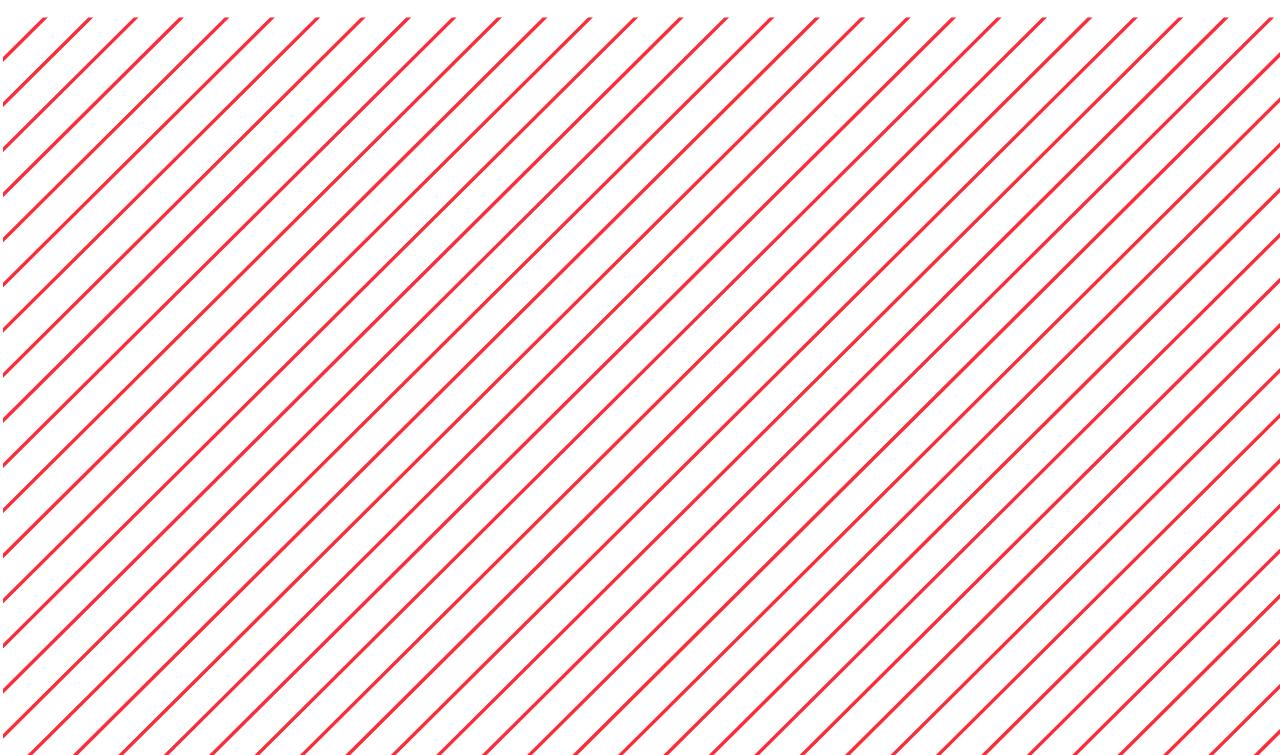
академия
больших
данных



Методы оптимизации в глубоком обучении

Андрей Бояров

Ведущий программист-исследователь в командах Машинного обучения почты и
Машинного зрения Mail.ru



Recap: Многоклассовая классификация

$$p(y = 1 | x) = \frac{1}{1 + \exp(-\langle w, x \rangle)}$$

- Как быть, если классов больше, чем два? Предположим, что классов 3

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

σ = softmax

\vec{z} = input vector

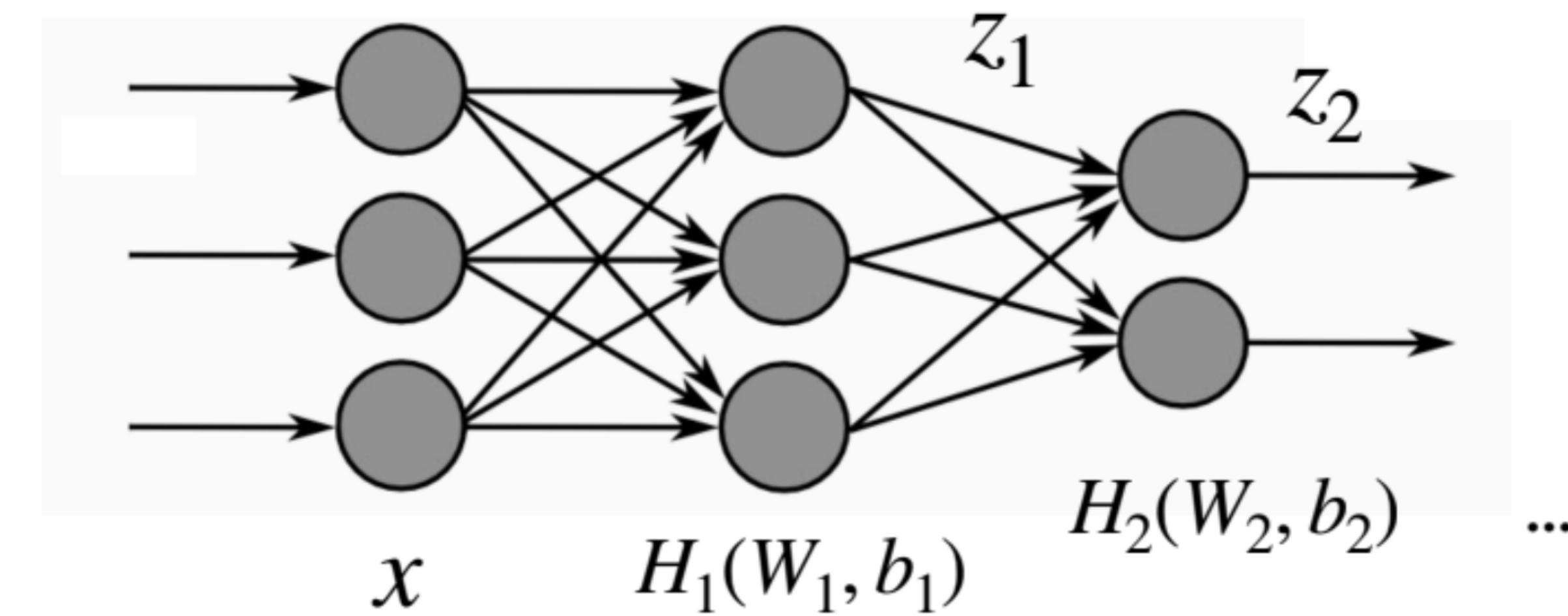
e^{z_i} = standard exponential function for input vector

K = number of classes in the multi-class classifier

e^{z_j} = standard exponential function for output vector

$$\text{softmax} \left(\begin{bmatrix} w_{00} & w_{01} \\ w_{10} & w_{11} \\ w_{20} & w_{21} \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \right) = \begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix}$$

Recap: многослойная сеть



$$z_1 = \text{Activation}(H_1(x)) = \text{Activation}(W_1 \times x + b_1)$$

$$z_2 = \text{Activation}(H_2(z_1)) = \text{Activation}(W_2 \times z_1 + b_2)$$

$$W^{t+1} = W^t - \eta \times \nabla Loss_W$$

Recap: Метод обратного распространения ошибки

x = input

$z = Wx + b_1$

$h = \text{ReLU}(z)$

$\theta = Uh + b_2$

$\hat{y} = \text{softmax}(\theta)$

$J = CE(y, \hat{y})$

Какие градиенты нам нужно посчитать, чтобы обновить веса слоёв?

$$\frac{\partial J}{\partial U}$$

$$\frac{\partial J}{\partial b_2}$$

$$\frac{\partial J}{\partial W}$$

$$\frac{\partial J}{\partial b_1}$$

Какие эффективно посчитать эти градиенты?

Ответ: chain rule

$$\frac{\partial J}{\partial U} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \theta} \frac{\partial \theta}{\partial U}$$

$$\frac{\partial J}{\partial b_2} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \theta} \frac{\partial \theta}{\partial b_2}$$

Recap: backprop одномерный случай

Backpropagation: a simple example

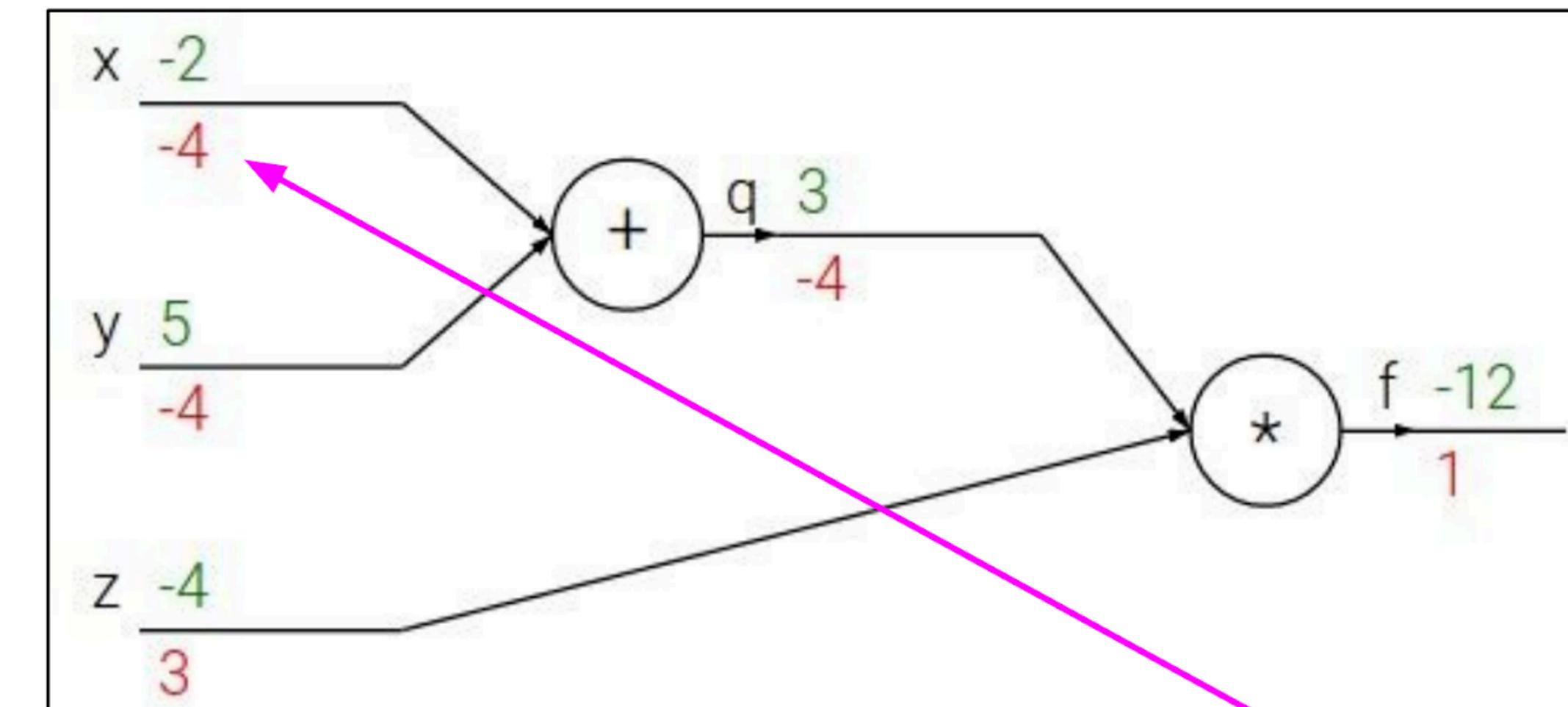
$$f(x, y, z) = (x + y)z$$

e.g. $x = -2$, $y = 5$, $z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

Upstream
gradient Local
gradient

$$\frac{\partial f}{\partial x}$$

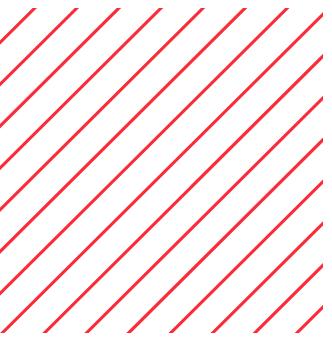
Recap: Якобиан

$$\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

$$\mathbf{f}(\mathbf{x}) = [f_1(x_1, \dots, x_n), f_2(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)]$$

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} w_{01} & w_{02} & w_{03} \\ w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

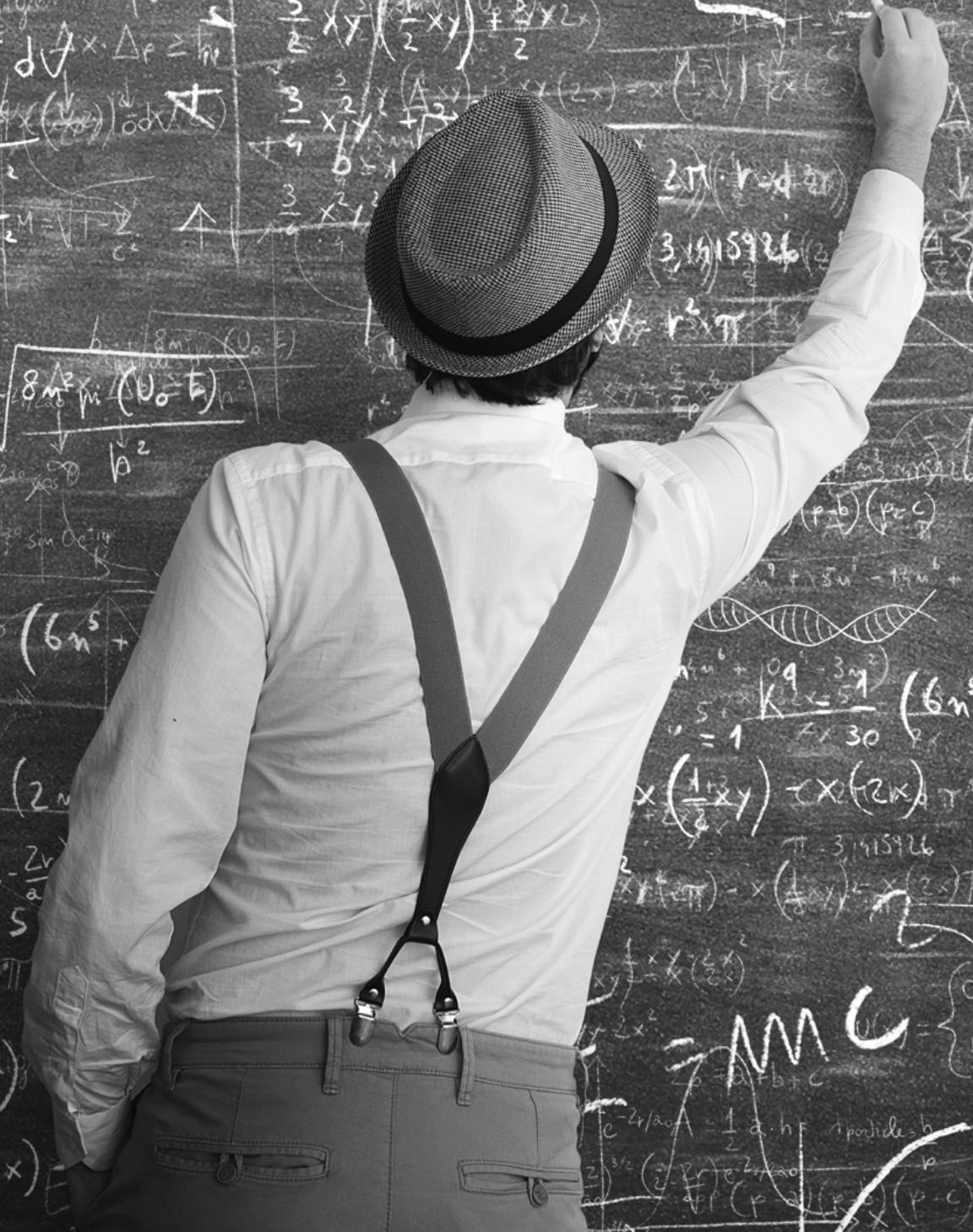
$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$



План лекции

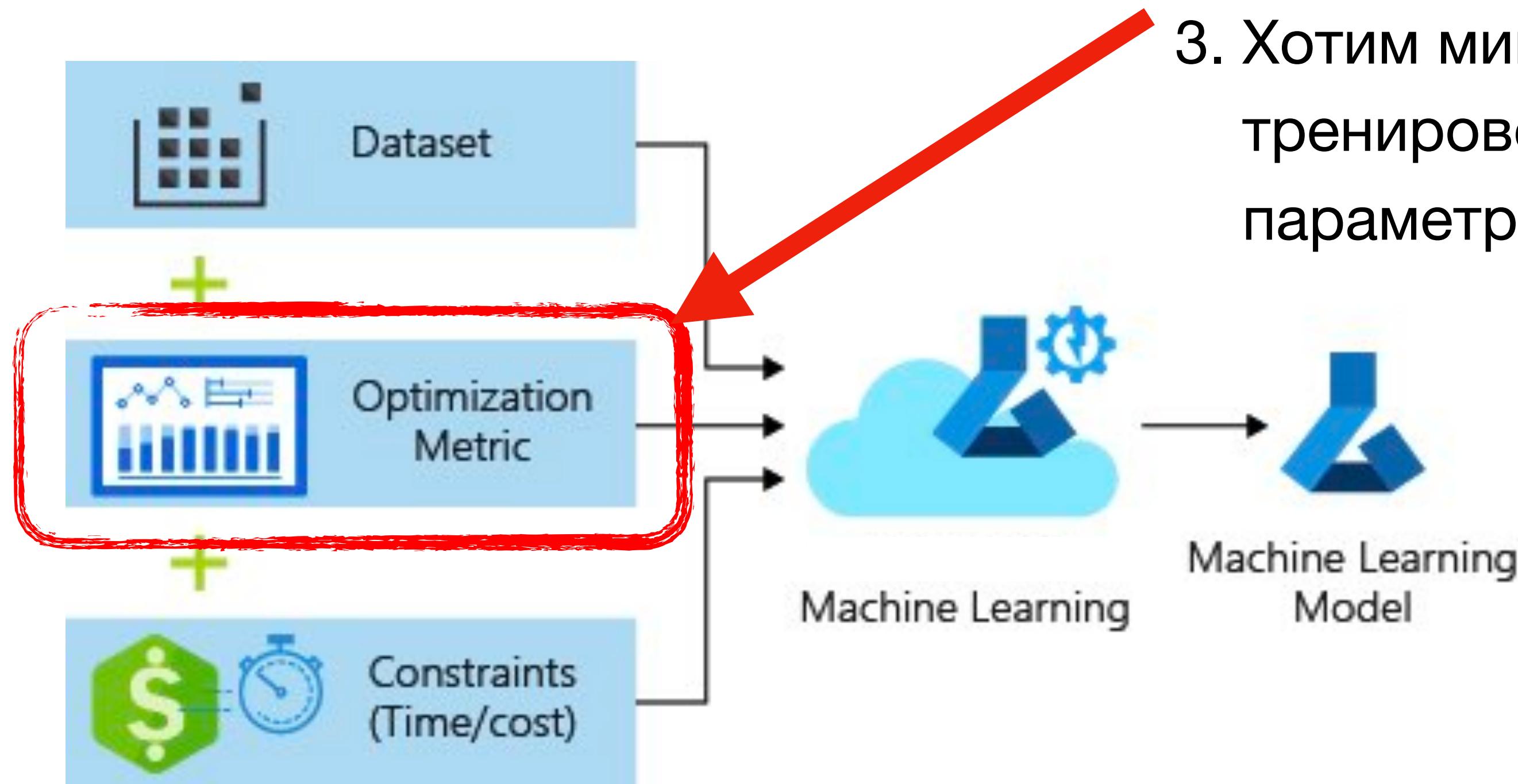
1. Постановка задачи оптимизации в машинном обучении
2. Сложности оптимизации глубоких нейронных сетей
3. Методы оптимизации
4. Learning rate
5. Регуляризация
6. Адаптация нейронных сетей с помощью оптимизации

Постановка задачи оптимизации в машинном обучении

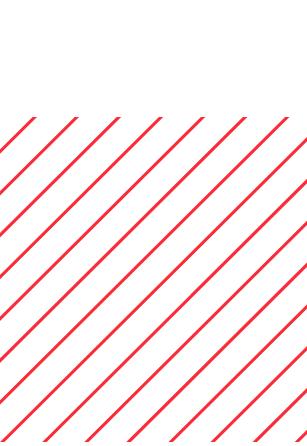


$$\begin{aligned}
 & \text{Постановка задачи оптимизации в машинном обучении}
 \end{aligned}$$

Machine learning pipeline



1. Параметризованная модель
2. Функция потерь
3. Хотим минимизировать функцию потерь на тренировочных данных путём подбора параметров модели



Примеры функций потерь

- Классификация
 - Бинарная: $l(\hat{f}, X, Y) = I\{\hat{f}(X) \neq Y\}$ ($\hat{f}(X), Y \in \{-1, 1\}$)
 - Hinge loss: $l(\hat{f}, X, Y) = \max\{0, 1 - Y\hat{f}(X)\}$
 - Softmax
- Регрессия: $l(\hat{f}, X, Y) = (\hat{f}(X) - Y)^2$ ($\hat{f}(X), Y \in \{-1, 1\}$)

Empirical risk minimization (ERM)

Введём некоторую **меру на семействе алгоритмов** F :

$$L(\hat{f}) = E_{X,Y} \left[l\left(\hat{f}_{\{x_i, y_i\}_{i=1}^n}, X, Y\right) \middle| \{x_i, y_i\}_{i=1}^n \right], \text{ for } \hat{f} = \hat{f}_{\{x_i, y_i\}_{i=1}^n} \in F$$

Оценка по принципу **минимизации эмпирического риска**:

$$\hat{f}_{ERM} = \arg \min_{f \in F} \sum_{i=1}^n l(f, x_i, y_i), \quad L(f_*) = \inf_{f \in F} L(f)$$

Статистическая теория обучения:

$$P \left(L(\hat{f}_{ERM}) - L(f_*) \leq C \sqrt{\frac{V}{n}} + \sqrt{\frac{2 \ln(\sigma^{-1})}{n}} \right) \geq 1 - \sigma,$$

где C - константа,

V - параметр семейства F , σ - уровень доверия

V – размерность Вапника-Червоненкиса (VC)

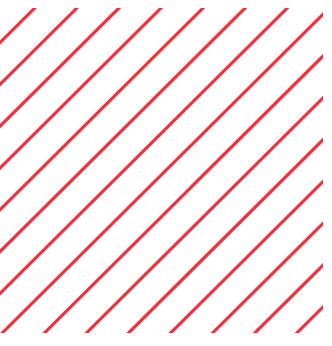
Maximum Likelihood Estimation (MLE)

Пусть $x_k, k = 1, \dots, n$ - н.о.р. с плотностью распределения $p_x(x|\theta)$

Оценка по методу максимума правдоподобия:

$$\begin{aligned}\tilde{\theta}_{MLE}(x) &= \arg \max_{\theta} p_x(x|\theta) = \arg \max_{\theta} \ln p_x(x|\theta) = \\ &= \arg \max_{\theta} \ln \prod_{i=1}^n p_{x_i}(x_i|\theta) = \arg \max_{\theta} \sum_{i=1}^n \ln p_{x_i}(x_i|\theta),\end{aligned}$$

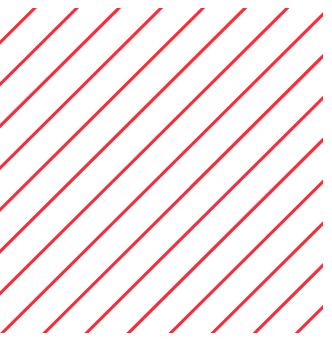
1. MLE-оценка является **состоятельной** при выполнении условий на $p_x(x|\theta)$
2. Из всех состоятельных оценок MLE-оценка является **наилучшей** (в смысле выполнения равенства Рао-Крамера)



Пример: метод наименьших квадратов

$$y_i = kx_i + b + \varepsilon_i \quad \varepsilon_i \in N(0, \sigma^2), \quad \theta = (k, b)^T,$$

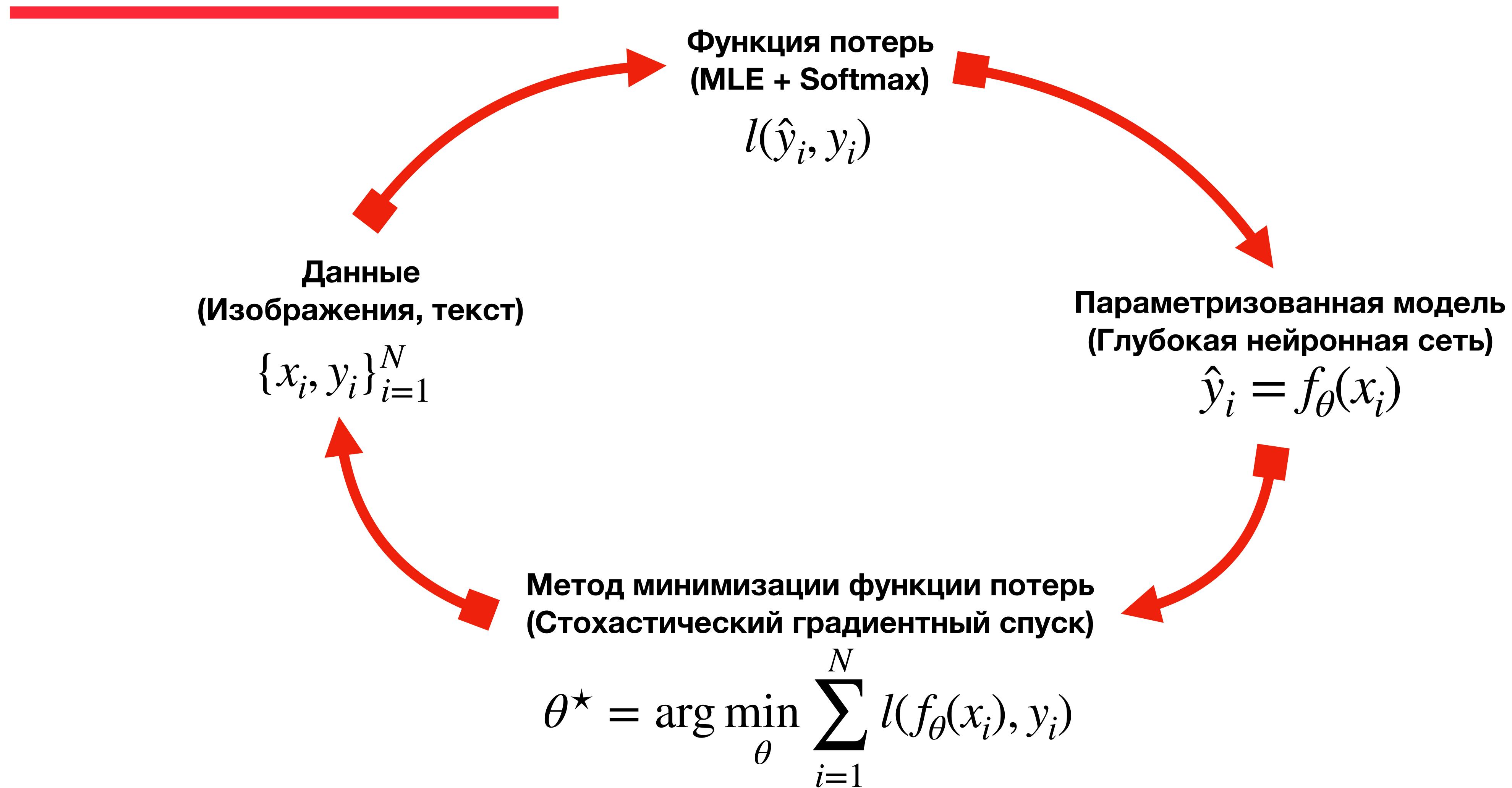
$$\mathbf{x} = A\theta + \varepsilon, \quad \mathbf{x} = \{y_i\}_{i=1}^n, \quad A = \begin{pmatrix} x_1 & \dots & x_n \\ 1 & \dots & 1 \end{pmatrix}^T, \quad \tilde{\theta}_{MLE}(\mathbf{x}) = \arg \min_{\theta} \|A\theta - \mathbf{x}\|_2^2.$$



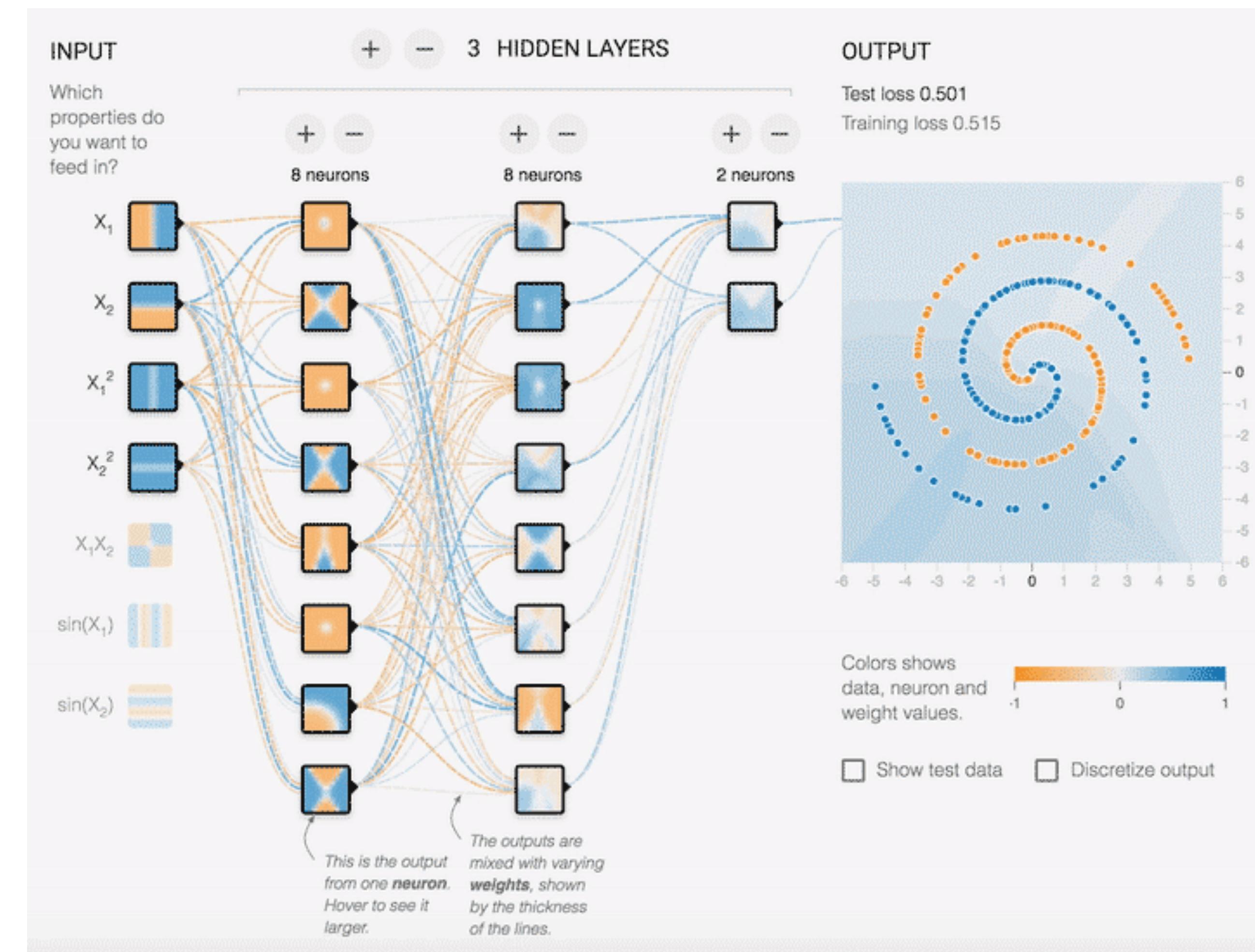
The No Free Lunch Theorem (Wolpert, 1996)

- Любой алгоритм классификации имеет в среднем **по всем возможным распределениям**, порождающим данные, **одинаковую вероятность ошибки** при классификации ранее не наблюдавшейся точки данных.
- К счастью, нам не нужно строить **универсальный** алгоритм машинного обучения.
- Нам нужно понять, какое **распределение данных** лучше всего подходит к решаемой нами задаче, и какой **алгоритм машинного обучения** лучше всего подойдёт для таких данных.

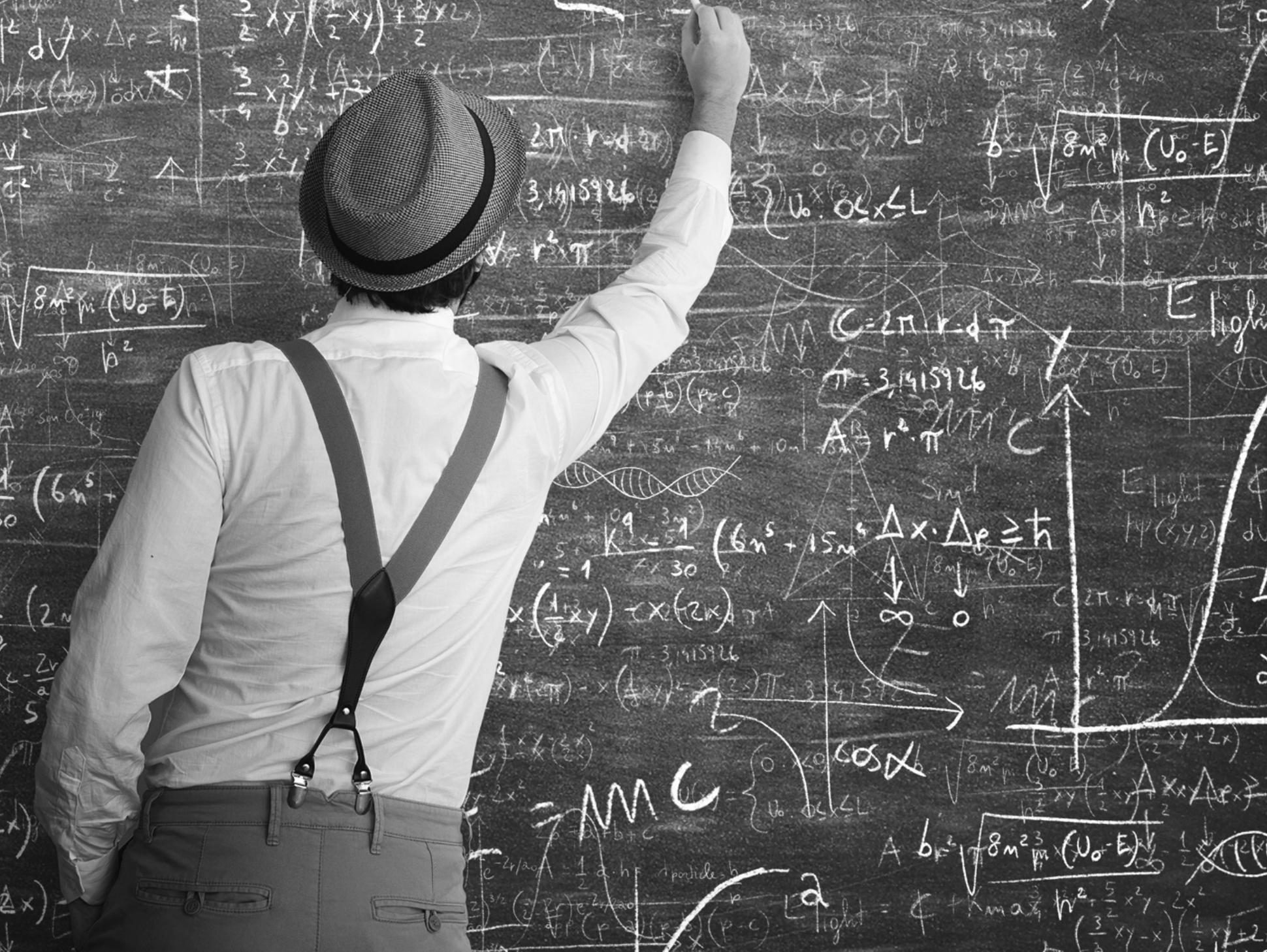
Задача оптимизации в машинном обучении



Пример: оптимизация параметров нейронной сети

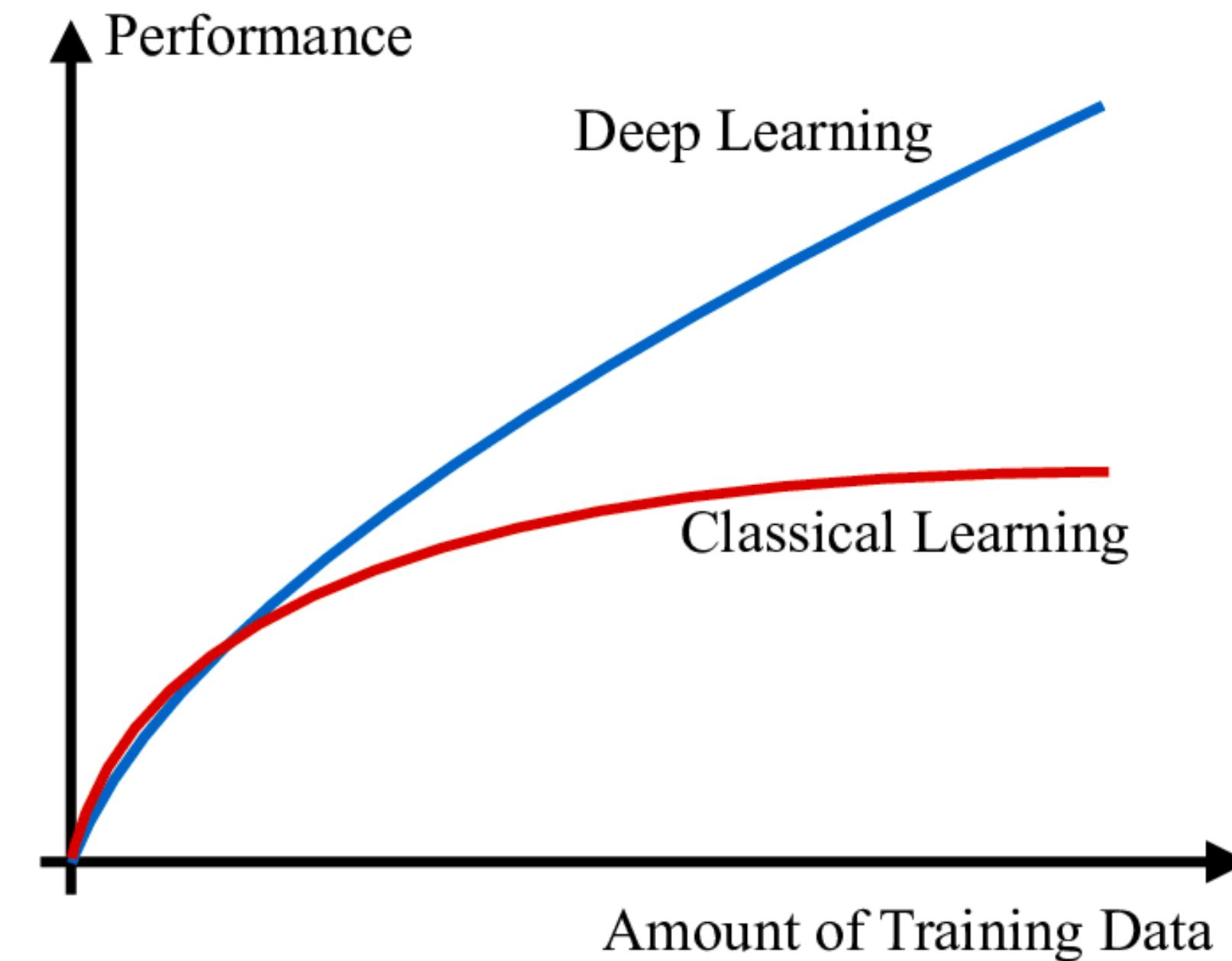


Сложности оптимизации глубоких нейронных сетей



Универсальная теорема аппроксимации (Cybenko, 1989)

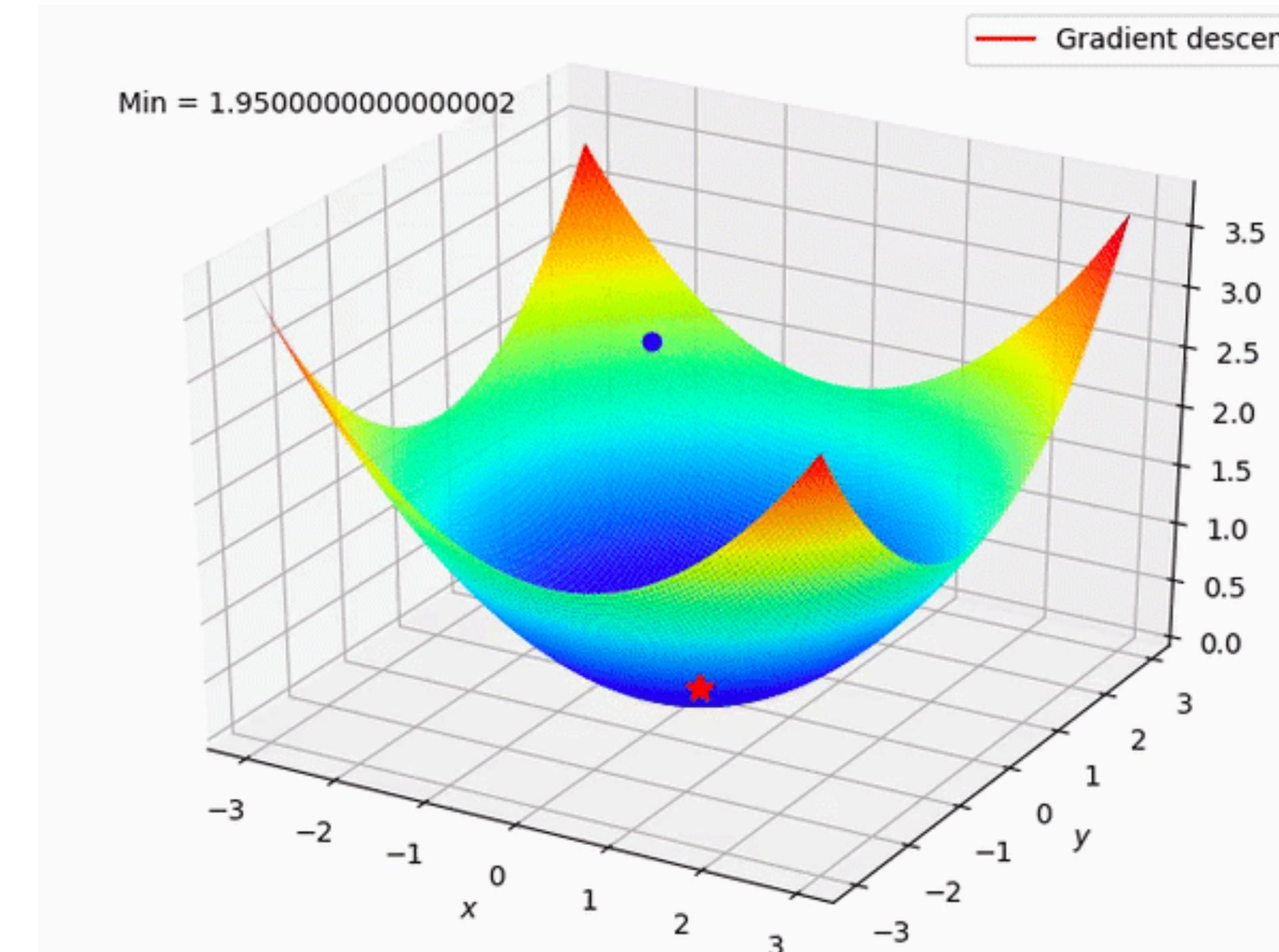
- Нейронная сеть прямого распространения сигнала с одним скрытым слоем может аппроксимировать **любую непрерывную функцию** многих переменных с **любой точностью**.
- Но для этого может понадобится **огромное количество нейронов** в этом скрытом слое.
- Поэтому используют **несколько слоёв**, что влечёт ряд сложностей.



Оптимизационная поверхность: логистическая регрессия

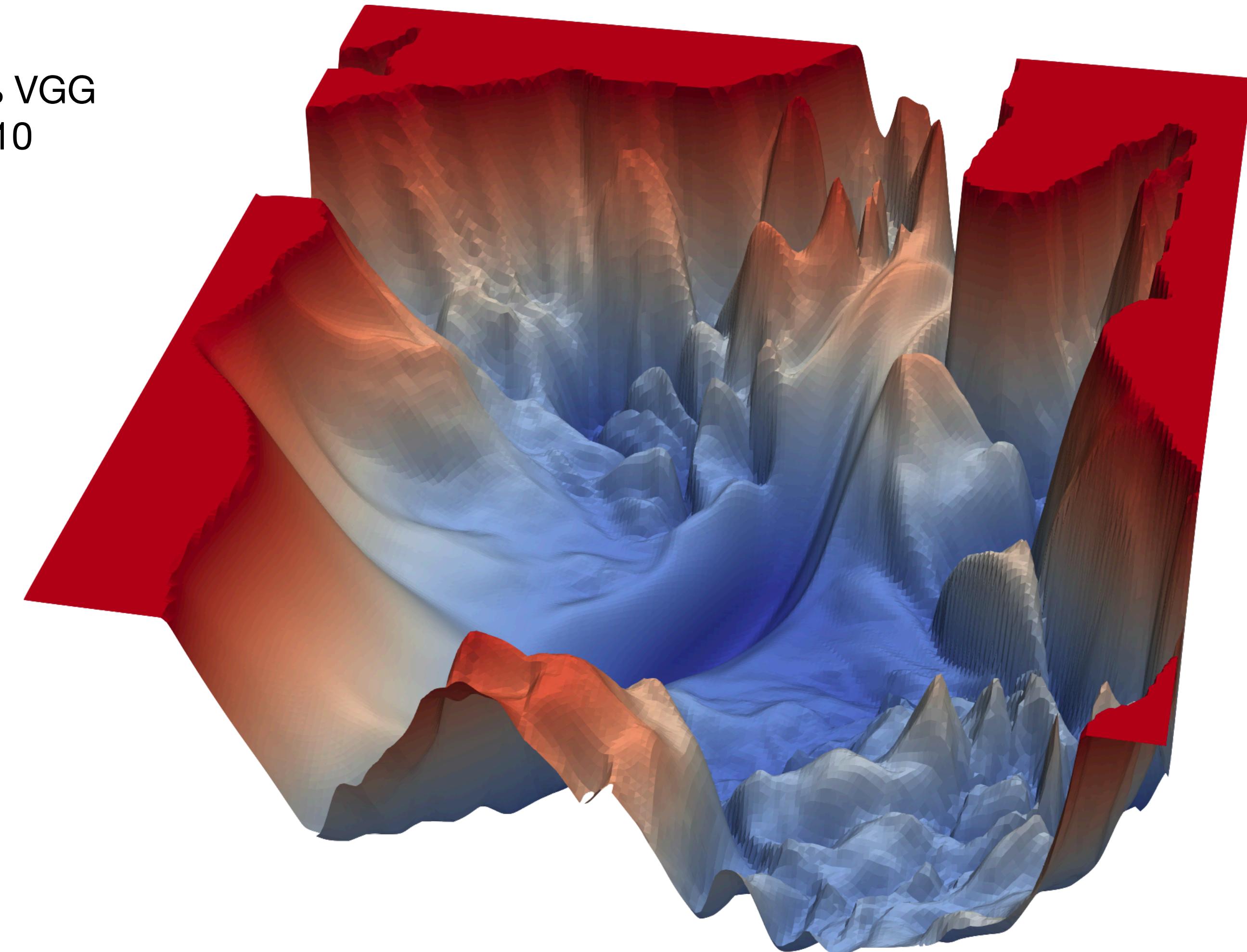
$$p(y = 1 | x) = \frac{1}{1 + \exp(-\langle w, x \rangle)}$$

MLE функция потерь:

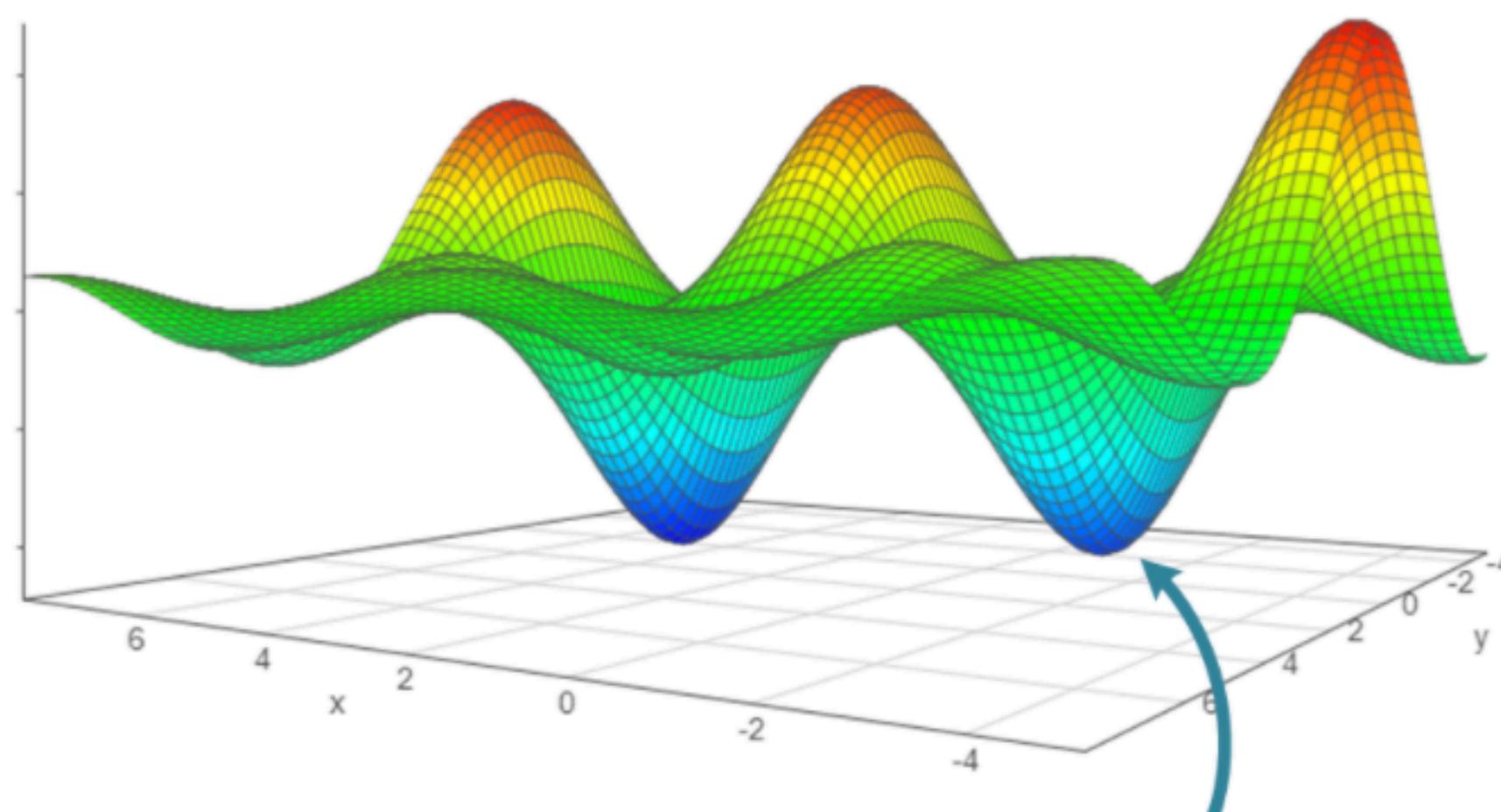


Оптимизационная поверхность: глубокая нейронная сеть

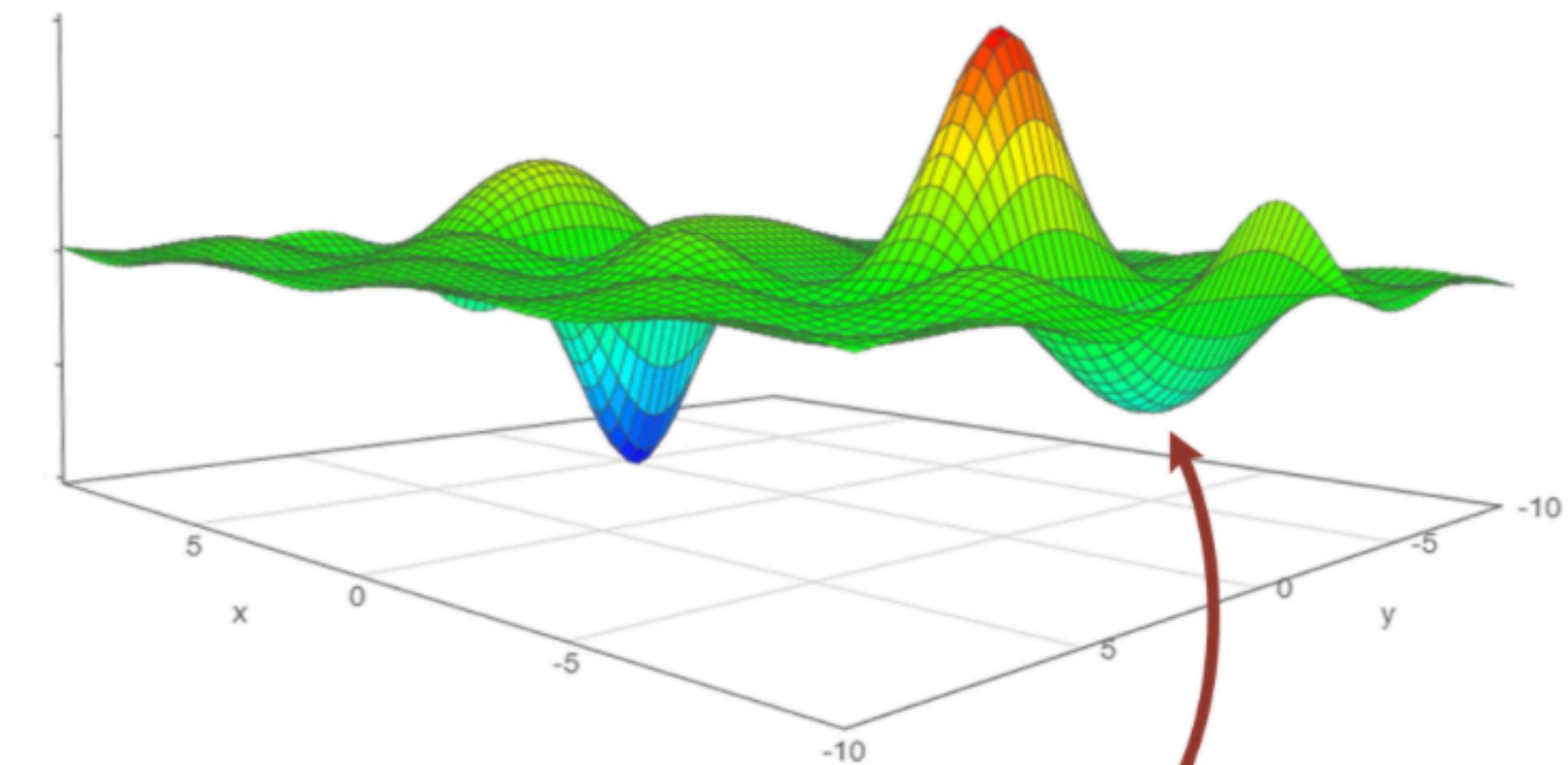
- Свёрточная сеть VGG
- Датасет: CIFAR-10



Локальные минимумы

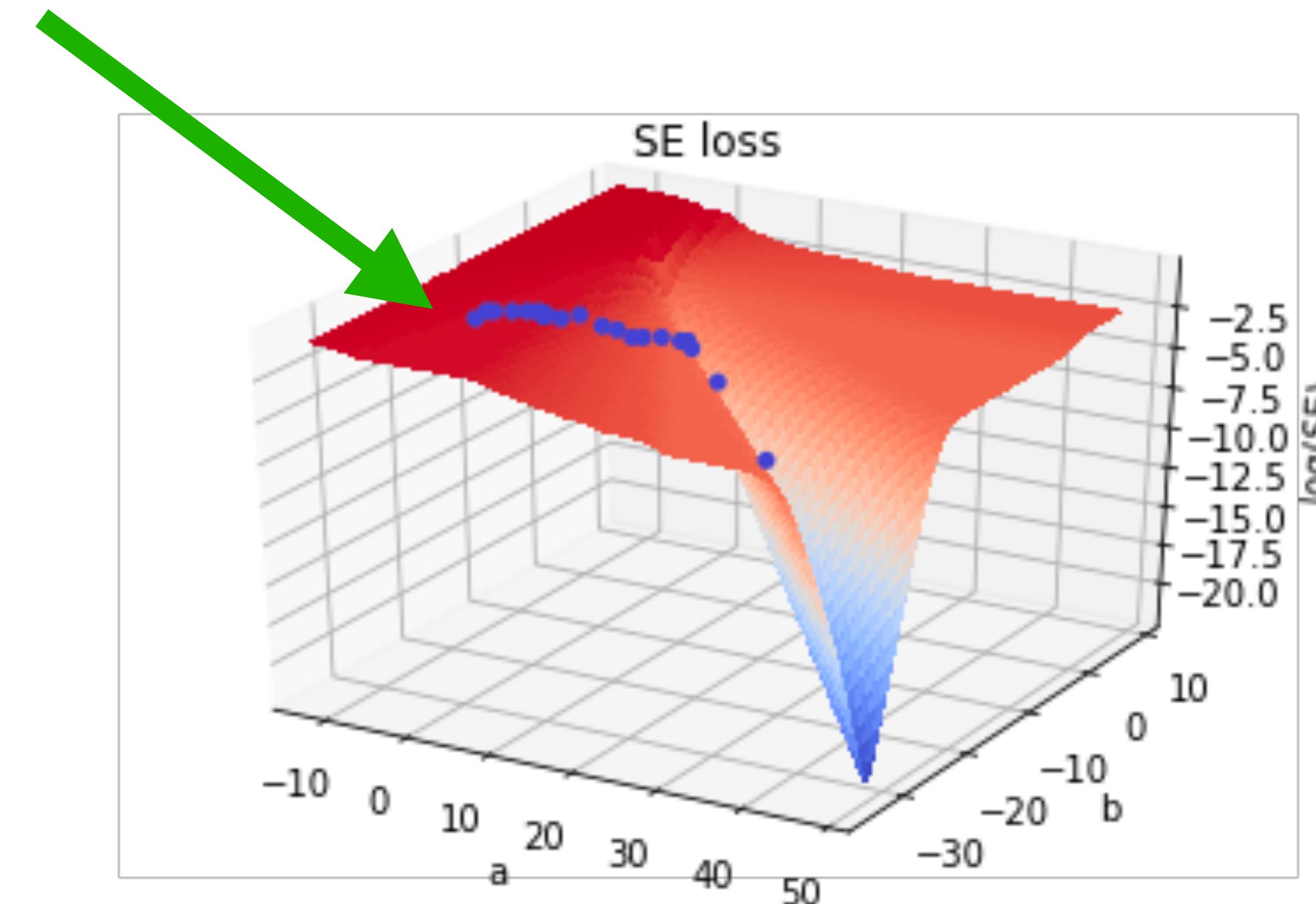


Good local minimum

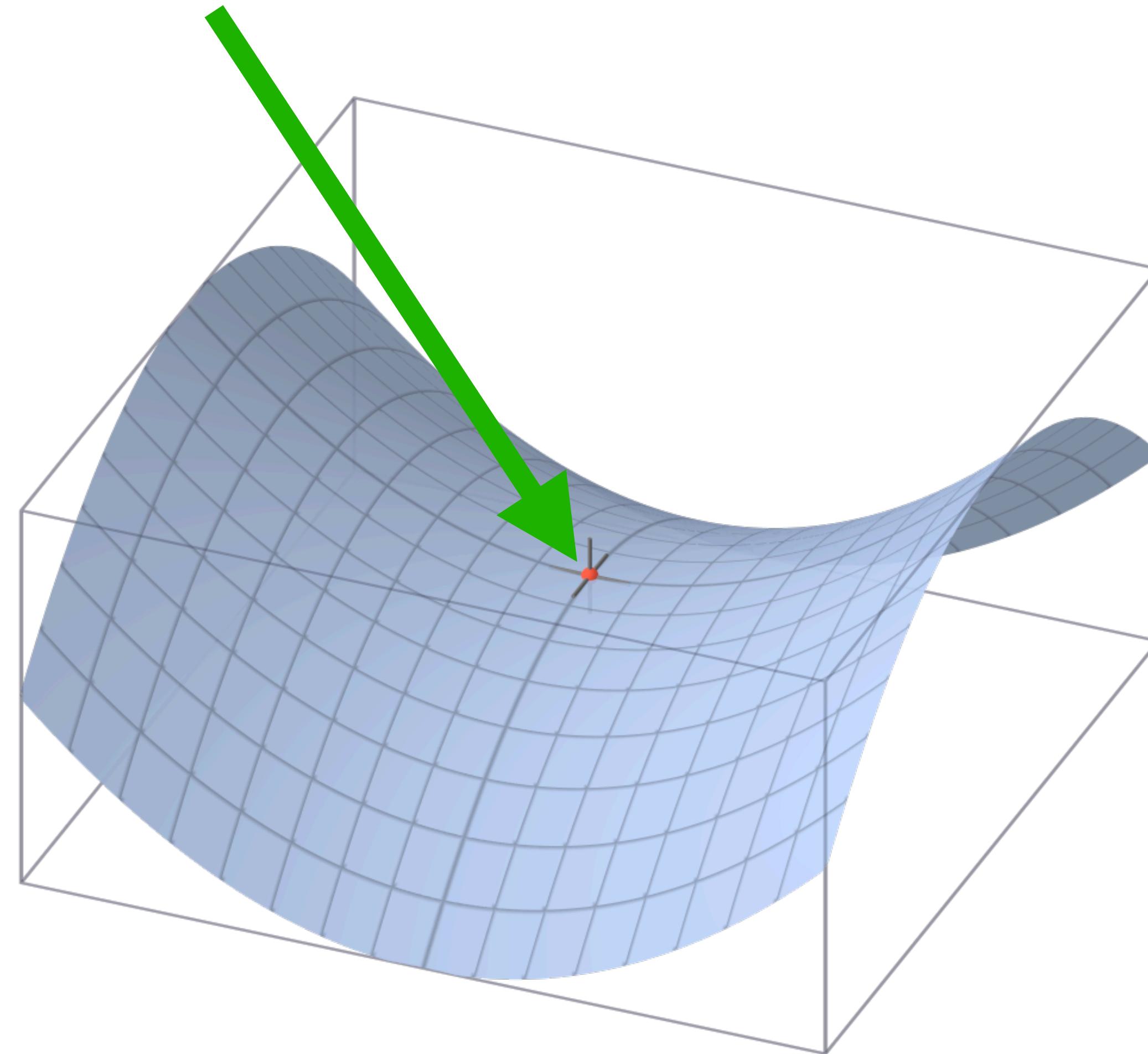


Poor local minimum

Плато



Седловая точка

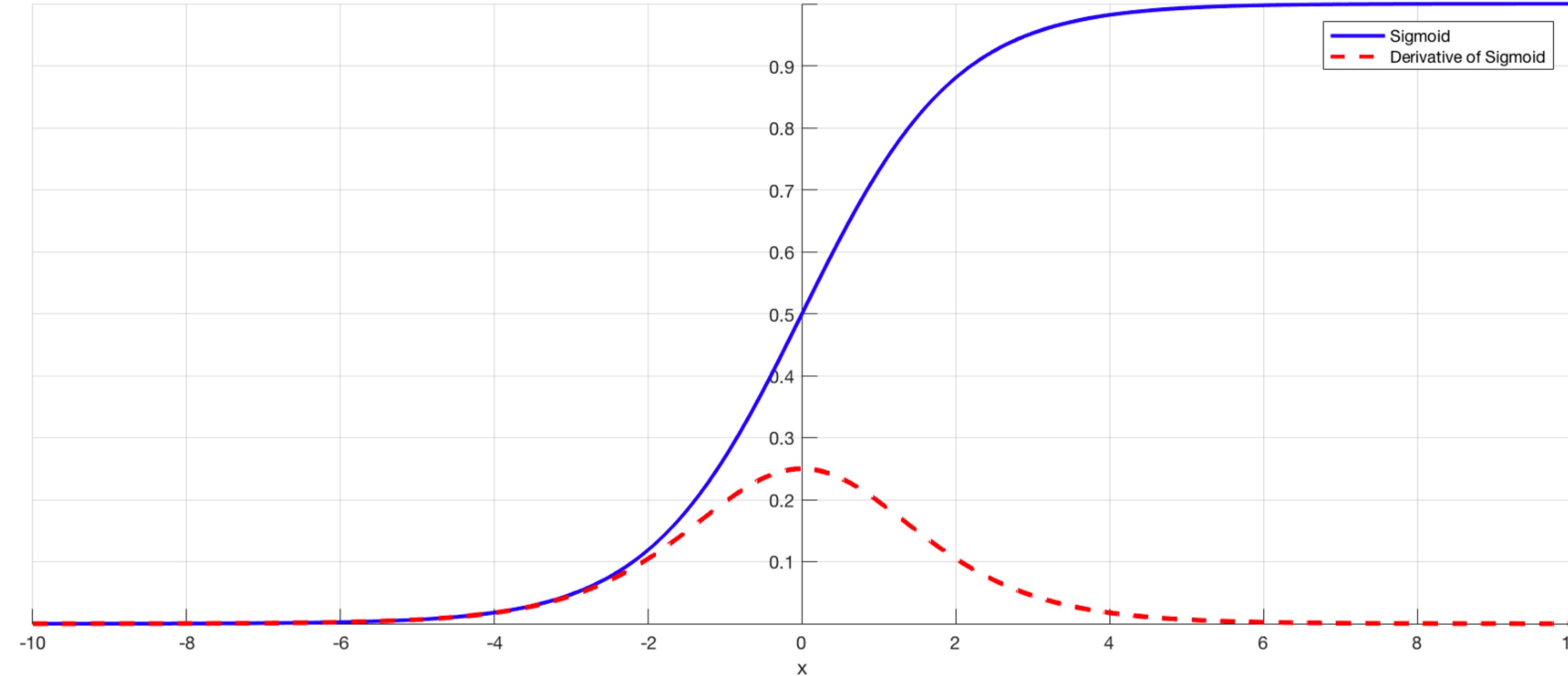


- Стационарная точка (производная равна 0)
- Не является экстремумом

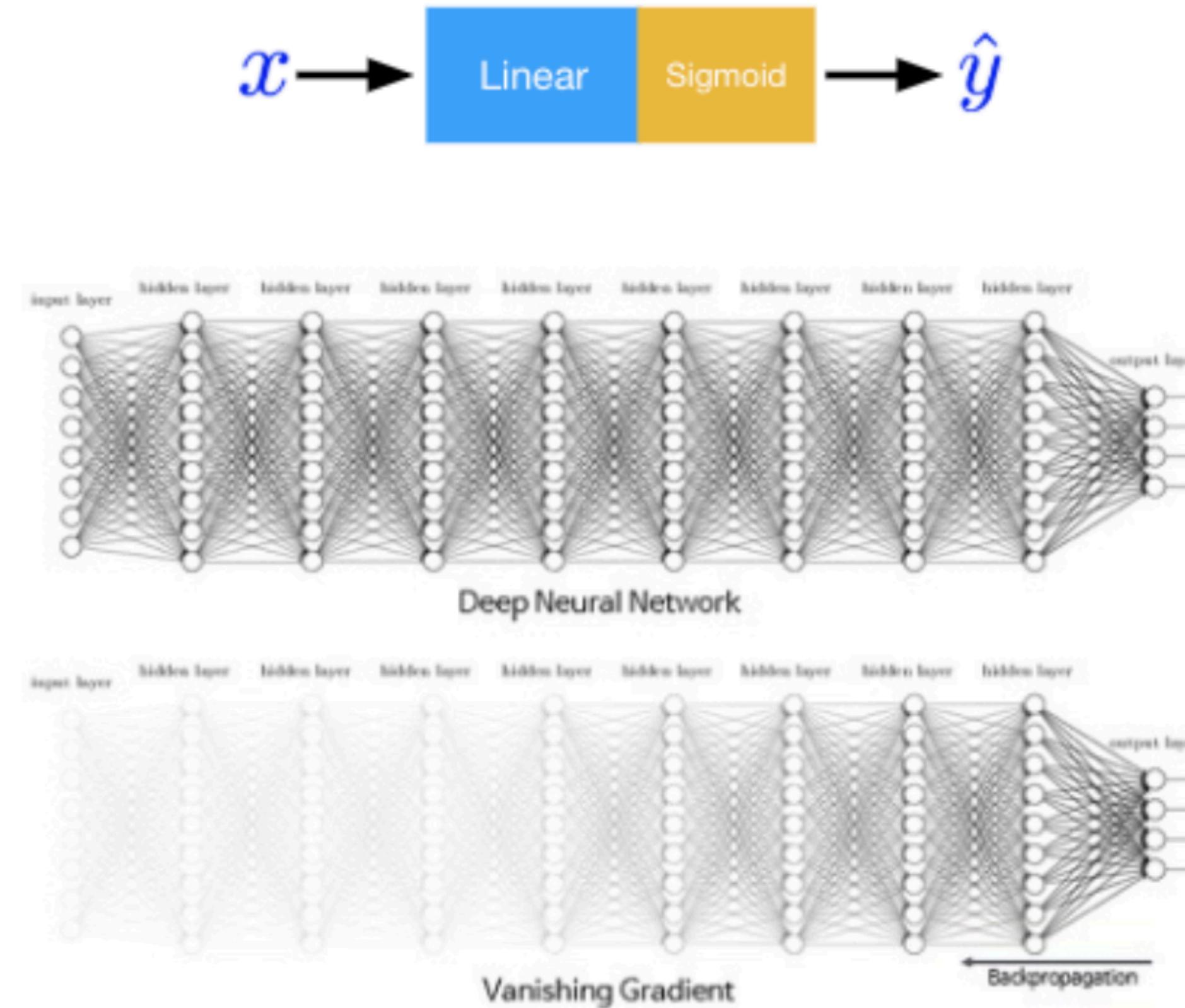
Размытие градиента (vanishing gradient)

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

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

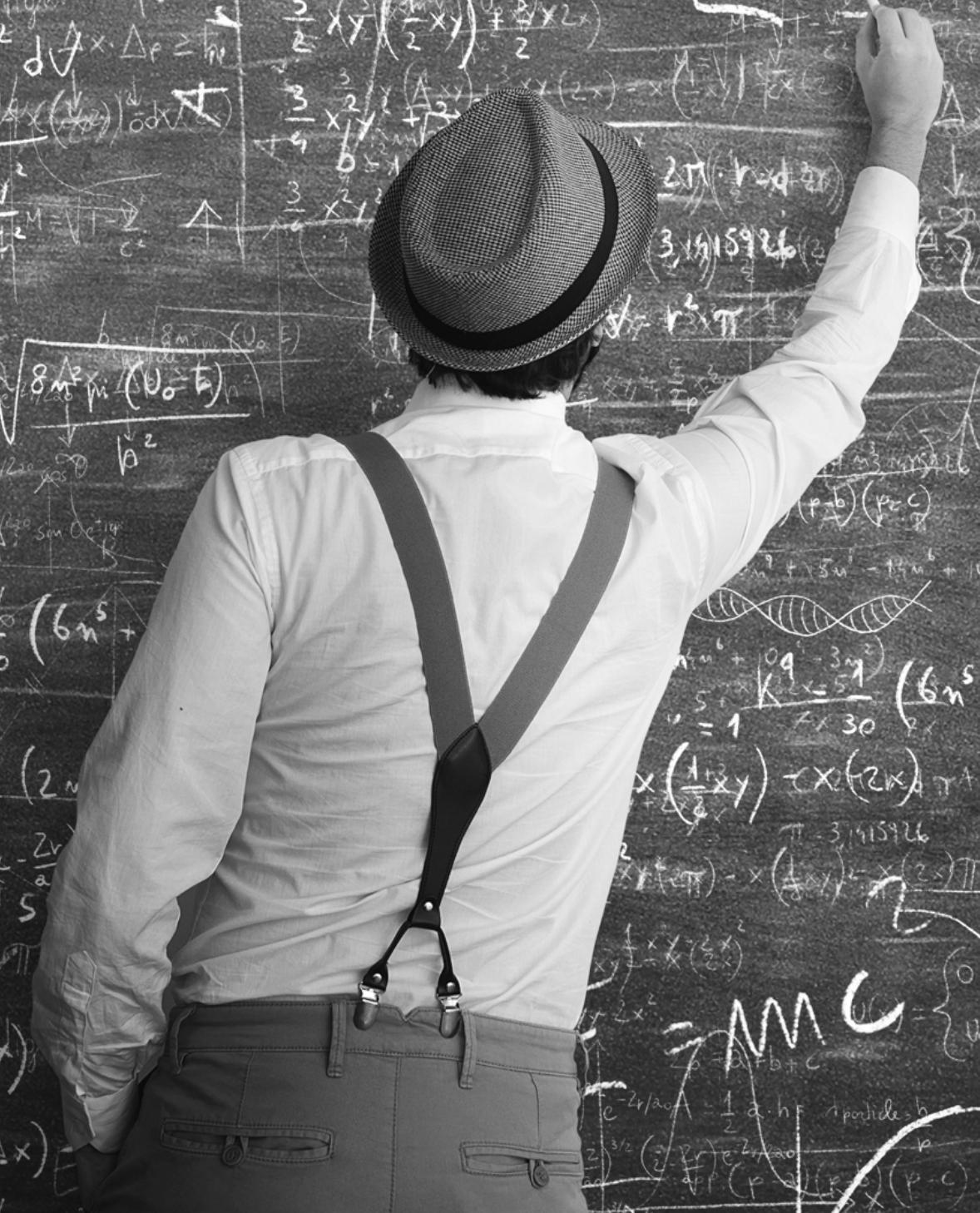


Размытие градиента (vanishing gradient)



- Варианты борьбы: другие функции активации (ReLU)
- Обратная ситуация: взрыв градиента (exploding gradient)

Методы оптимизации

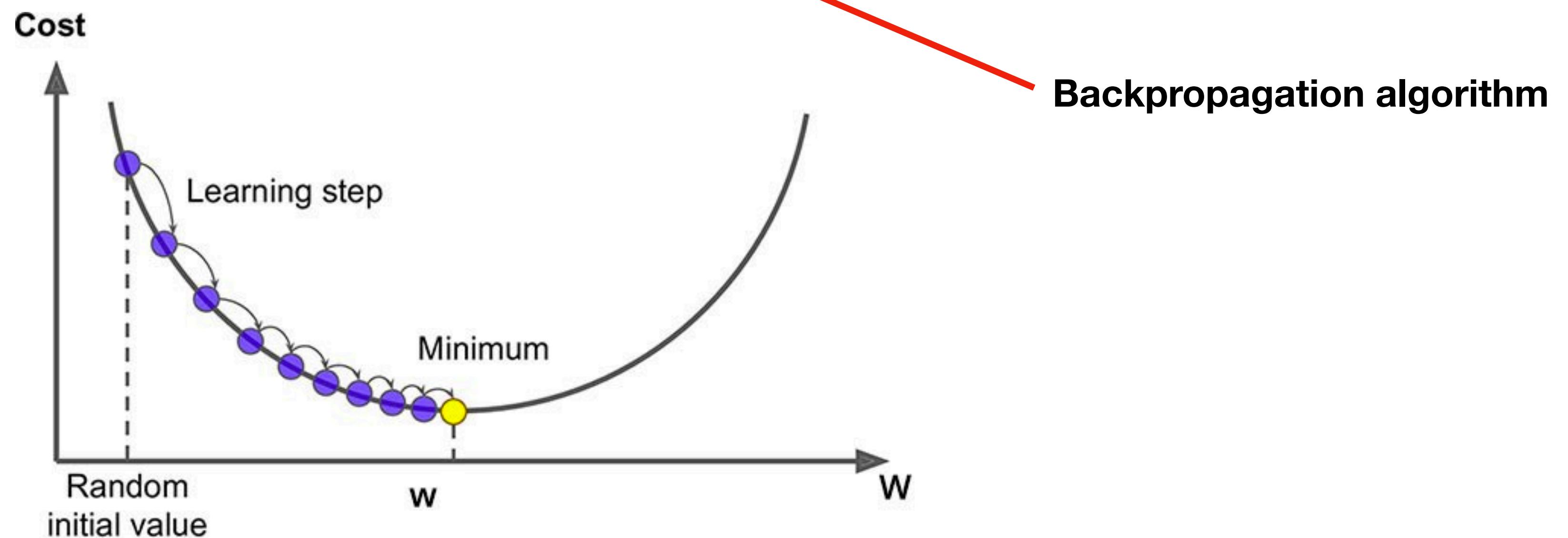


Gradient descent

ФУНКЦИЯ ПОТЕРЬ: $J(\theta) = \sum_{i=1}^N l(f_\theta(x_i), y_i)$, где N – все доступные тренировочные данные

η – шаг алгоритма (learning rate)

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} J(\theta)$$



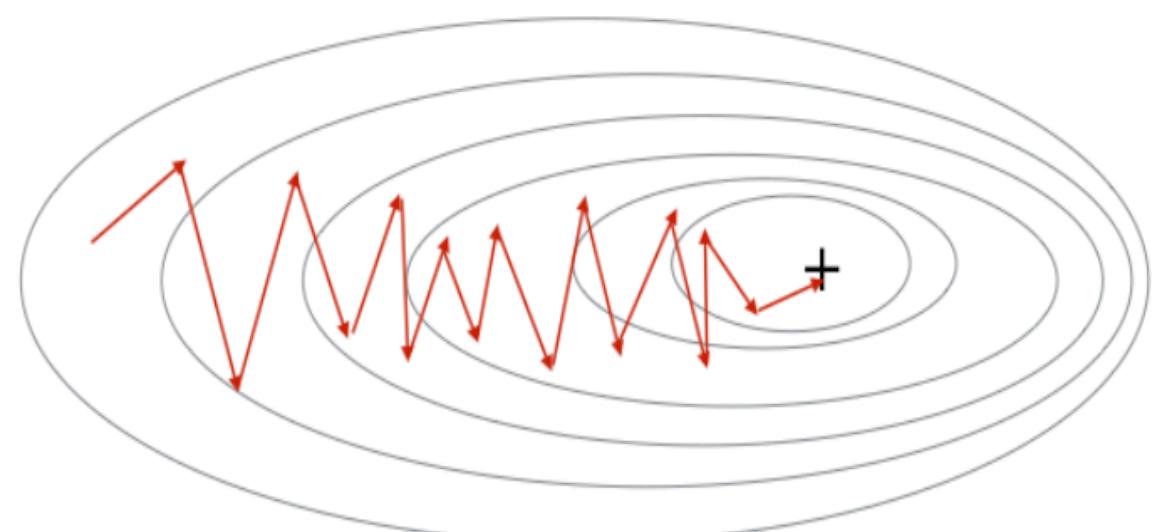
Stochastic gradient descent

Обновляем параметры для одного элемента данных (x_i, y_i) $J(\theta) = l(f_\theta(x_i), y_i)$

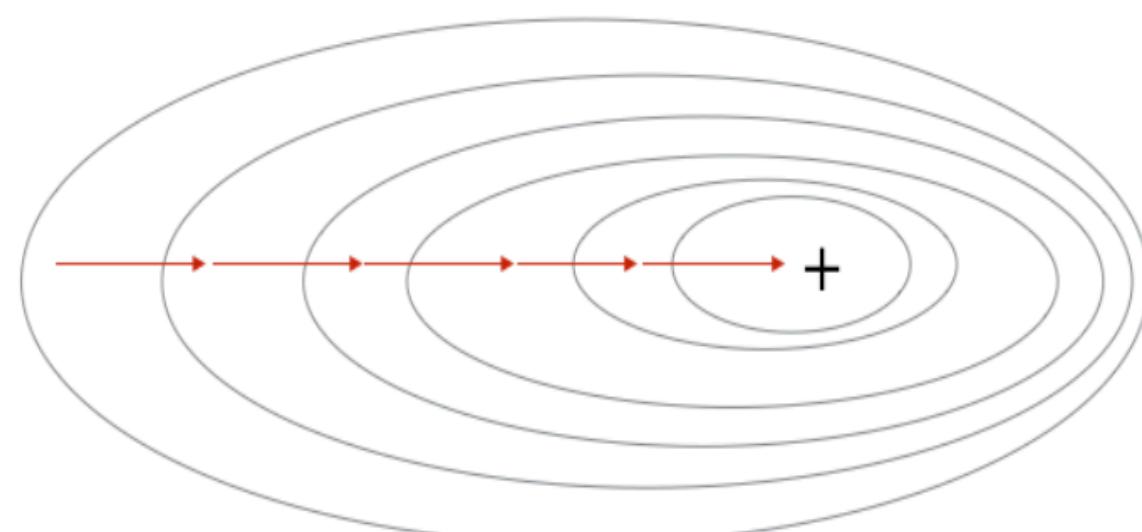
$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta)$$

$$\mathbb{E}_i(-\nabla_\theta l(f_\theta(x_i), y_i)) = \frac{1}{N} \sum_{i=1}^N \nabla_\theta l(f_\theta(x_i), y_i) = -\nabla_\theta J(\theta)$$

Stochastic Gradient Descent



Gradient Descent



- Проще реализуемый
- Помогает с проблемой локальных минимумов

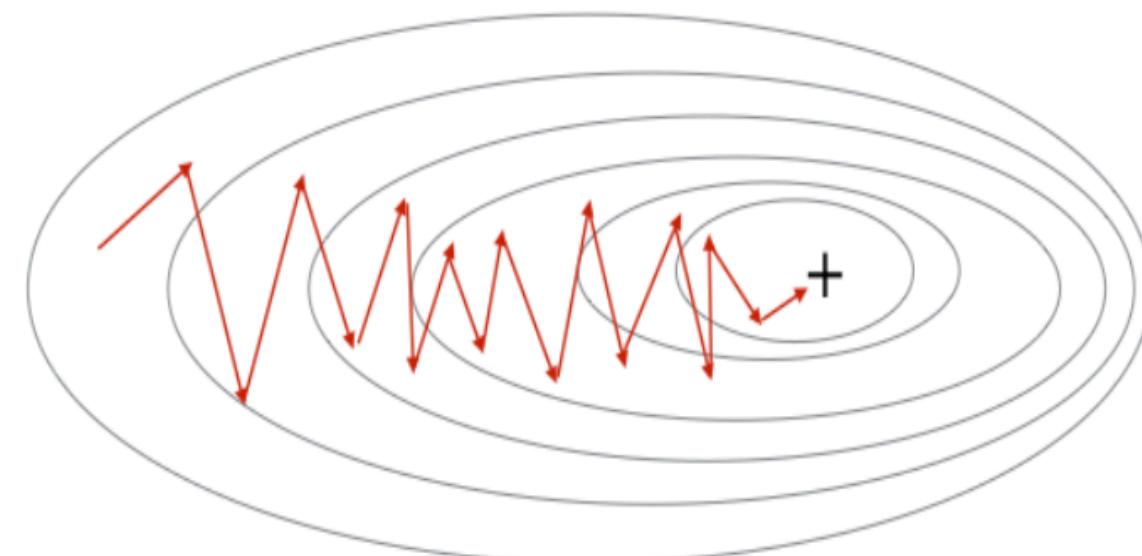
Mini-batch gradient descent

Обновляем параметры не для одного, а для M элементов данных

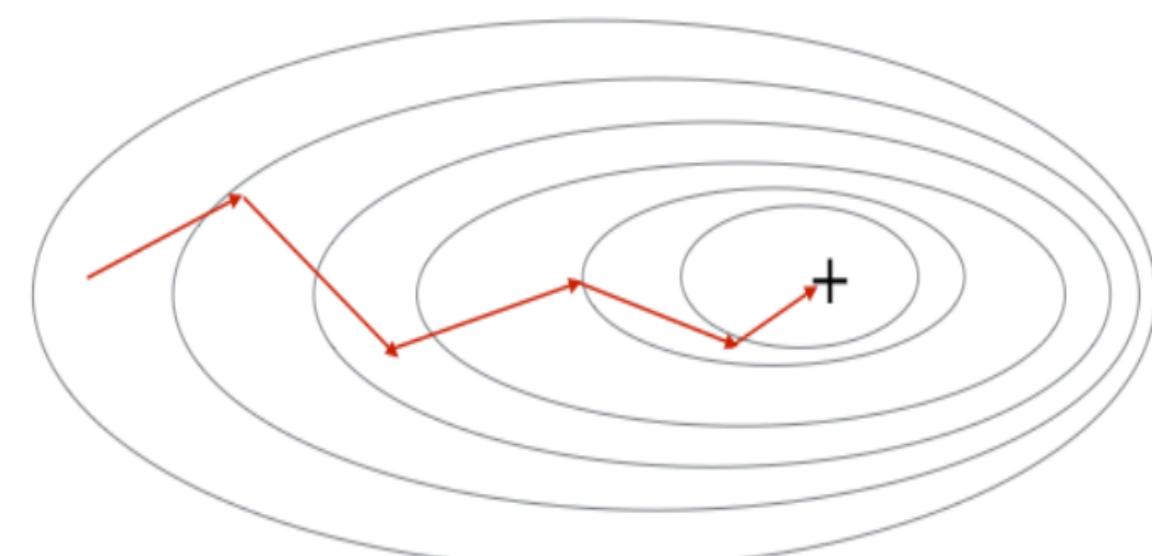
$$J(\theta) = \sum_{i=1}^M l(f_\theta(x_i), y_i)$$

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} J(\theta)$$

Stochastic Gradient Descent



Mini-Batch Gradient Descent

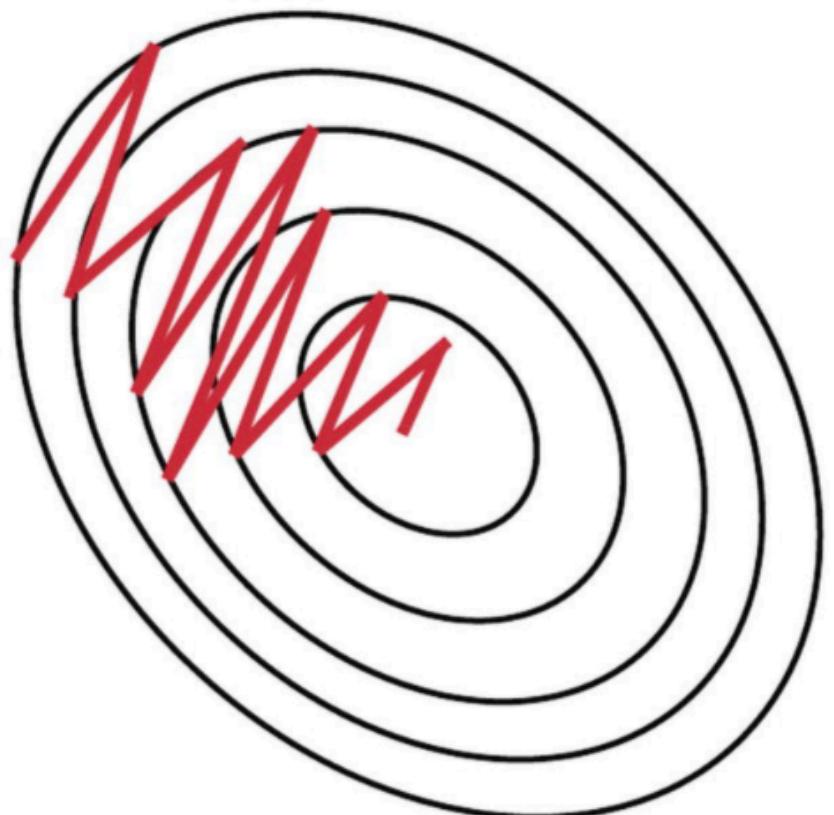


- Вычислительно эффективнее
- Размер батча M обычно берут 64, 256, 512, 1024
- В дальнейшем рассматриваем именно такой вариант функции потерь
- Часто под SGD имеют ввиду этот метод

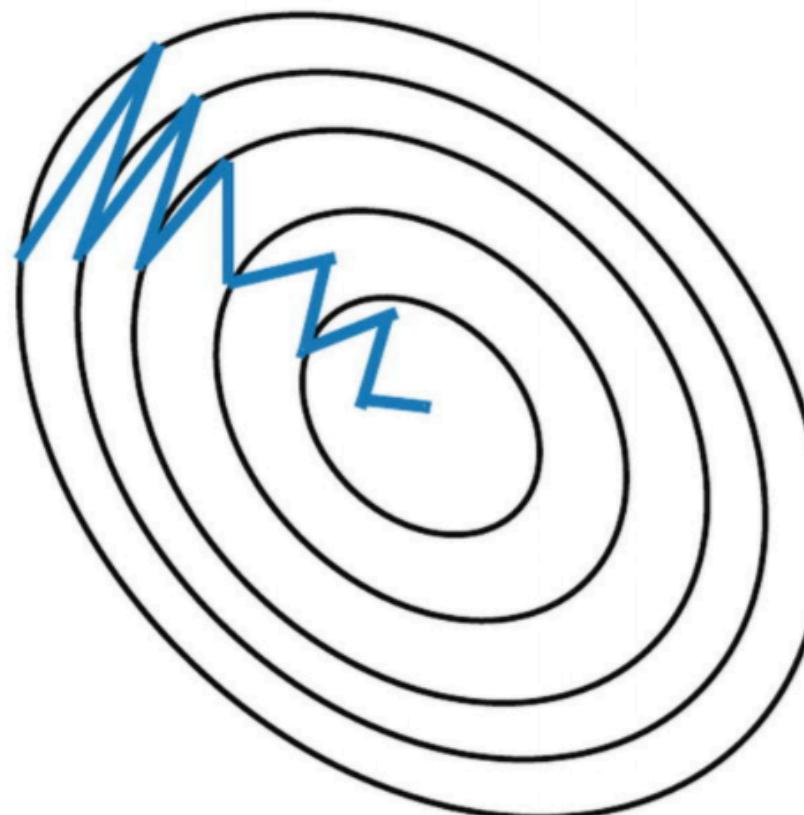
Momentum (Polyak, 1964)

$$v_t = \gamma v_{t-1} + \eta \nabla_{\theta} J(\theta)$$

$$\theta_{t+1} = \theta_t - v_t$$



Stochastic Gradient
Descent **without**
Momentum



Stochastic Gradient
Descent **with**
Momentum

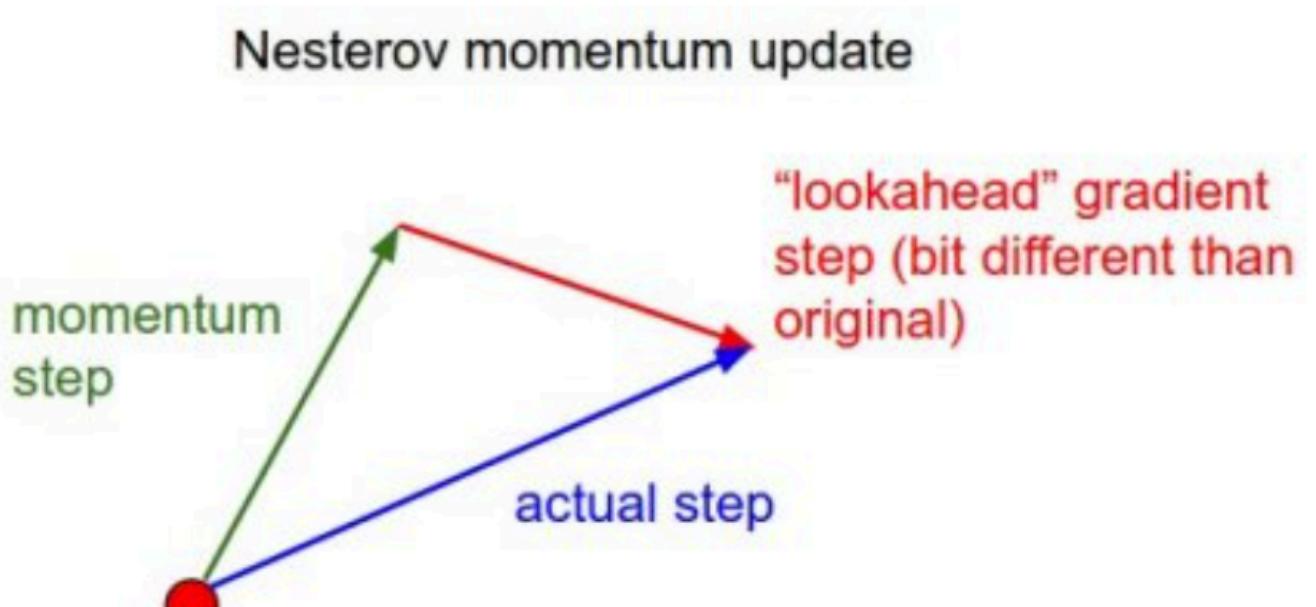
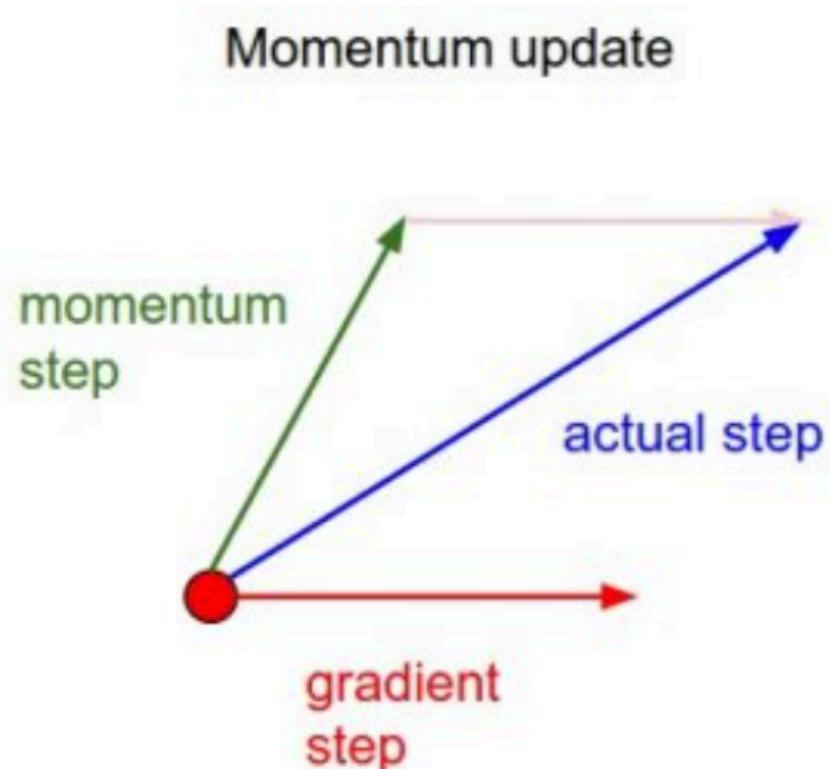
SGD with and without momentum

- Рядом с оптимумом часто возникают области, где SGD осциллирует
- Часто используют $\gamma = 0.9$

Nesterov Momentum (Nesterov, 1983, Sutskever et al., 2013)

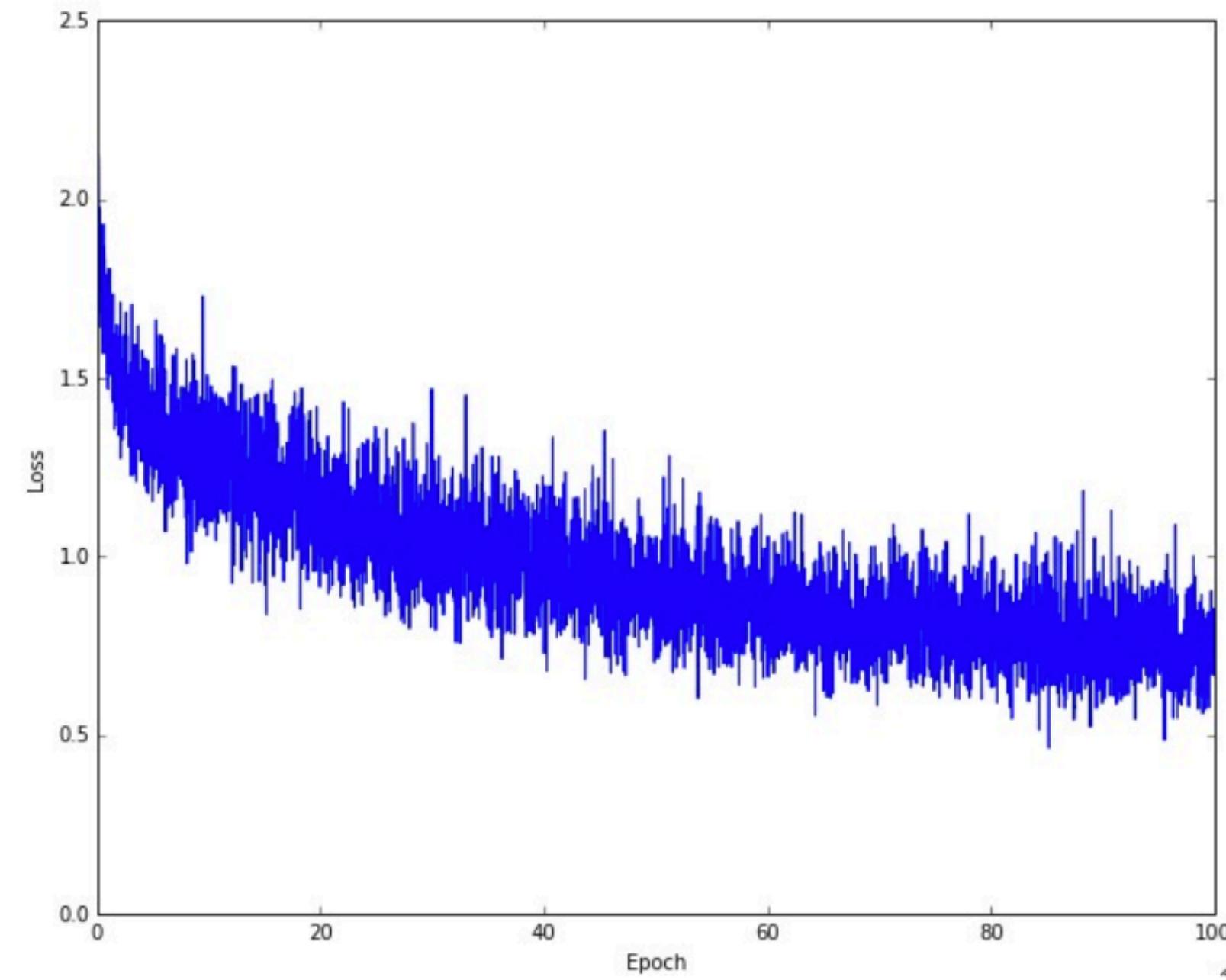
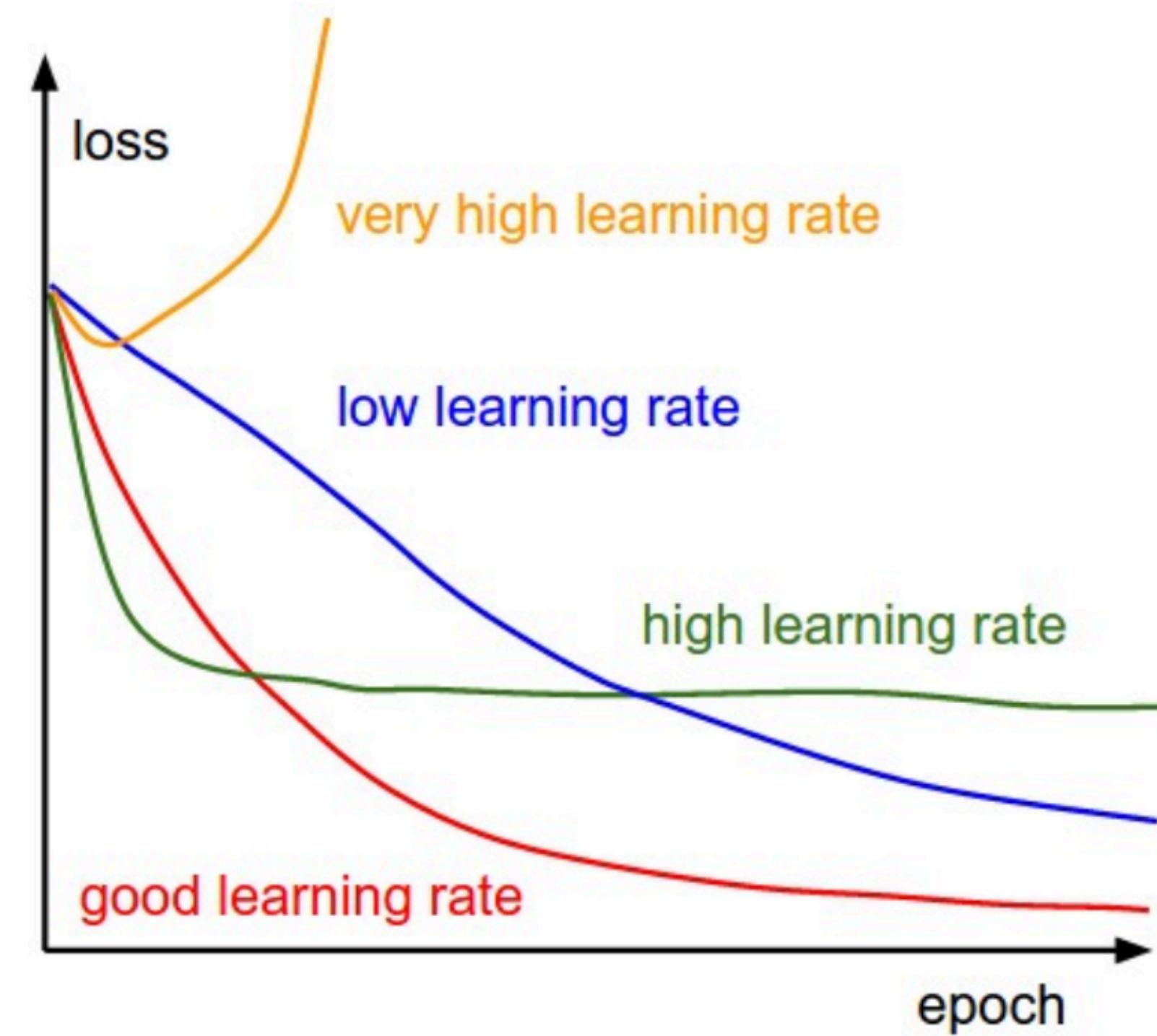
$$v_t = \gamma v_{t-1} + \eta \nabla_{\theta} J(\theta - \gamma v_{t-1})$$

$$\theta_{t+1} = \theta_t - v_t$$

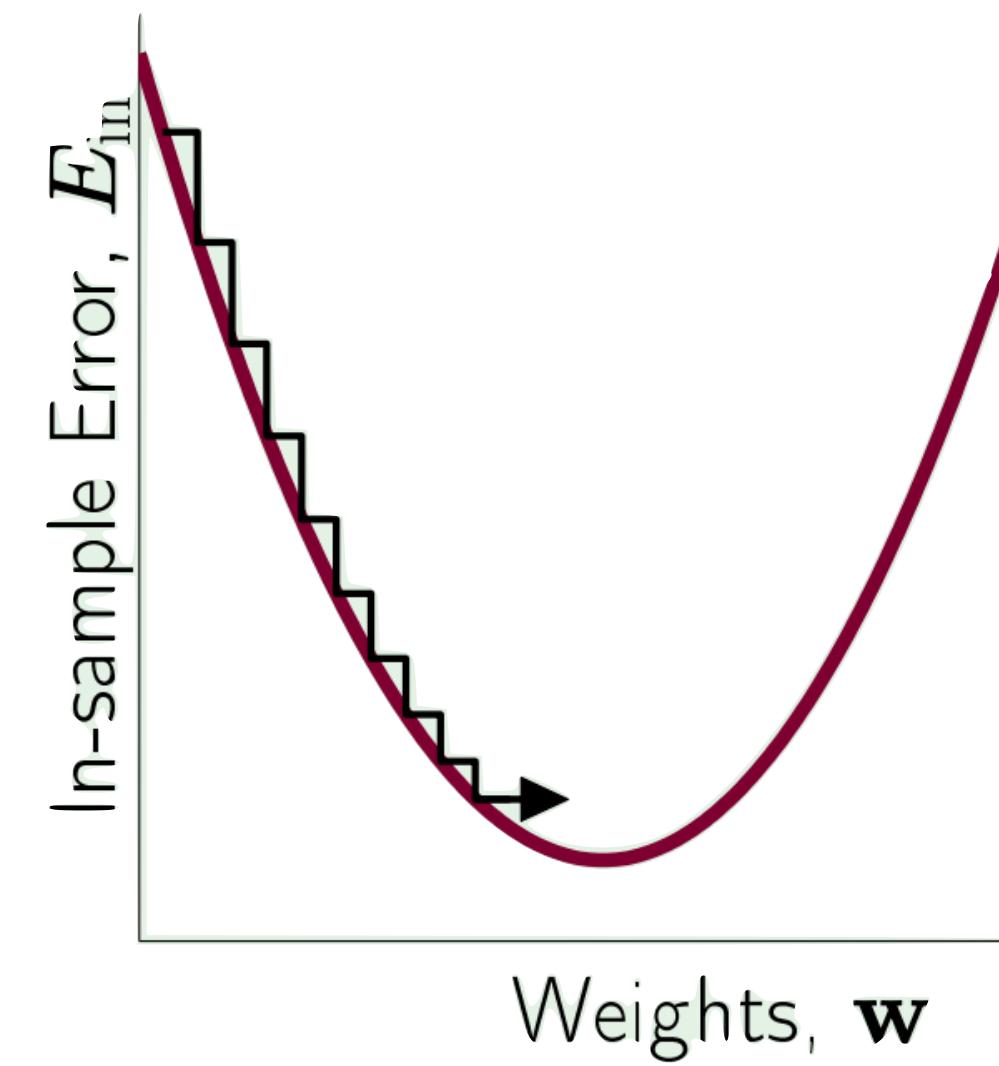


- Будем вычислять градиент для будущей позиции параметра
- Метод более устойчив, чем обычный Momentum

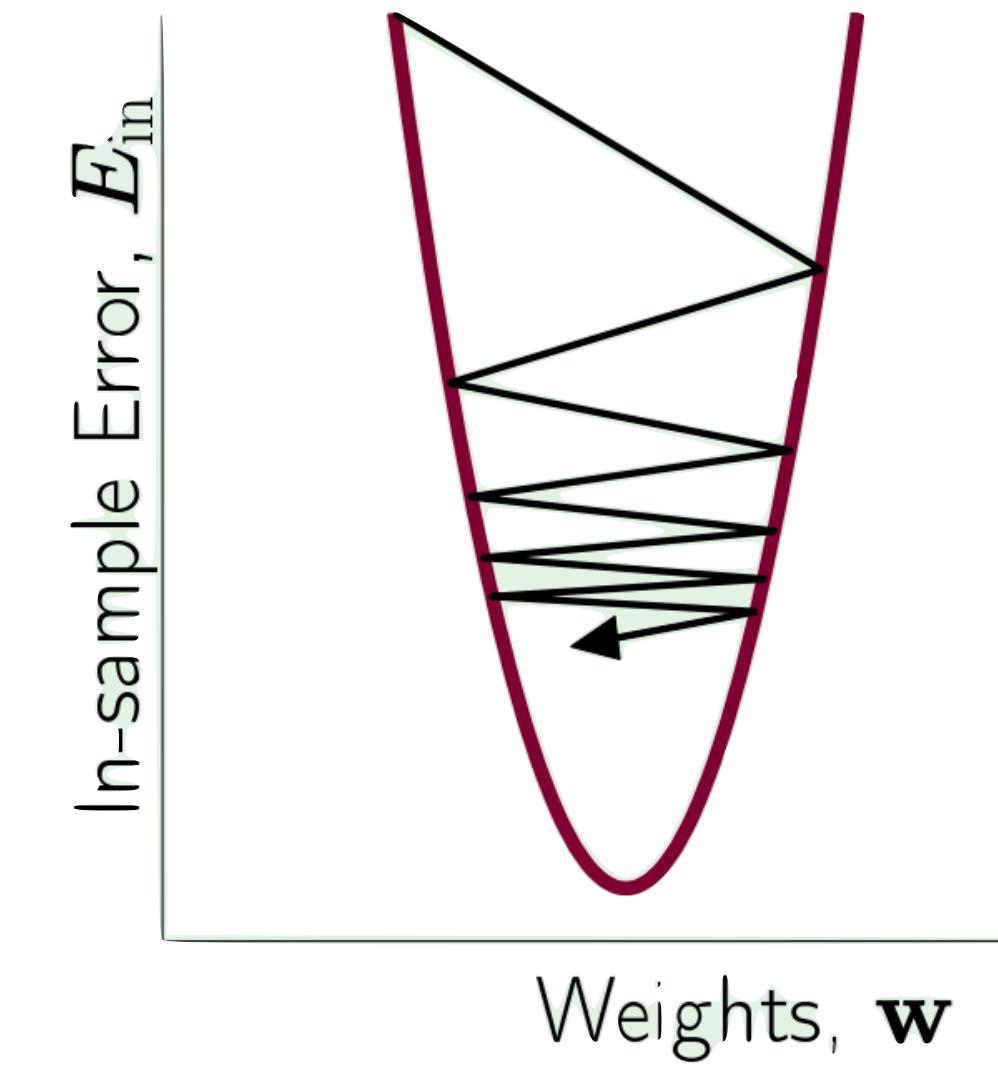
Значение learning rate



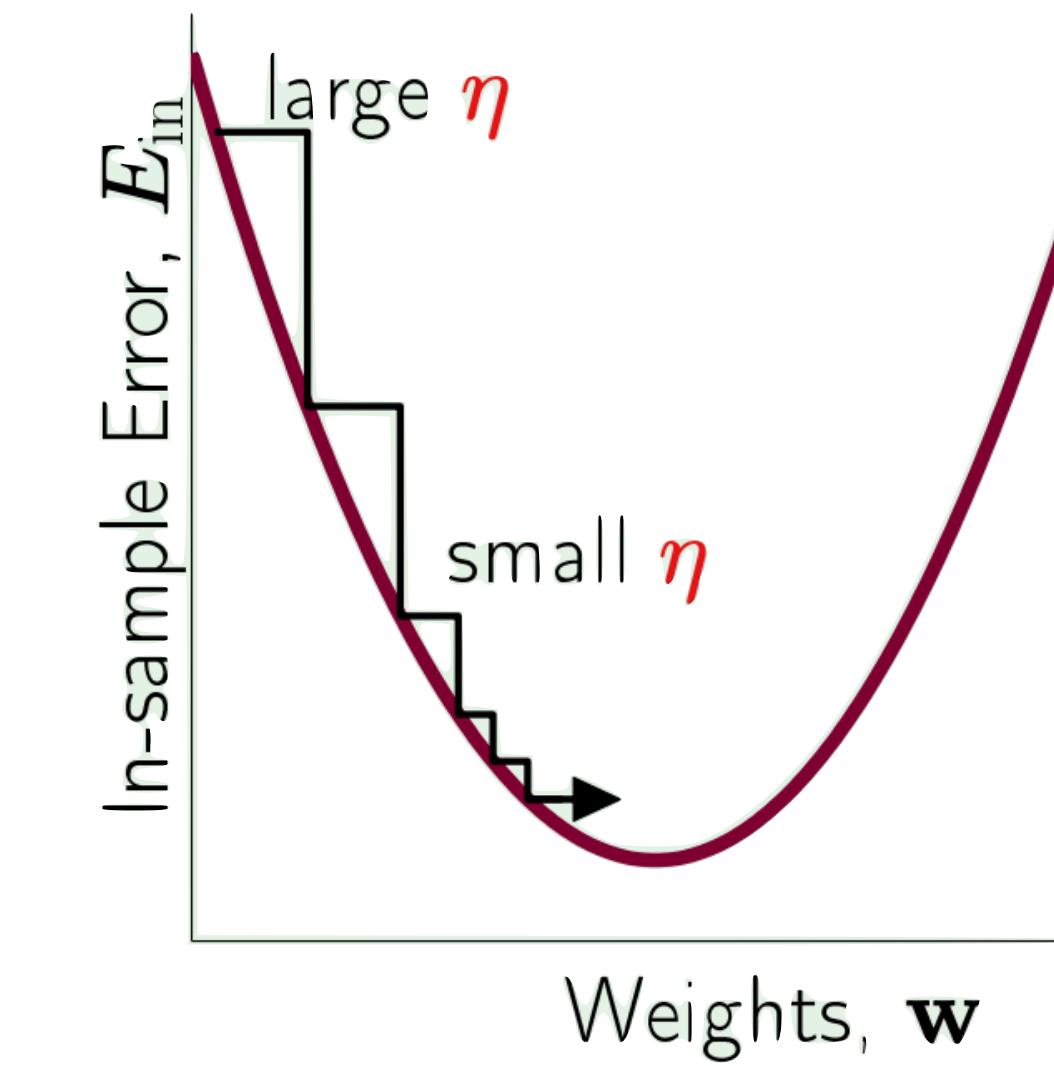
Адаптация learning rate



η too small



η too large

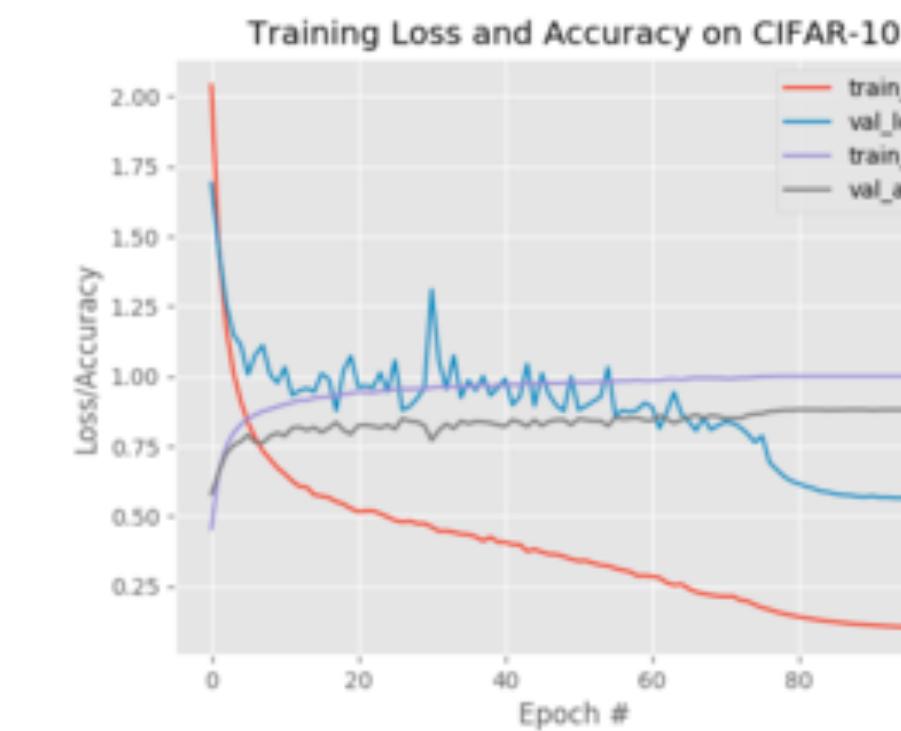
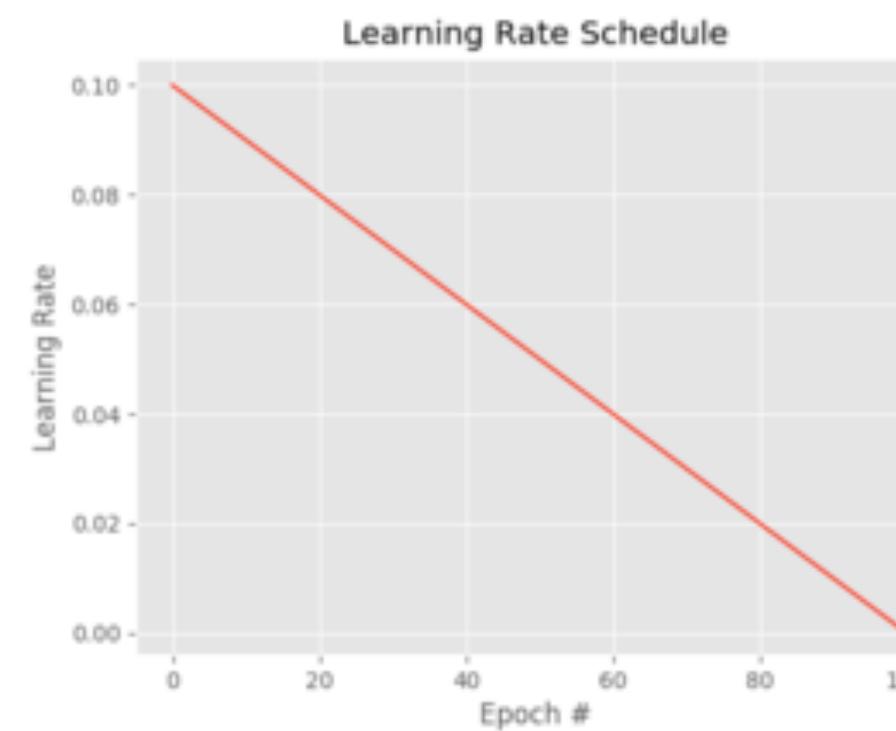


variable η – just right

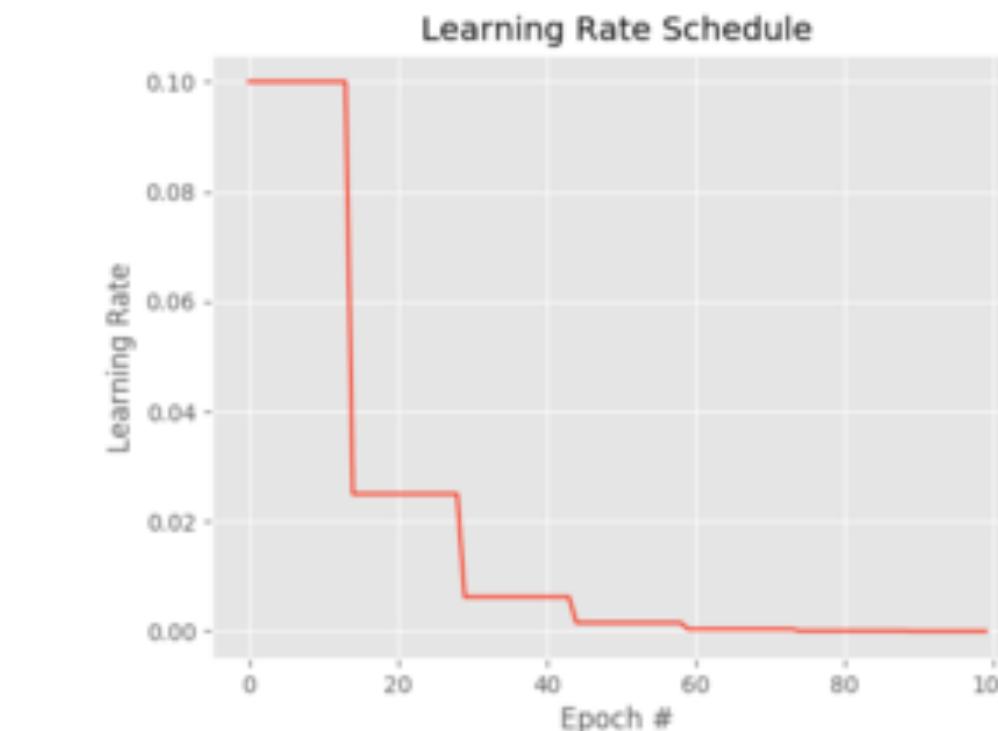
Основная идея: если значение градиента **большое**, то шаг должен быть **уменьшиться**,
если значение градиента **маленькое**, то шаг должен **увеличиться**

Изменение learning rate (scheduler)

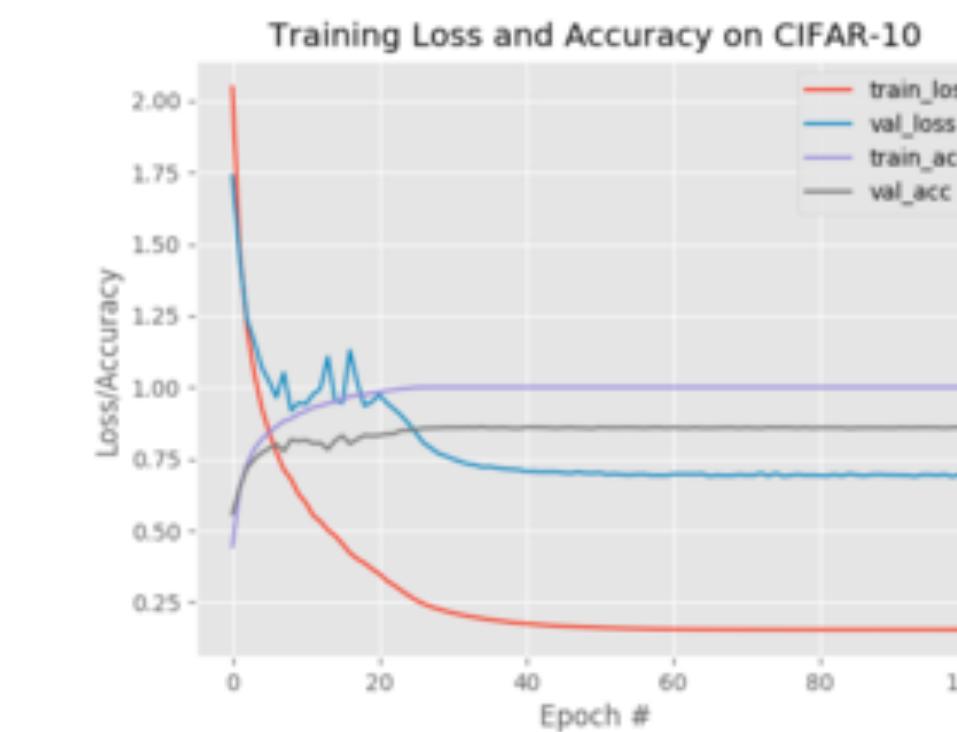
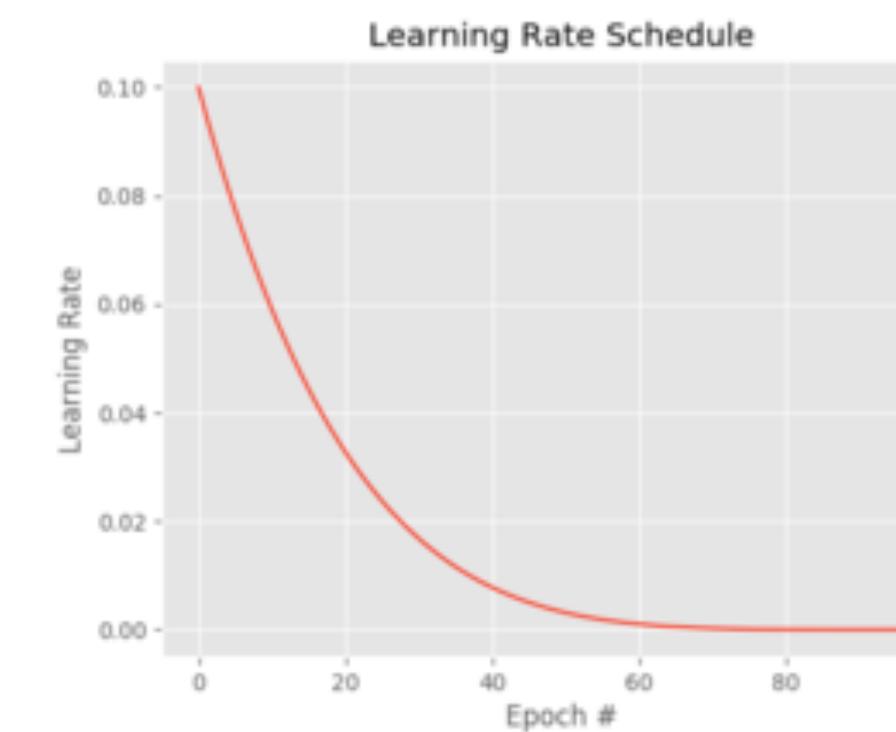
Линейный (Linear)

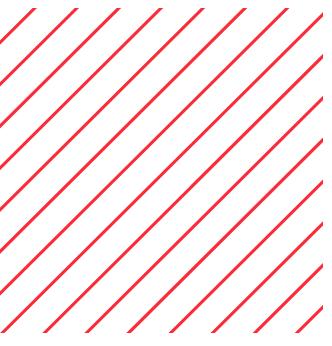


Пошаговый (Step-based)



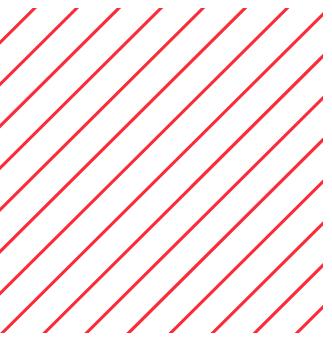
Экспоненциальный (Exponential)





Адаптивный подбор learning rate

- Следующие методы адаптивно подбирают его, обратно пропорционально квадратному корню величины градиента:
 - Adagrad (Duchi et al., 2011)
 - Adadelta (Zeiler, 2012)
 - RMSprop (Hinton, 2016)
 - AdaBelief (Zhuang, 2020)



Adaptive Moment Estimation (Adam) (Kingma et al., 2015)

$$g_t = \nabla_{\theta} J(\theta)$$

$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$ – первый момент градиента

$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ – второй момент градиента

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

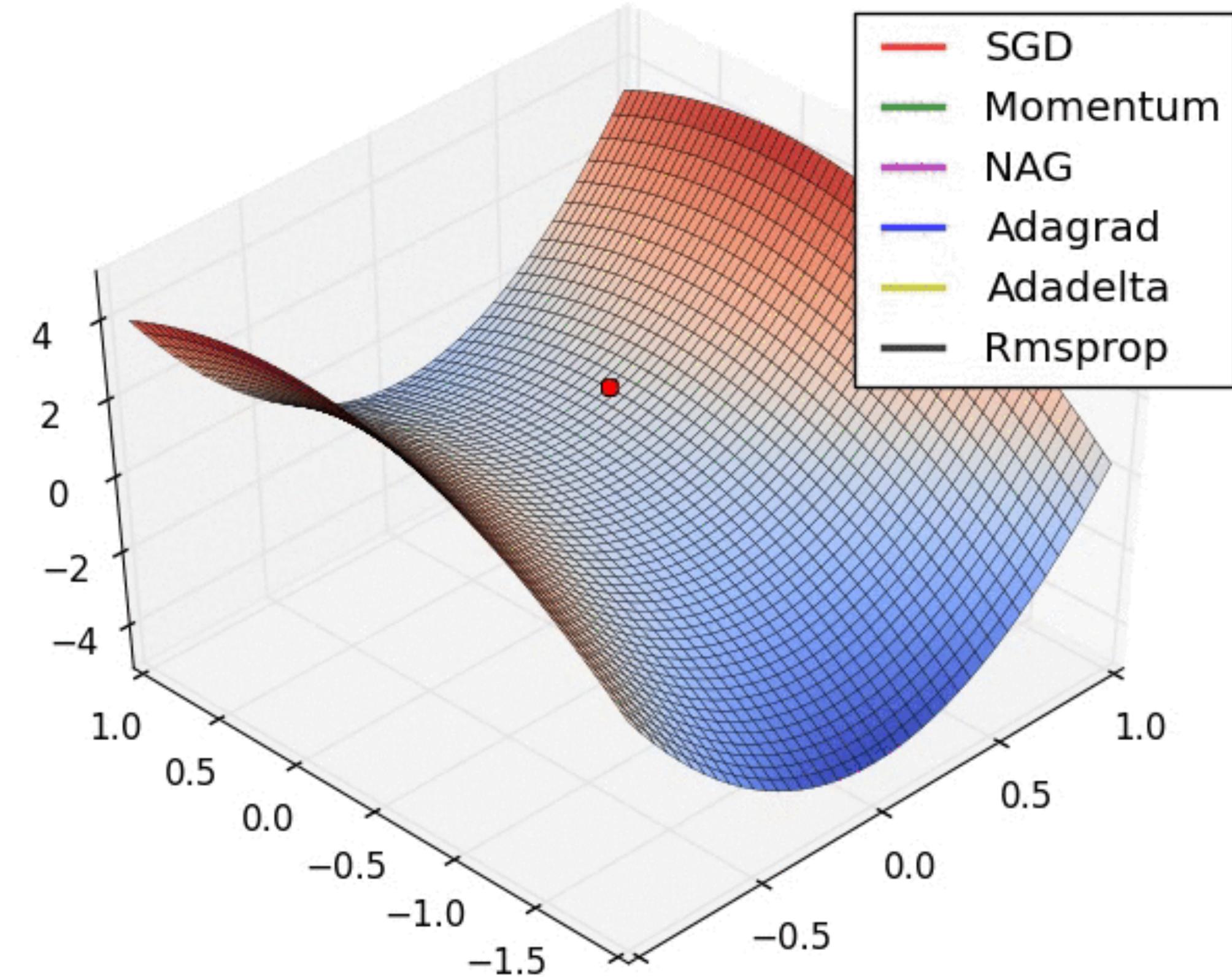
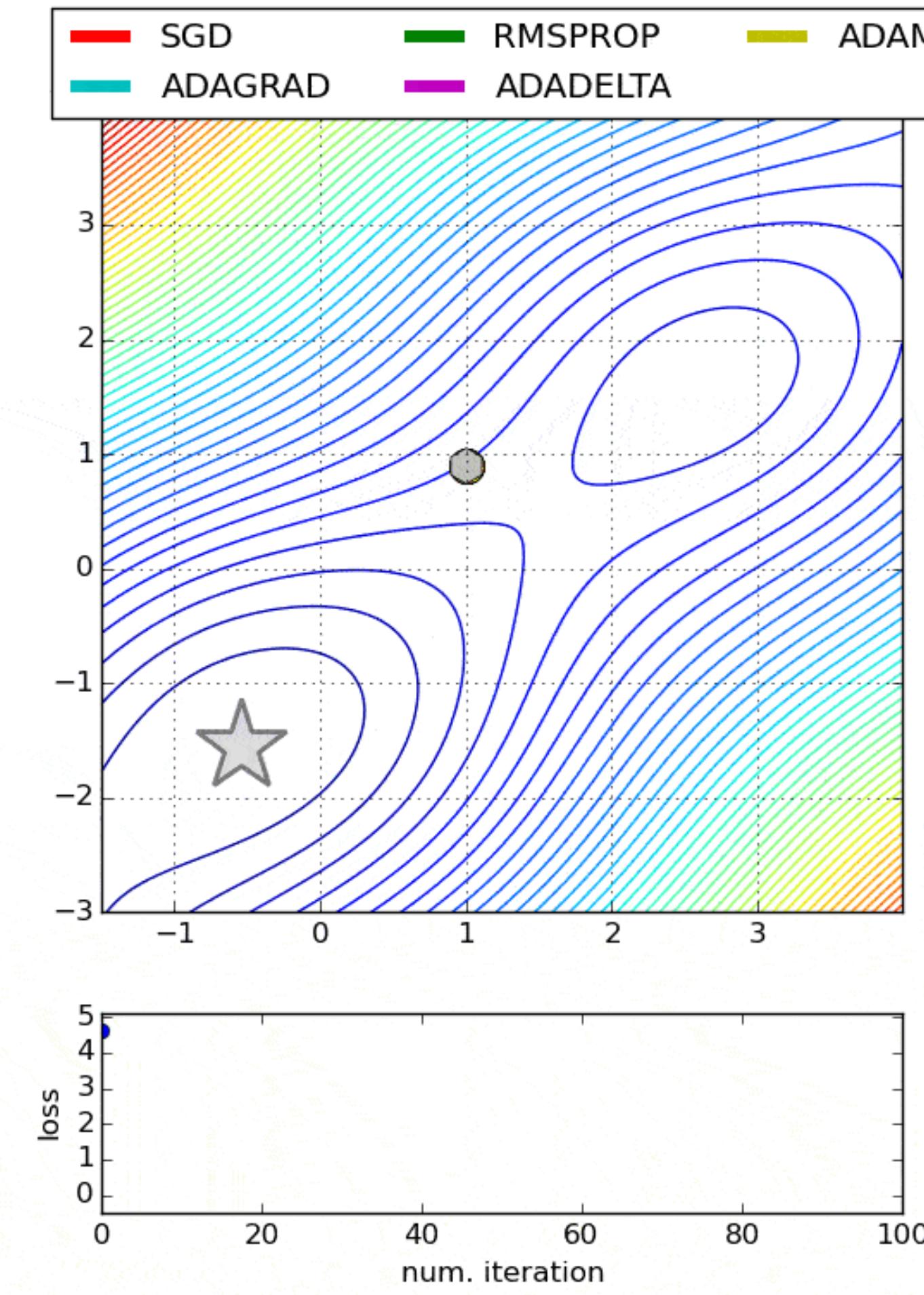
– поправки моментов на смещение

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

- $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$

- Наиболее успешный (в среднем) на практике метод

Сравнение методов



Сравнение методов

CIFAR-10 - CNN

A slightly larger convolutional network for the Cifar-10 data set, with three conv and three fully-connected layers.

	Optimizer	Test Accuracy	Speed
#1	Adam	84.75 %	36.0
#2	Momentum	84.41 %	40.7
#3	SGD	83.71 %	42.5

SVHN - Wide ResNet

16-4

The **Wide ResNet 16-4** for the Street View House Numbers data set using the variant with 16 conv layers and a widening factor of 4.

	Optimizer	Test Accuracy	Speed
#1	Momentum	95.53 %	10.8
#2	SGD	95.37 %	28.3
#3	Adam	95.25 %	12.1

F-MNIST - VAE

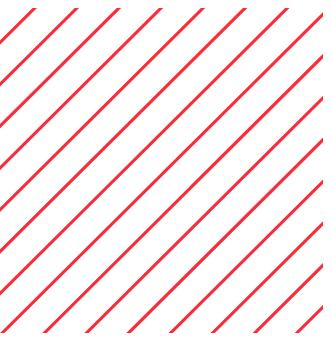
A basic variational autoencoder for the Fashion-MNIST data set with three convolutional and three deconvolutional layers.

	Optimizer	Test Loss	Speed
#1	Adam	23.07	1.0
#2	SGD	23.80	1.0
#3	Momentum	59.23	1.0

Tolstoi - Char RNN

A recurrent neural network for character-level language modeling on the novel *War and Peace* by Leo Tolstoy using two LSTM layers.

	Optimizer	Test Accuracy	Speed
#1	SGD	62.07 %	47.7
#2	Momentum	61.30 %	88.0
#3	Adam	61.23 %	62.8



Без использования градиента (прямые методы)

- **Стохастическая аппроксимация:** известны только значения $f(\theta_t)$
- Алгоритм Роббинса-Монро (1951)
 - $\theta_{t+1} = \theta_t - \beta_t(f(\theta_t) - \alpha)$
 - Сходится при ряде ограничений на $f(\theta_t)$
- **Simultaneous perturbation stochastic approximation** (SPSA) (Spall, 1992)
 - $$\theta_{t+1} = \theta_t - \alpha_t \Delta_t \frac{f(\theta_t + \beta_t \Delta_t) - f(\theta_t - \beta_t \Delta_t)}{2\beta_t}$$
 - Δ_t – случайное возмущение, α_t , β_t – параметры
 - Метод сходится при почти любом шуме в измерениях $f(\theta_t)$
- Методы стохастической аппроксимации применяются, когда нет возможности качественно измерить градиент, например, в обучении с подкреплением (reinforcement learning)

Резюме



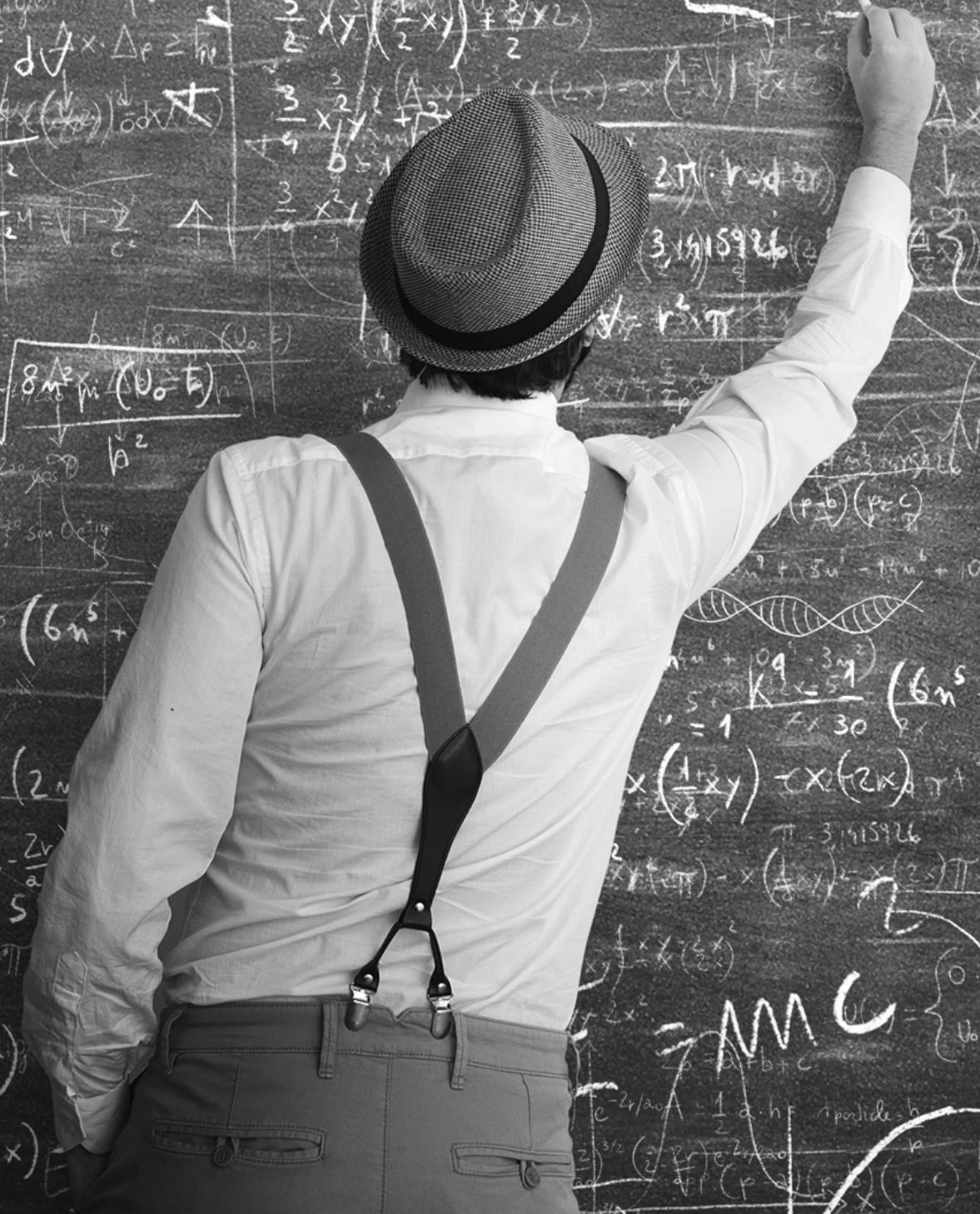
Andrej Karpathy 
@karpathy

...
3e-4 is the best learning rate for Adam, hands down.

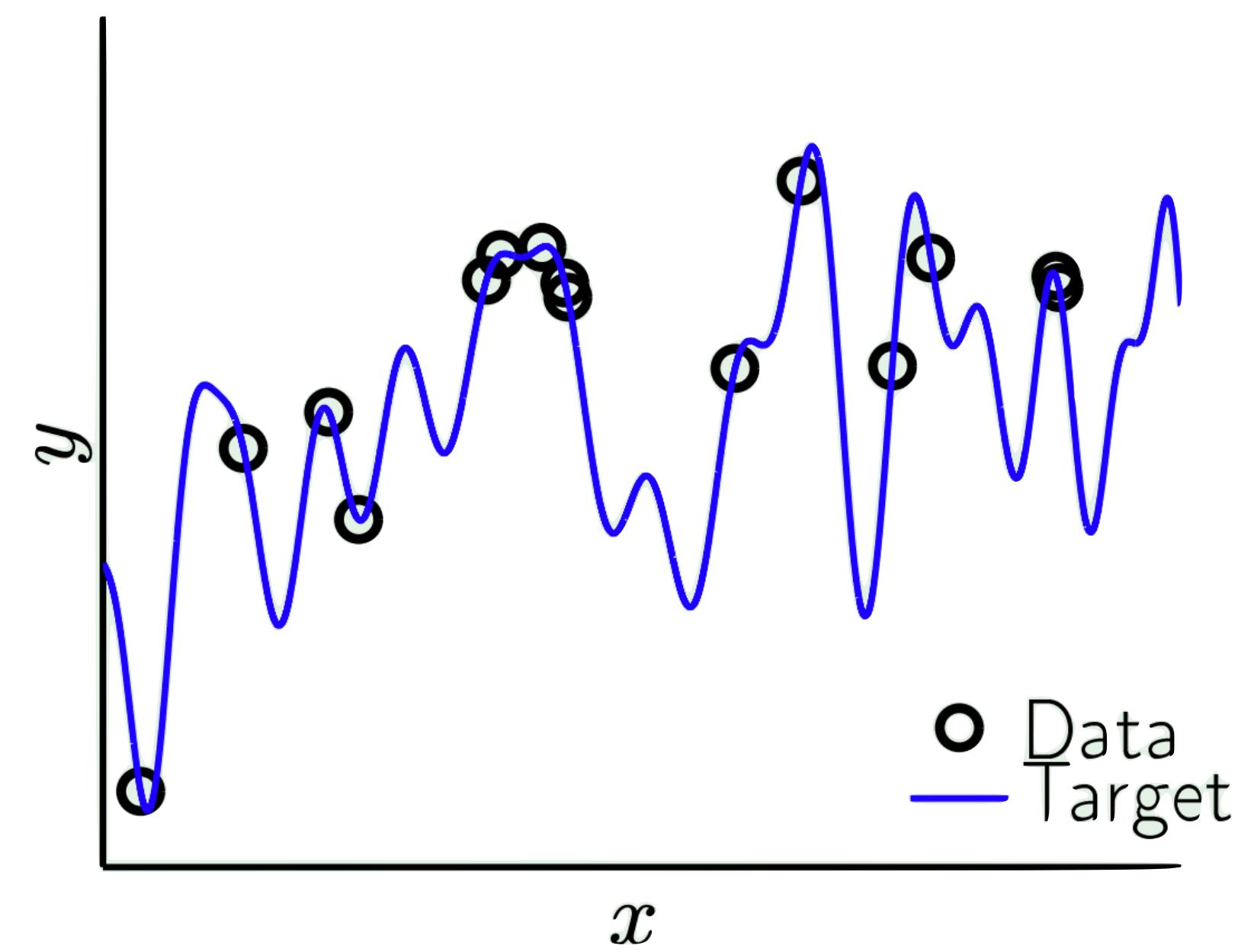
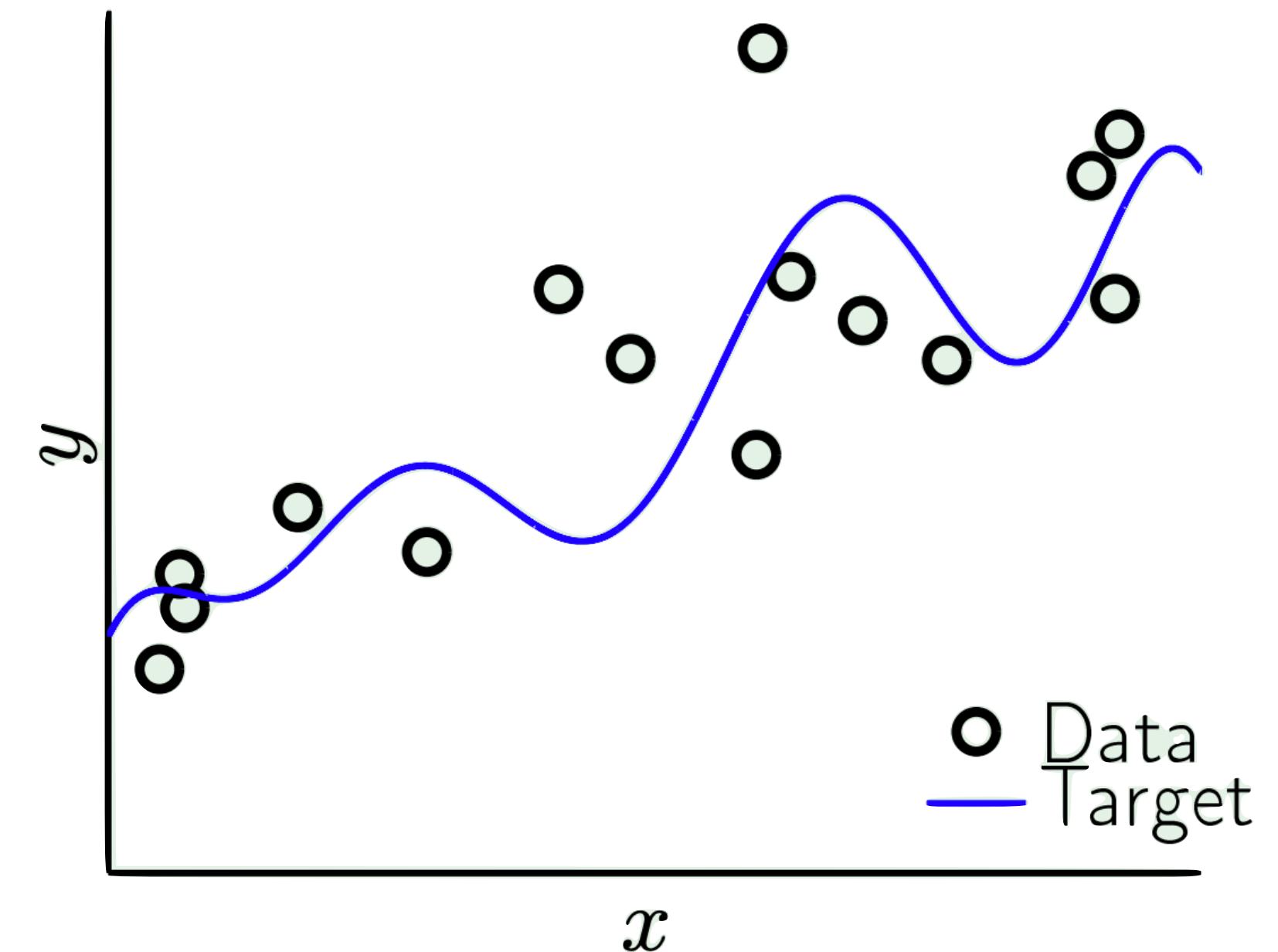
[Перевести твит](#)

6:01 AM · 24 нояб. 2016 г. · Twitter Web Client

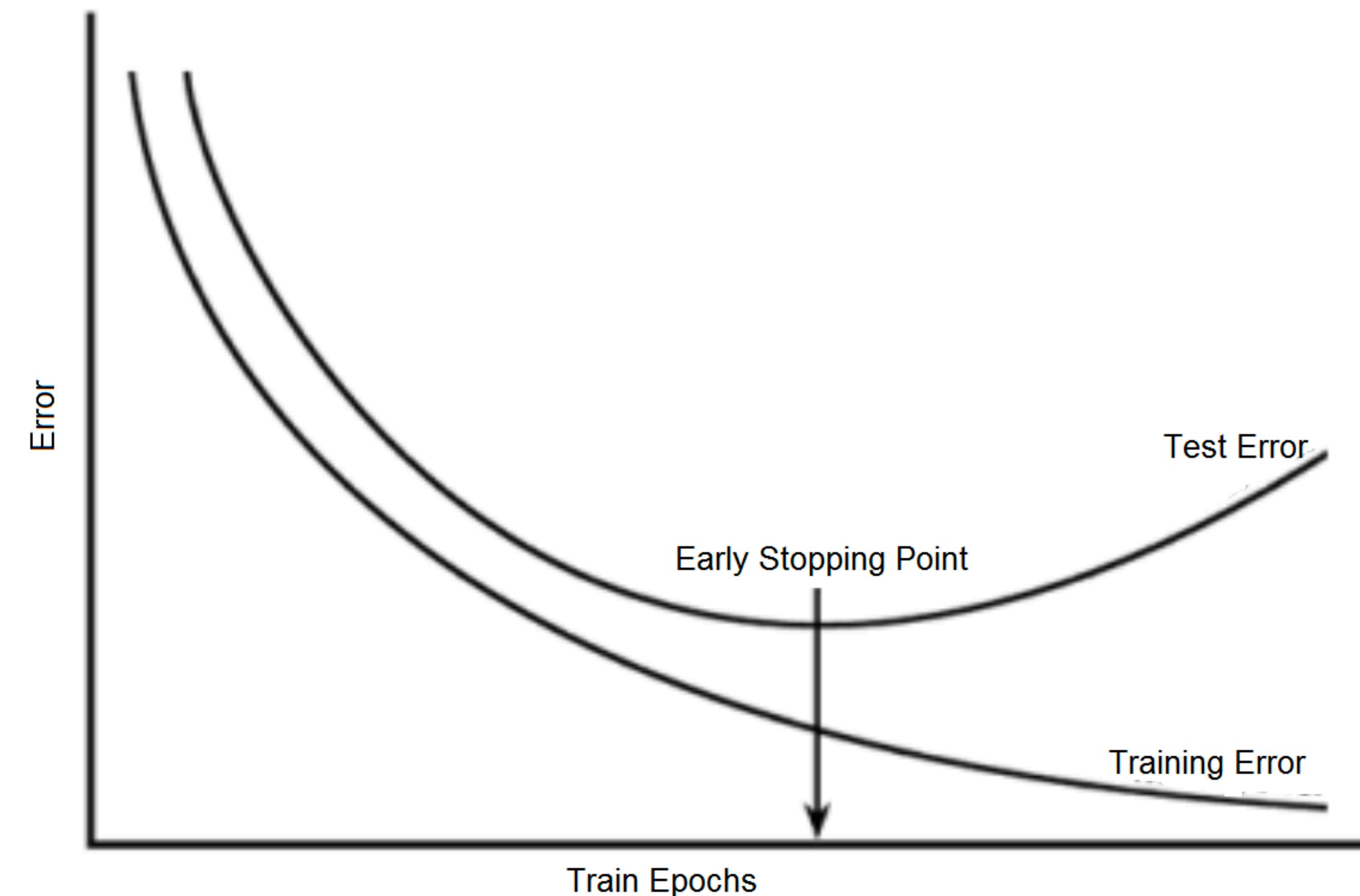
Регуляризация нейронных сетей



Переобучение



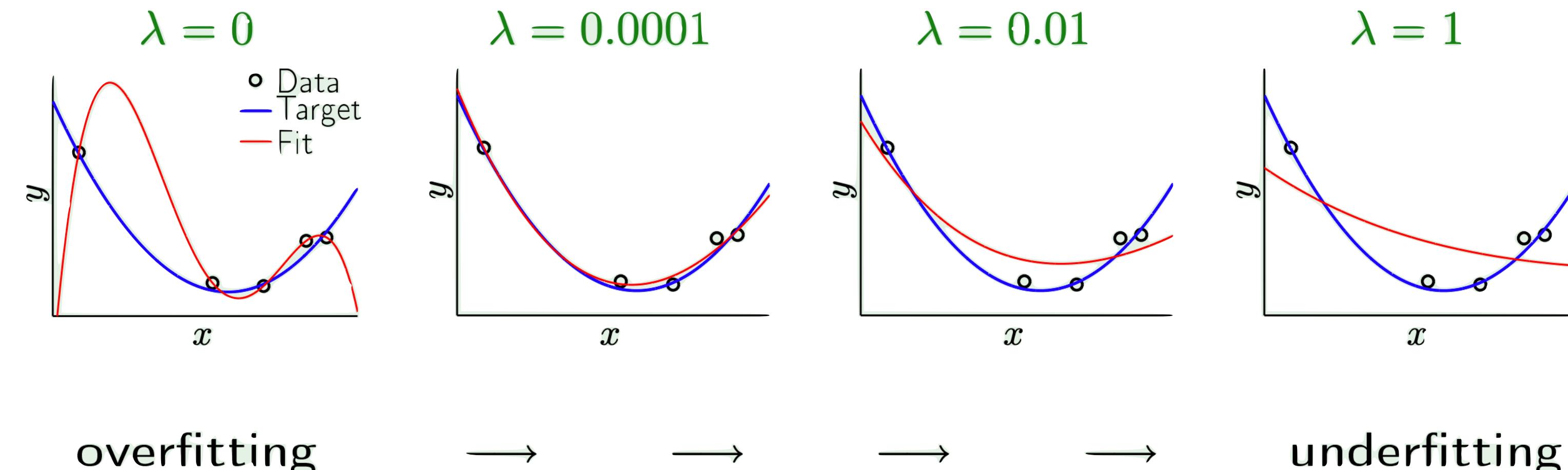
Early stopping



L2 регуляризация

$$\theta^* = \arg \min_{\theta} \sum_{i=1}^N l(f_{\theta}(x_i), y_i) + \lambda \sum_{j=1}^T \theta_j^2$$

- Интуиция: хотим, чтобы параметры модели были не очень большими
- λ – параметр регуляризации

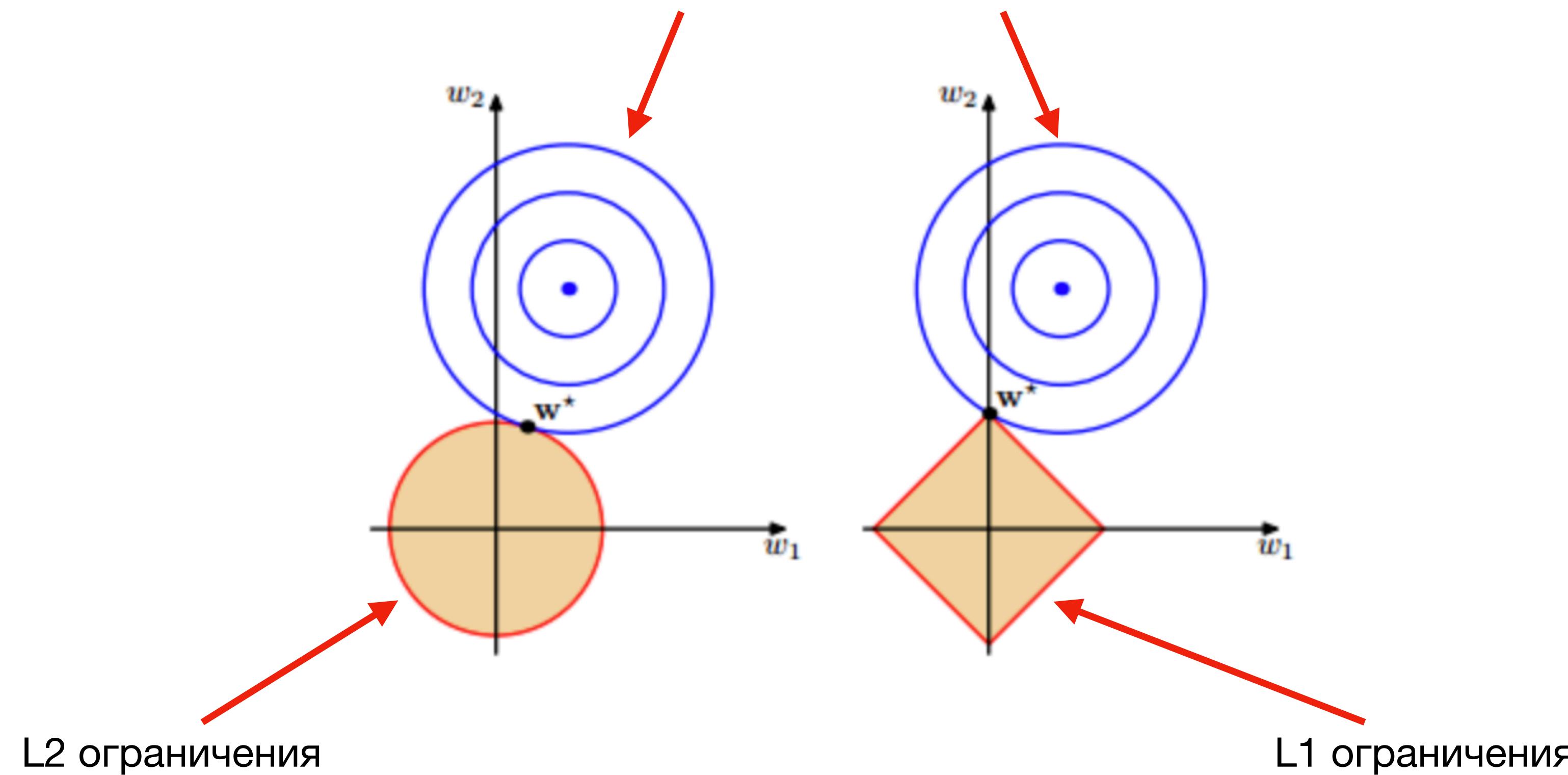


L1 регуляризация (LASSO)

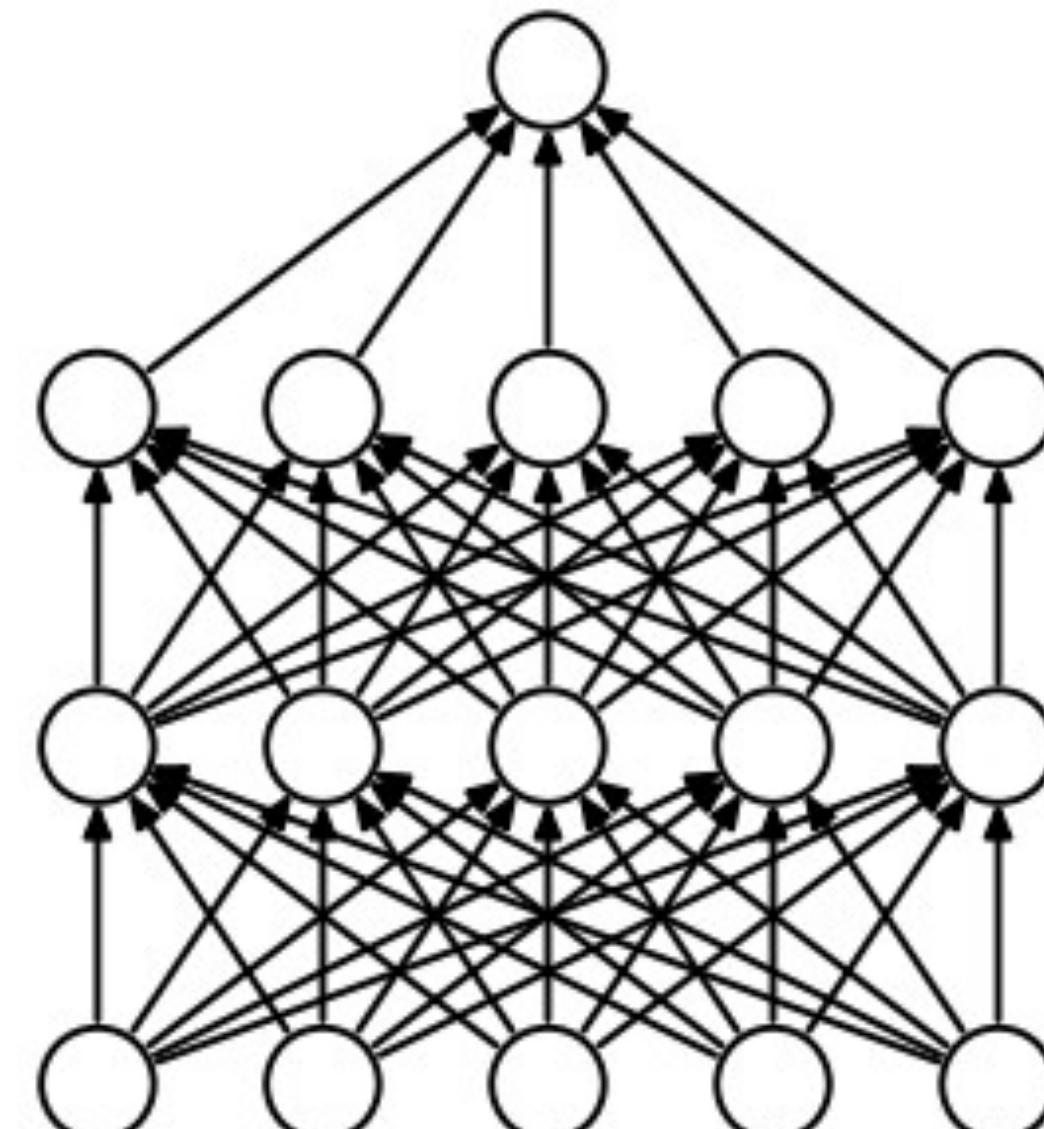
- $\theta^* = \arg \min_{\theta} \sum_{i=1}^N l(f_{\theta}(x_i), y_i) + \lambda \sum_{j=1}^T |\theta_j|$
- Интуиция: хотим, чтобы незначимые параметры модели были близки к 0
- λ — параметр регуляризации

L2 vs L1

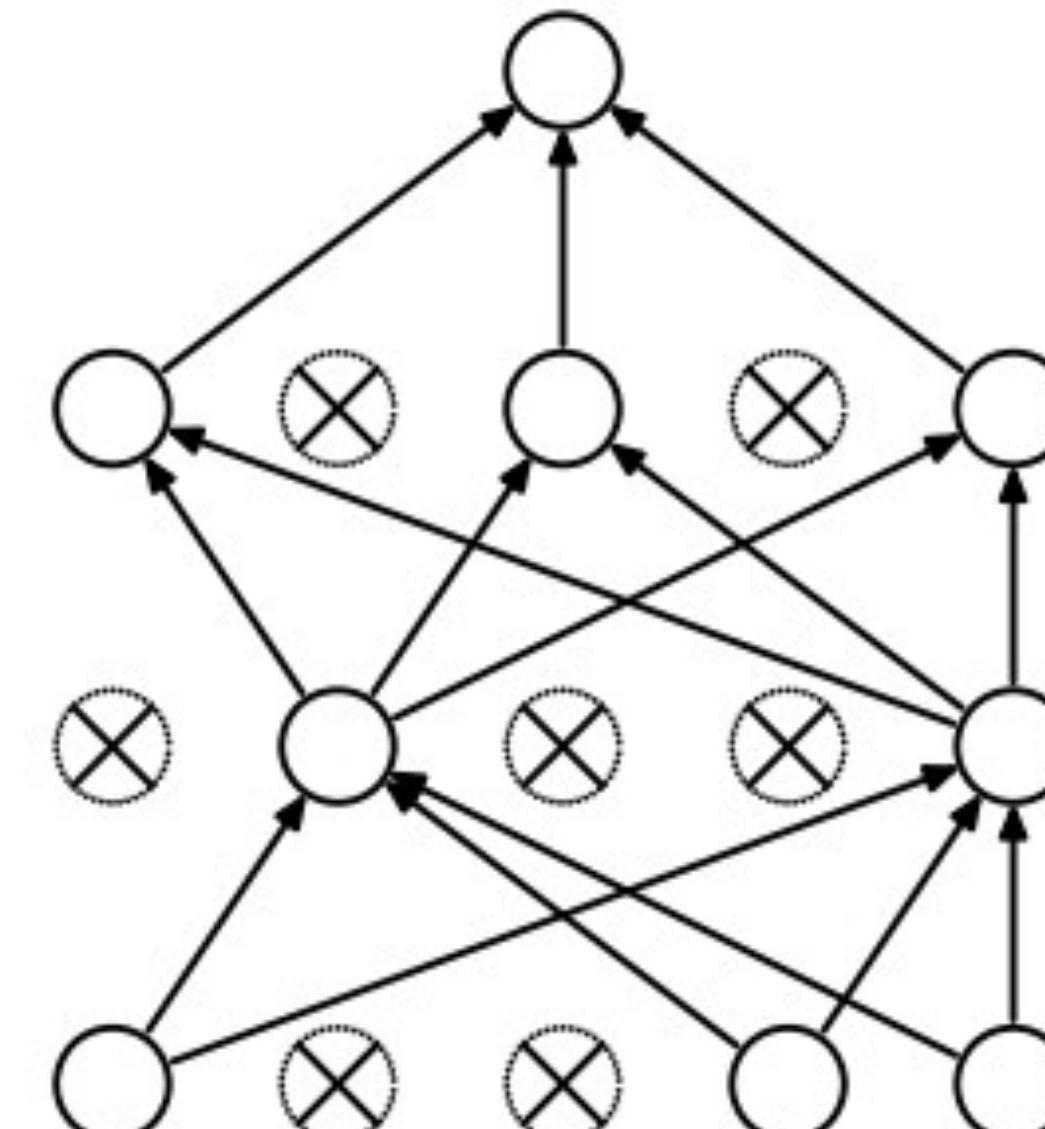
Функция потерь без регуляризации



Dropout (Srivastava et al., 2014)

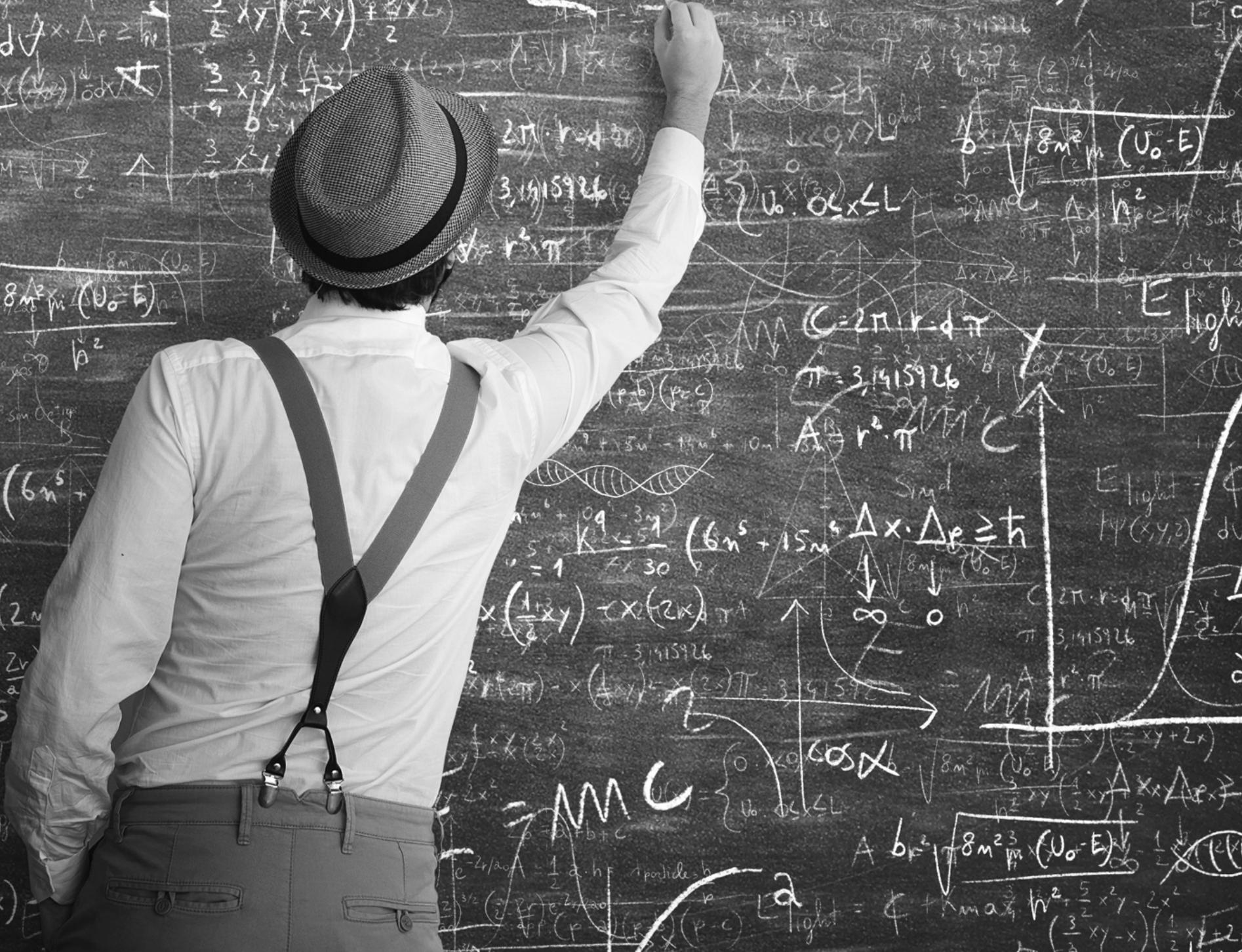


(a) Standard Neural Net



(b) After applying dropout.

Адаптация нейронных сетей с помощью оптимизации

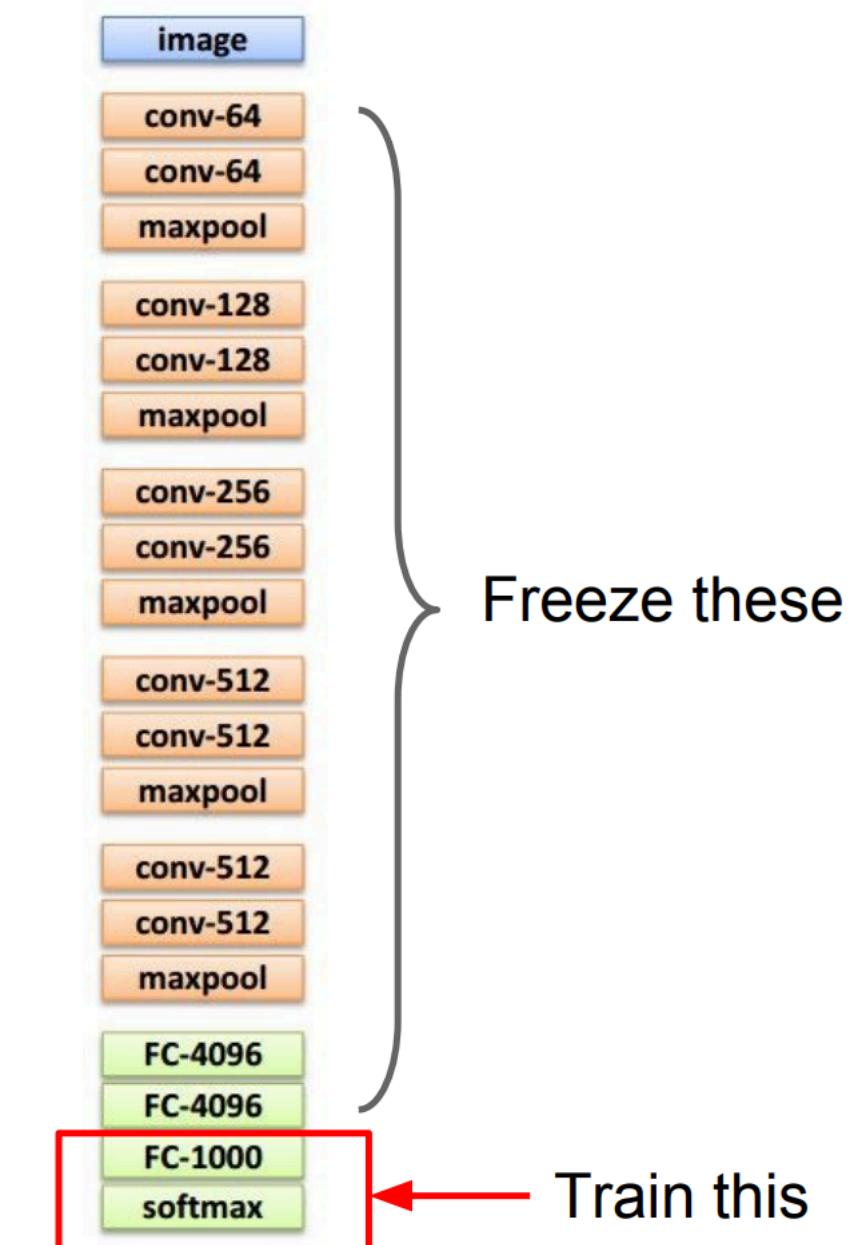


Transfer learning

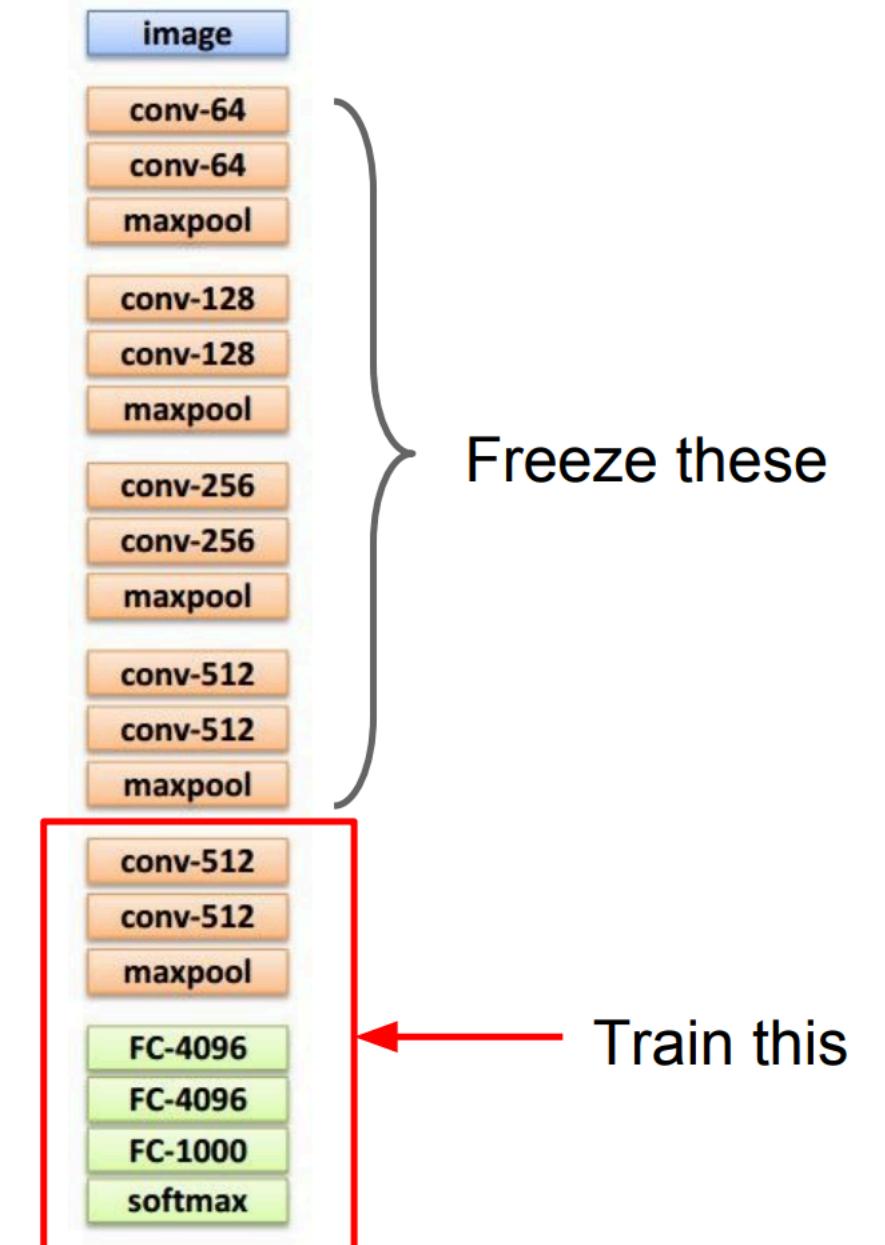
1. Обучим нейронную сеть
на базе ImageNet



2. Мало новых данных



3. Среднее количество
новых данных



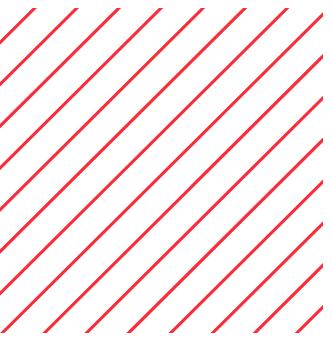
Freeze these

Train this

Freeze these

Train this

Небольшой learning rate



Заключение

- Постановка задачи оптимизации в глубоком обучении
- Сложности и особенности этой оптимизации
- Основные методы оптимизации
- Learning rate
- Регуляризация
- Transfer learning
- В следующий раз: сверхточные нейронные сети