

# 4. Градиентный спуск



# Линейная регрессия

Линейная модель вида

$$a(x) = w_0 + \langle w, x \rangle$$

Подбираем веса, которые минимизируют ошибку, например, MSE

$$\text{MSE}(a, X) = \frac{1}{l} \sum_{i=1}^l (a(x_i) - y_i)^2$$

Имеет аналитическое решение, минимизирующее MSE:

$$w = (X^T X)^{-1} X^T y$$

👉  $(X^T X)^{-1}$  — сложно, долго, зачастую невозможно

👉 Красивое решение есть для MSE, для других функционалов может не существовать

👉 Для борьбы с переобучением добавляем регуляризаторы, например, нормы весов

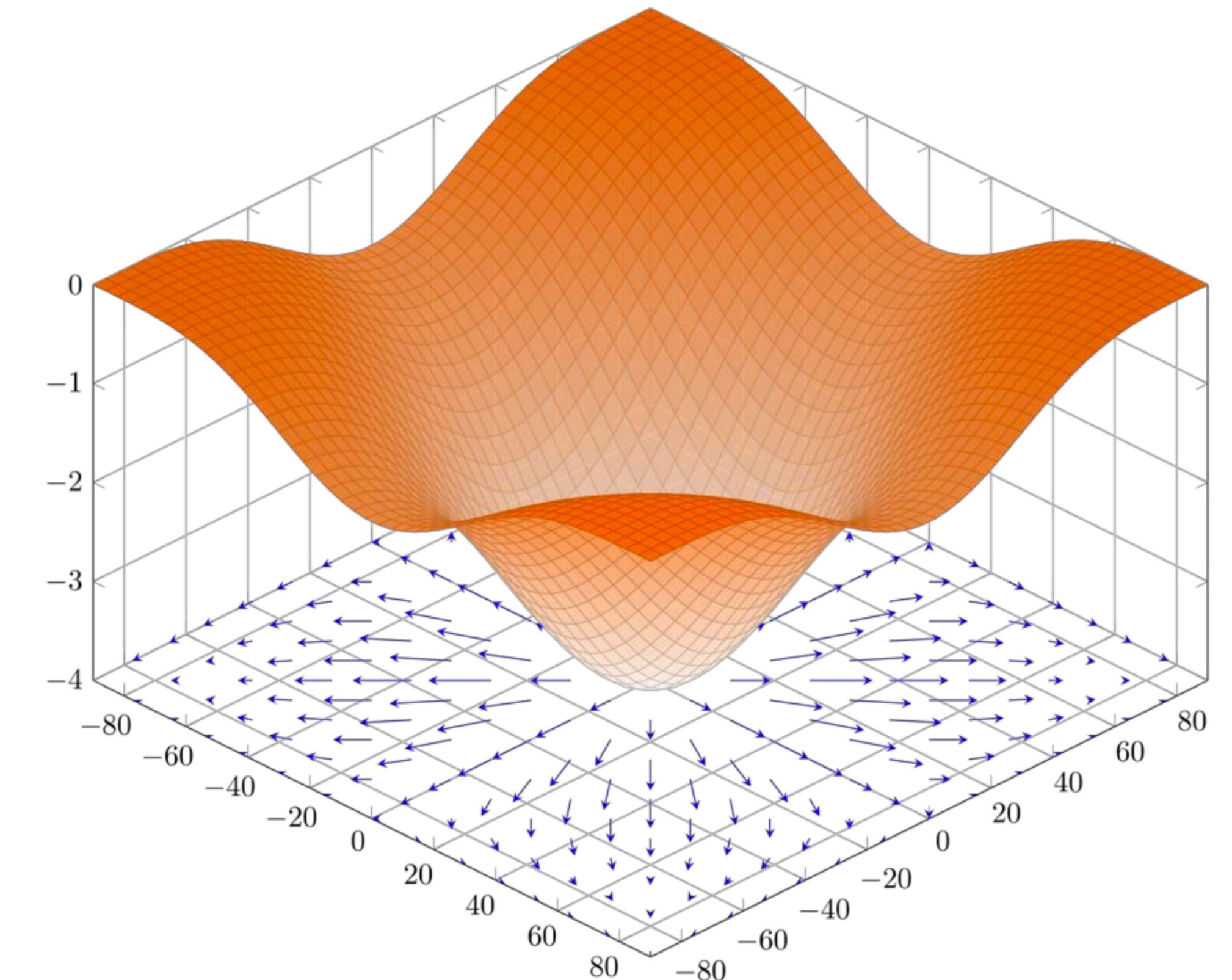
# Градиент

Вектор частных производных

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

Зафиксируем  $x_0$ :

- 👉  $\nabla f(x_0)$  указывает в сторону наиболее быстрого роста функции
- 👉 А антиградиент  $(-\nabla f(x_0))$  — в сторону наискорейшего убывания



# Как это помогает?



# **Градиентный спуск**

# Градиентный спуск

1. Стартуем из случайной точки

Инициализуем как-то  $w$

2. Вычисляем градиент в этой точке —  $\nabla Q(w^t)$

функционал ошибки должен быть

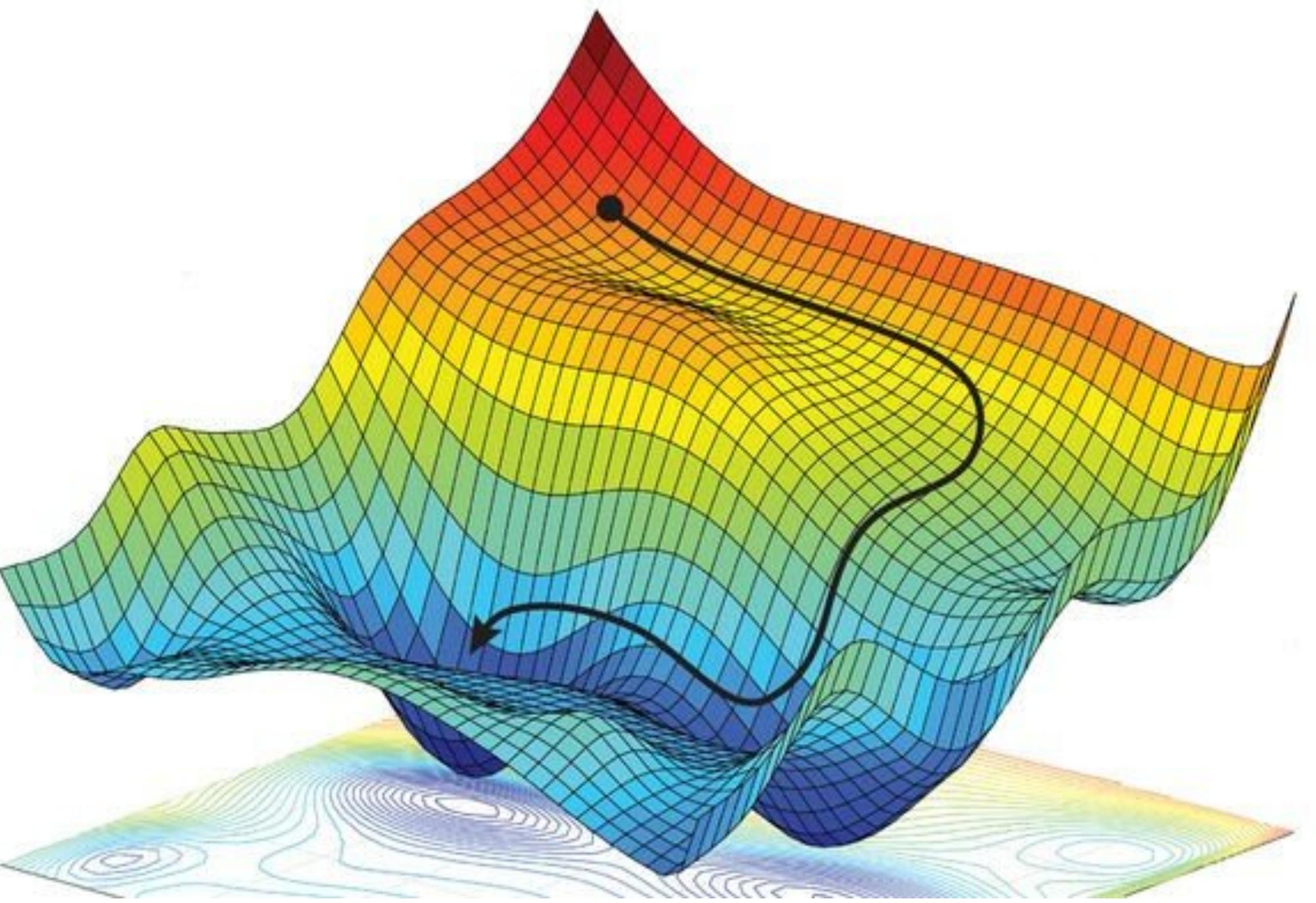
**дифференцируемым!**

3. Сдвигаемся по антиградиенту

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

4. Повторяем пока не окажемся в локальном  
минимуме

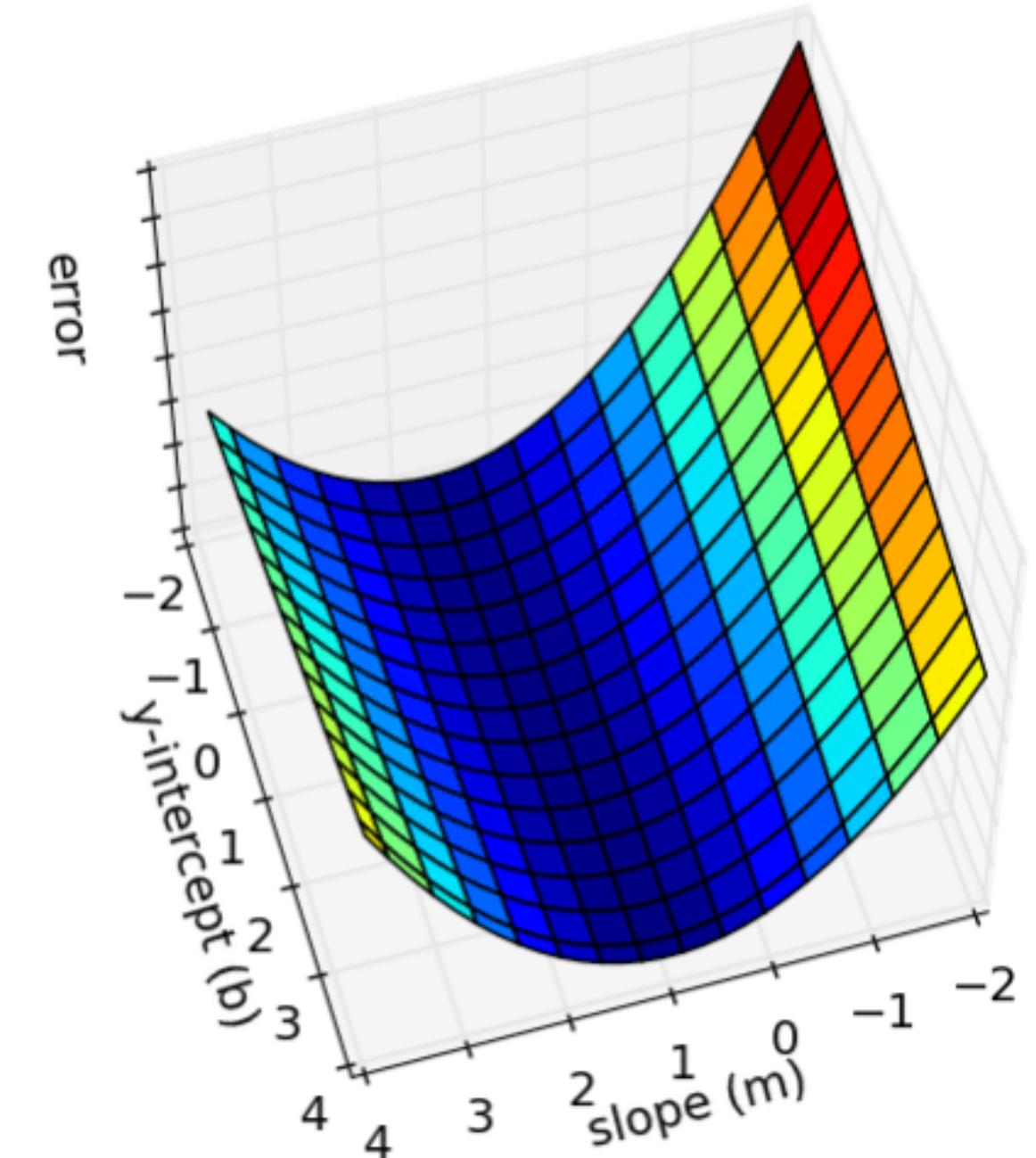
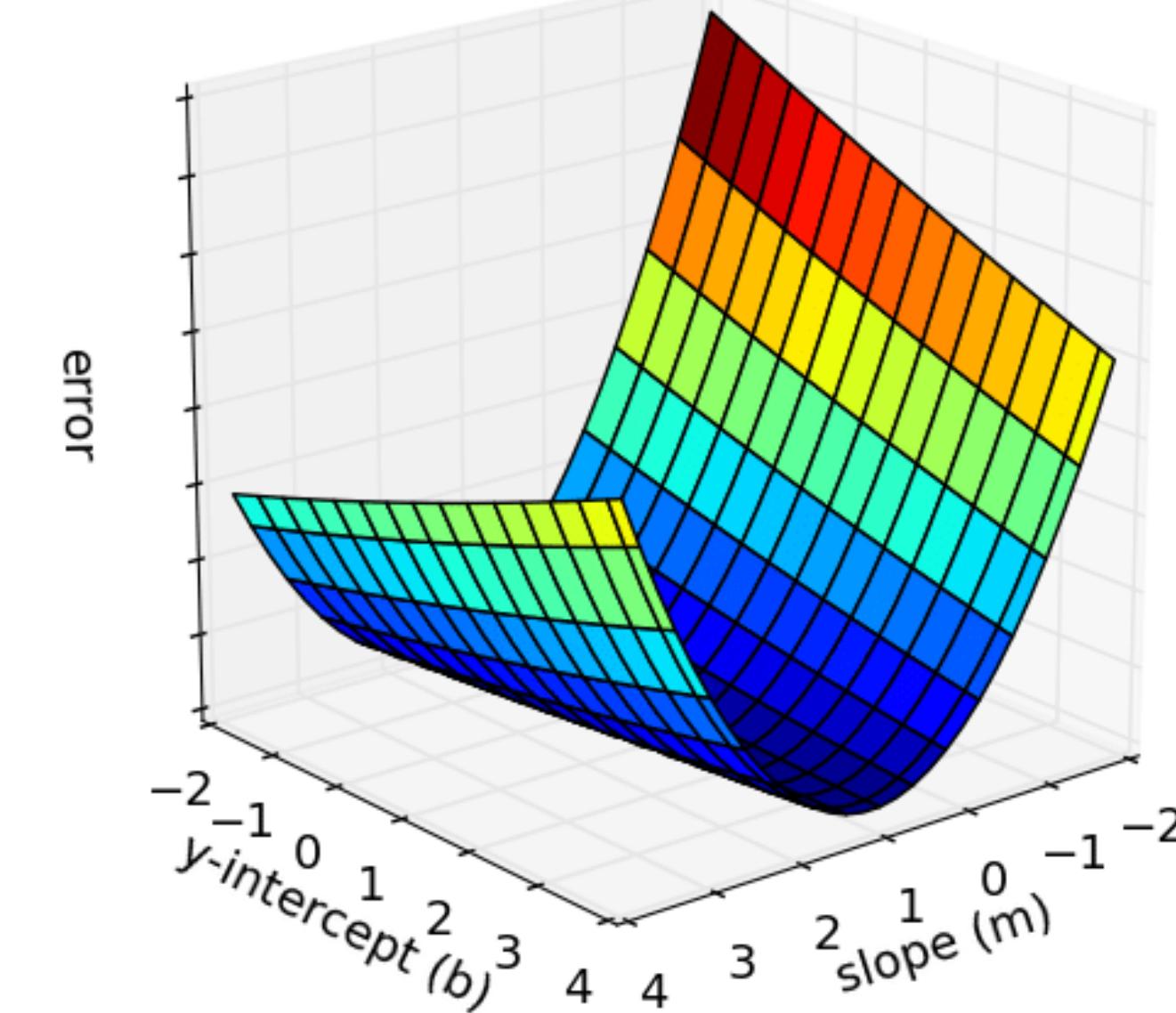
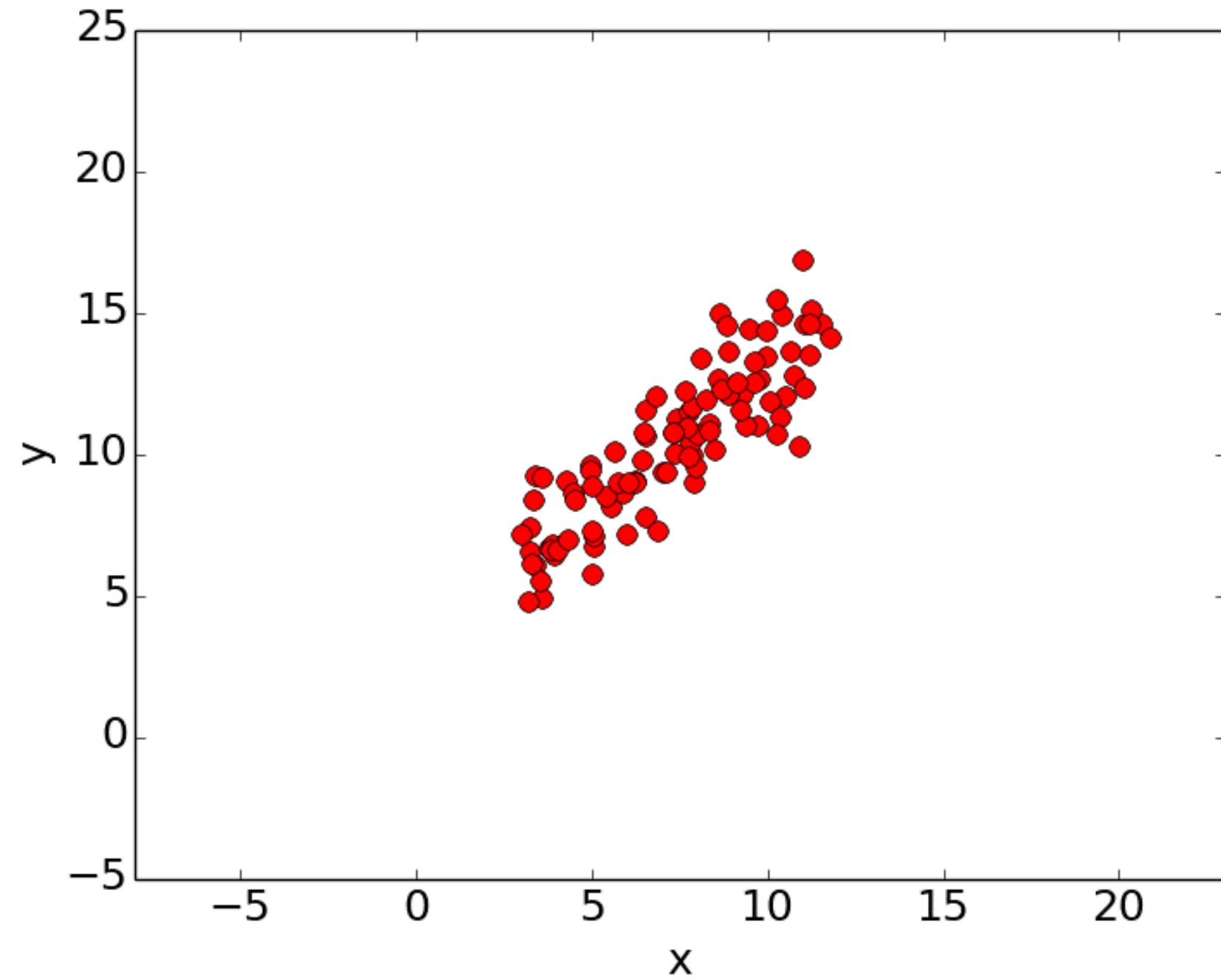
$\eta$  — learning rate / размер шага, гипер-параметр



# Парная регрессия

Модель:  $a(x) = mx + b$

Функционал ошибки:  $Q(m, b) = \frac{1}{l} \sum_{i=1}^l (mx_i + b - y_i)^2$



# Парная регрессия

$$Q(m, b) = \frac{1}{l} \sum_{i=1}^l (mx_i + b - y_i)^2$$

👉  $\frac{\partial Q}{\partial m} = \frac{2}{l} \sum_{i=1}^l x_i(mx_i + b - y_i)$

👉  $\frac{\partial Q}{\partial b} = \frac{2}{l} \sum_{i=1}^l (mx_i + b - y_i)$

$$\nabla Q(m, b) = \left( \frac{2}{l} \sum_{i=1}^l x_i(mx_i + b - y_i), \frac{2}{l} \sum_{i=1}^l (mx_i + b - y_i) \right)$$

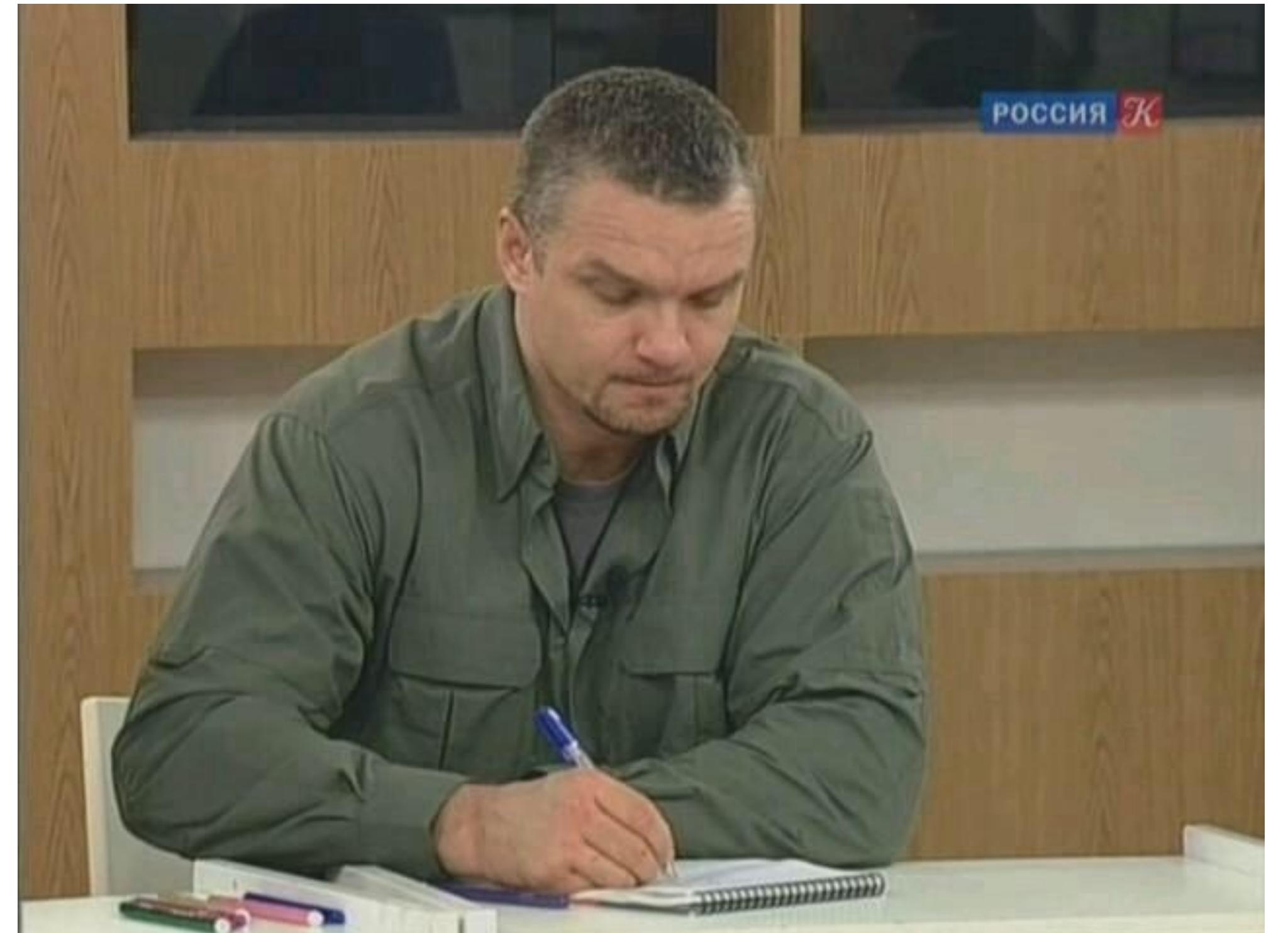
# Технические детали

## Инициализация весов

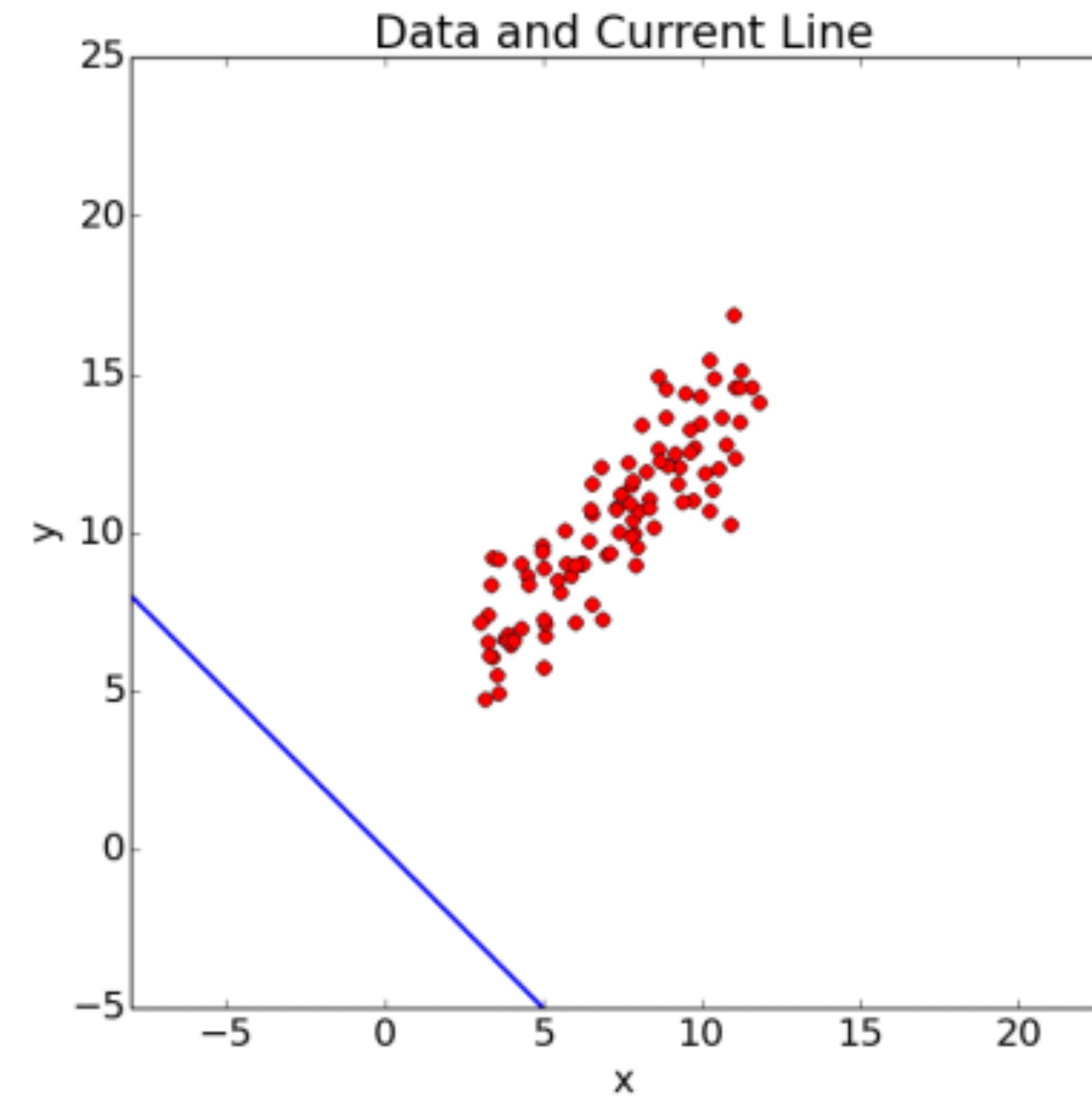
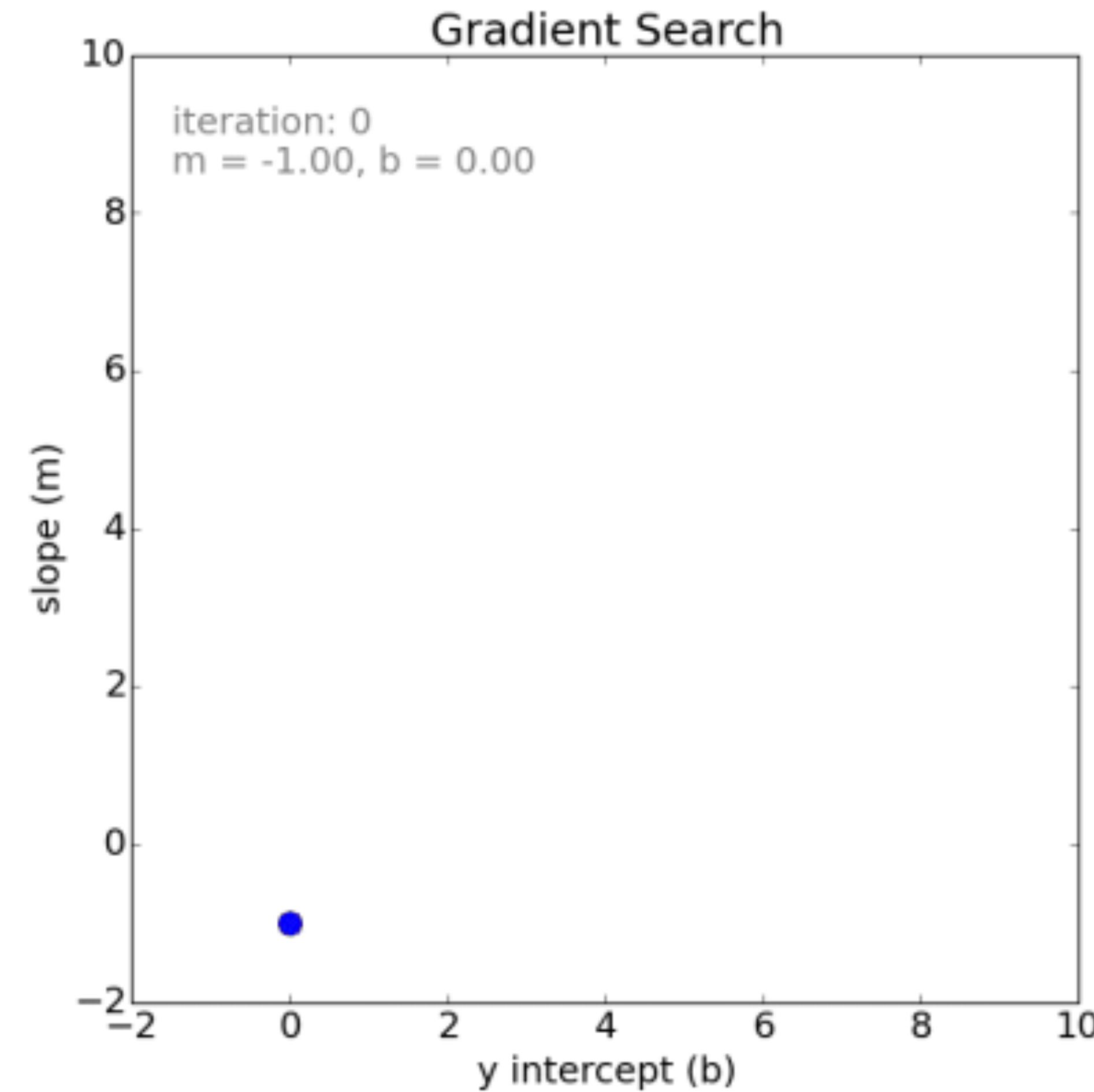
- 👉 Можно брать из стандартного нормального распределения
- 👉 Что если все инициализировать нулями? А просто одинаковым числом?
- 👉 Bias полезно инициализировать нулем

## Сходимость

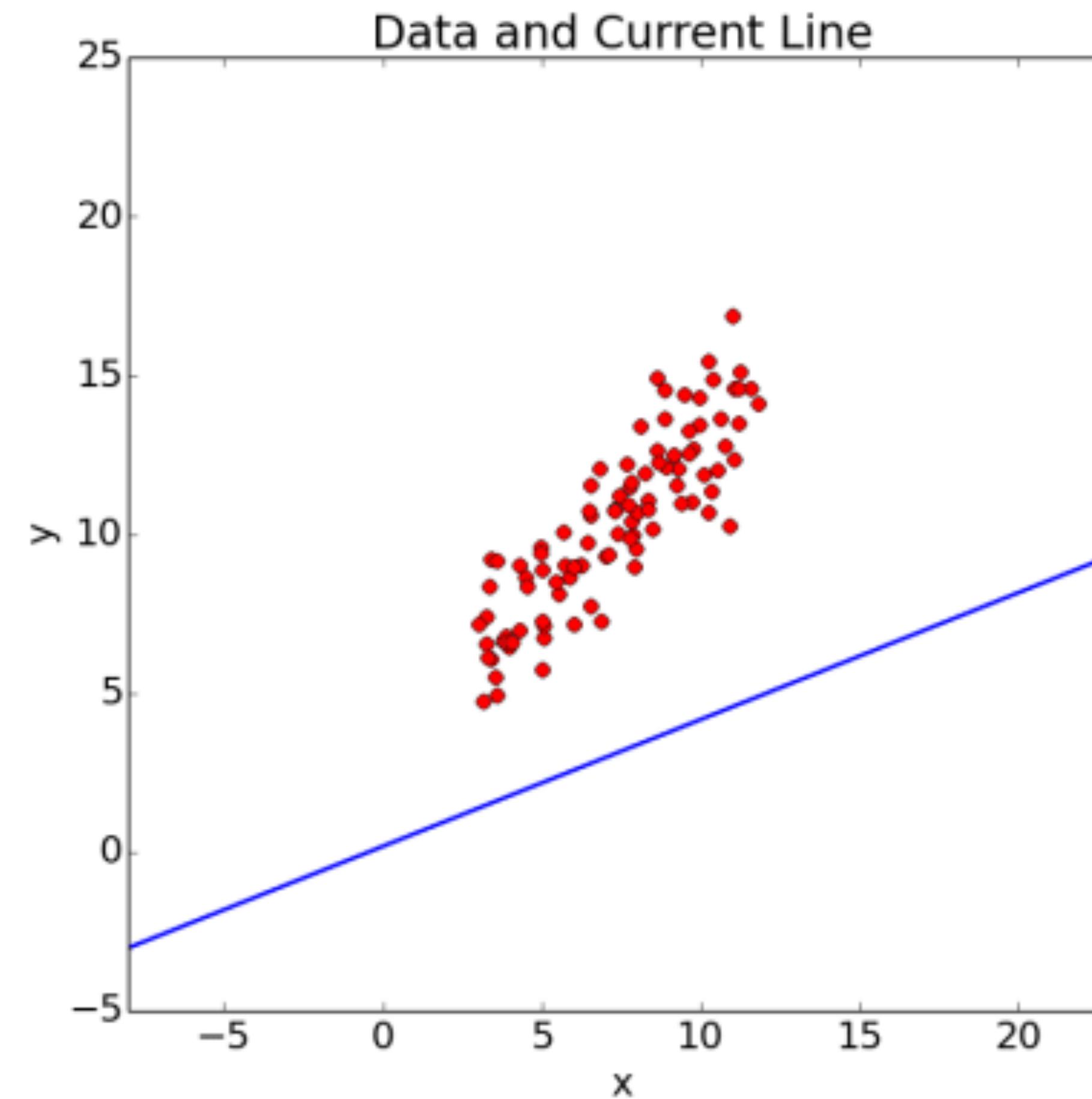
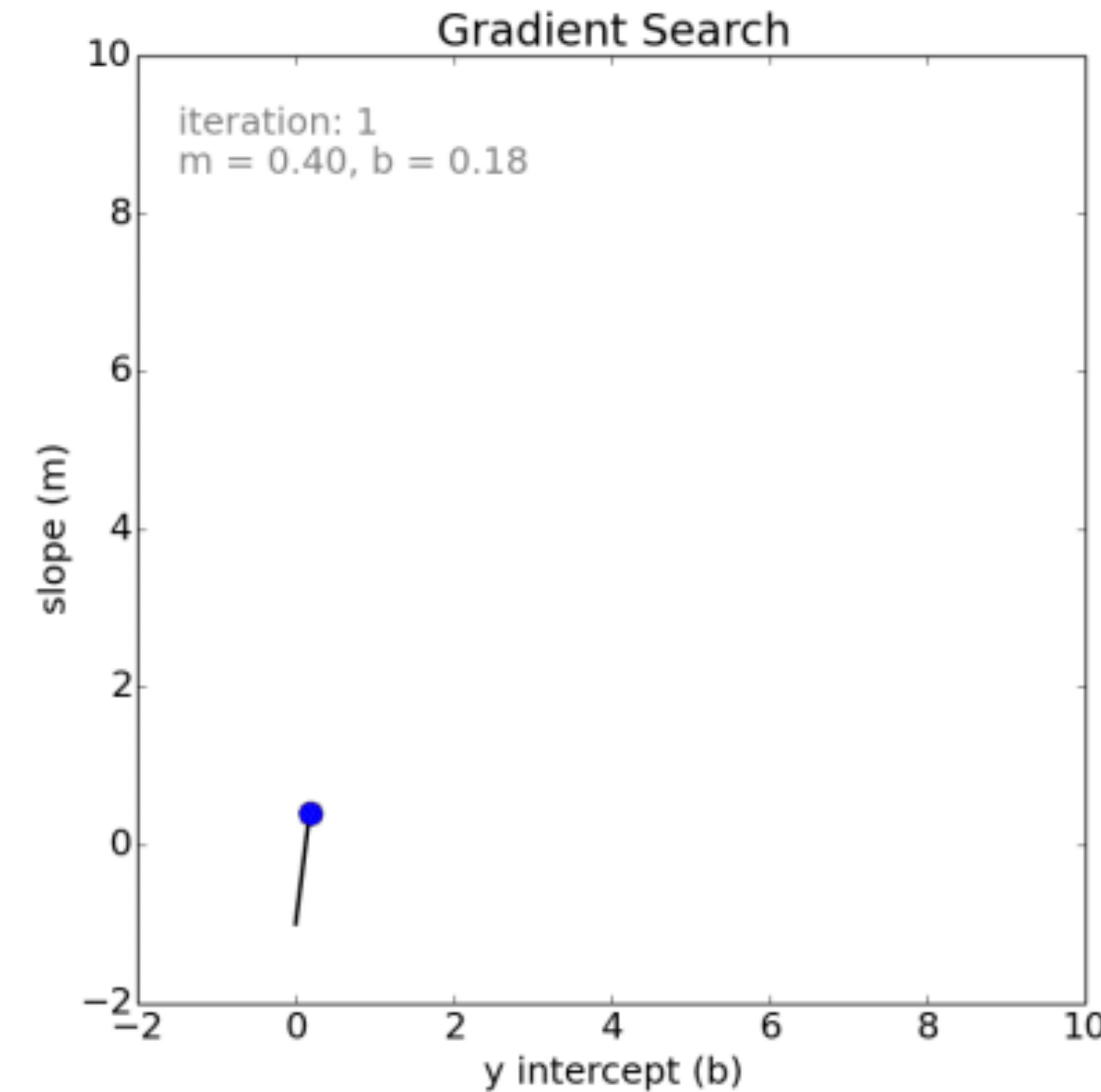
- 👉 До локального минимума может быть далеко, непонятно как определить
- 👉 Останавливаем процесс, если
  - $\|w^t - w^{t-1}\| < \epsilon$
  - $\|\nabla Q(w^t)\| < \epsilon$
  - Пока уменьшается ошибка на валидации



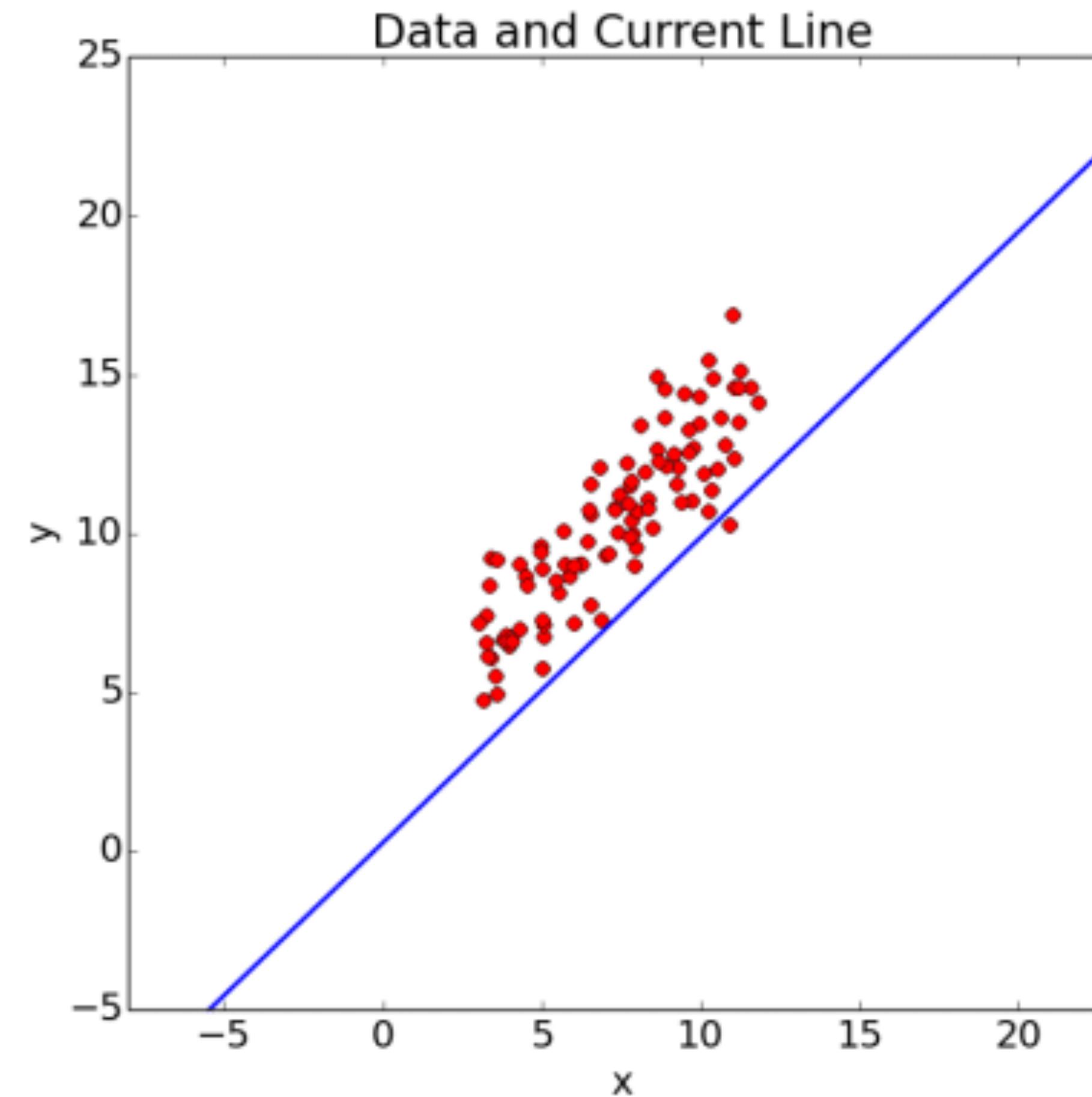
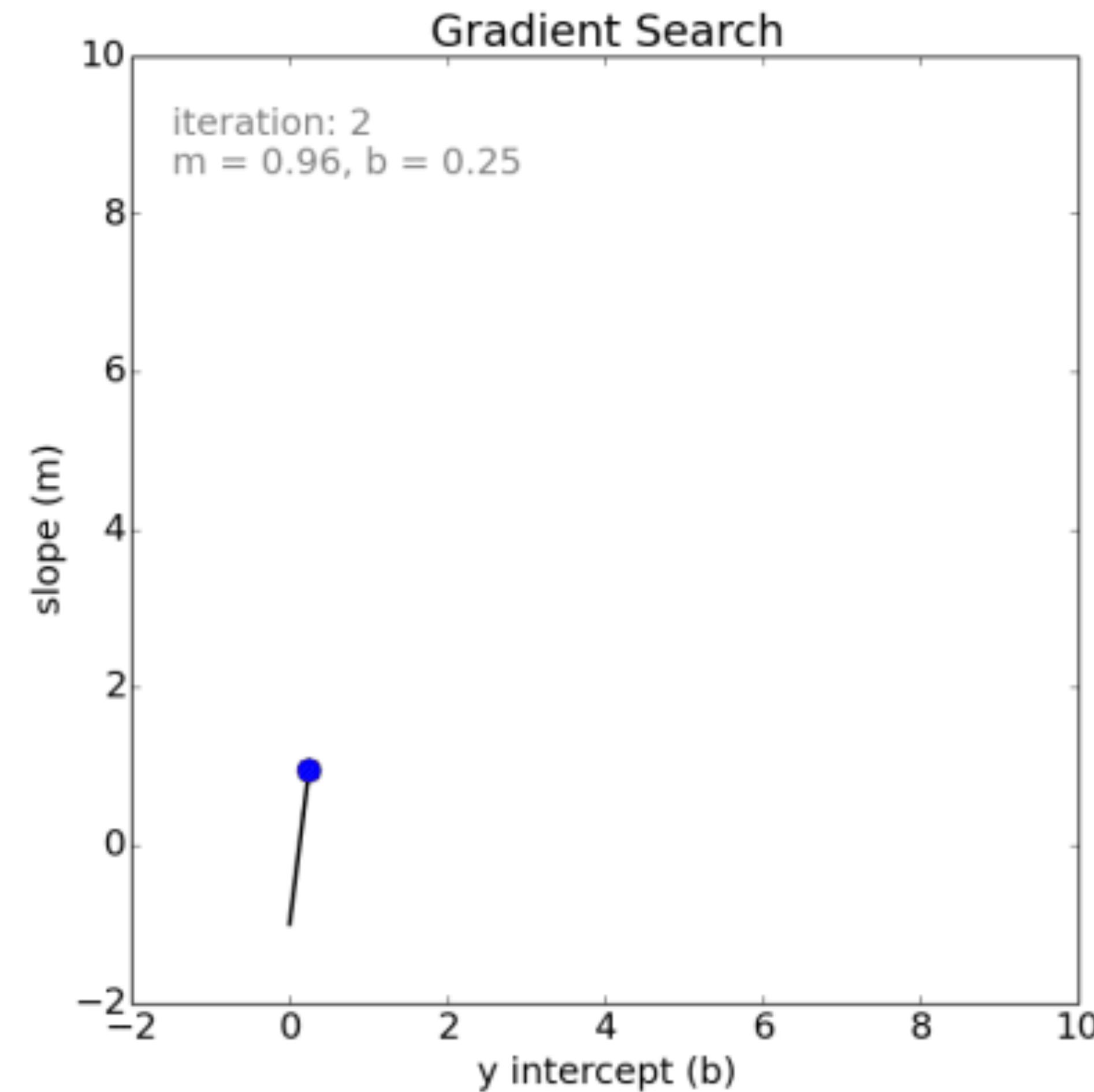
# Небольшой пример



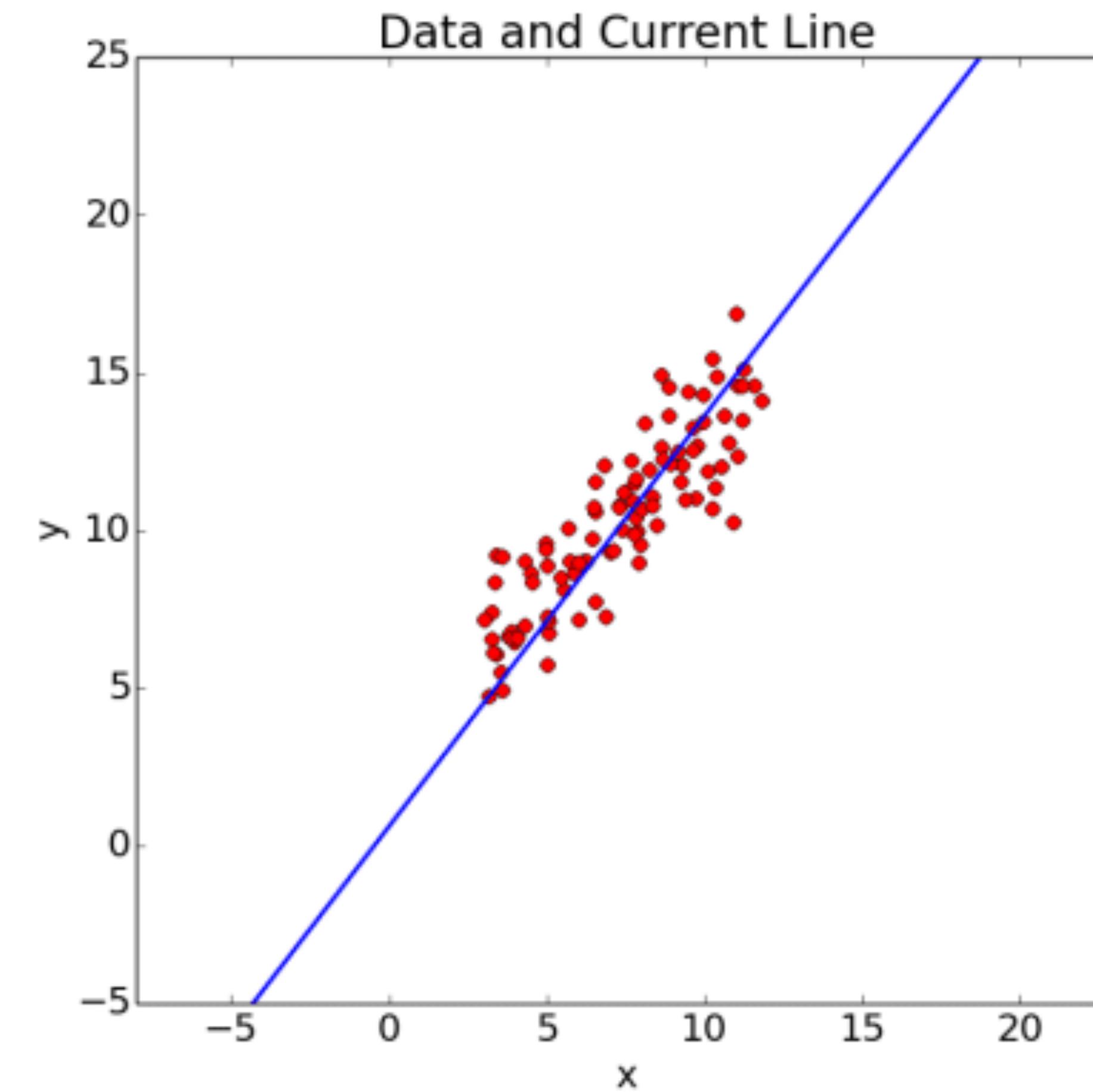
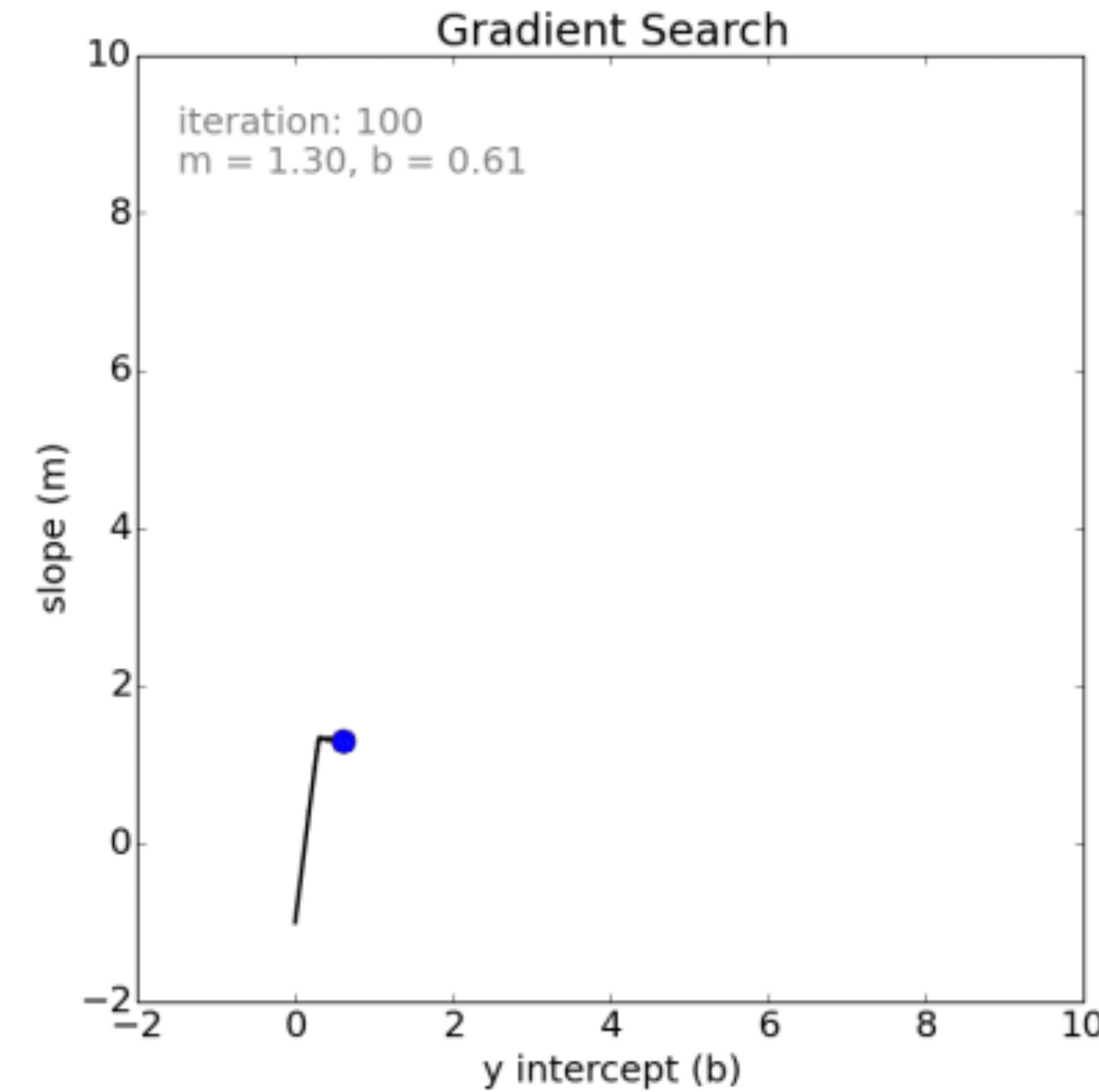
# Небольшой пример



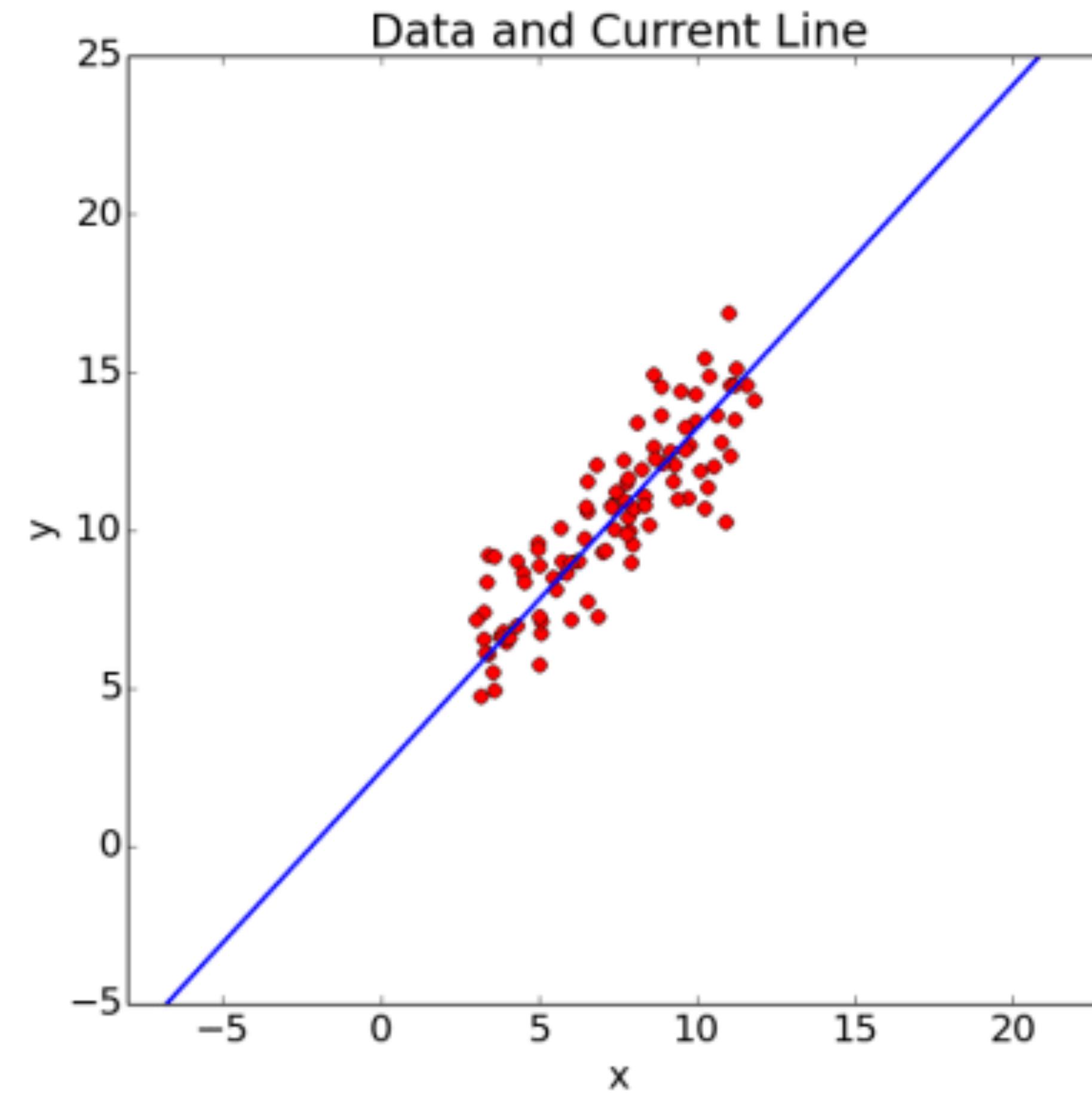
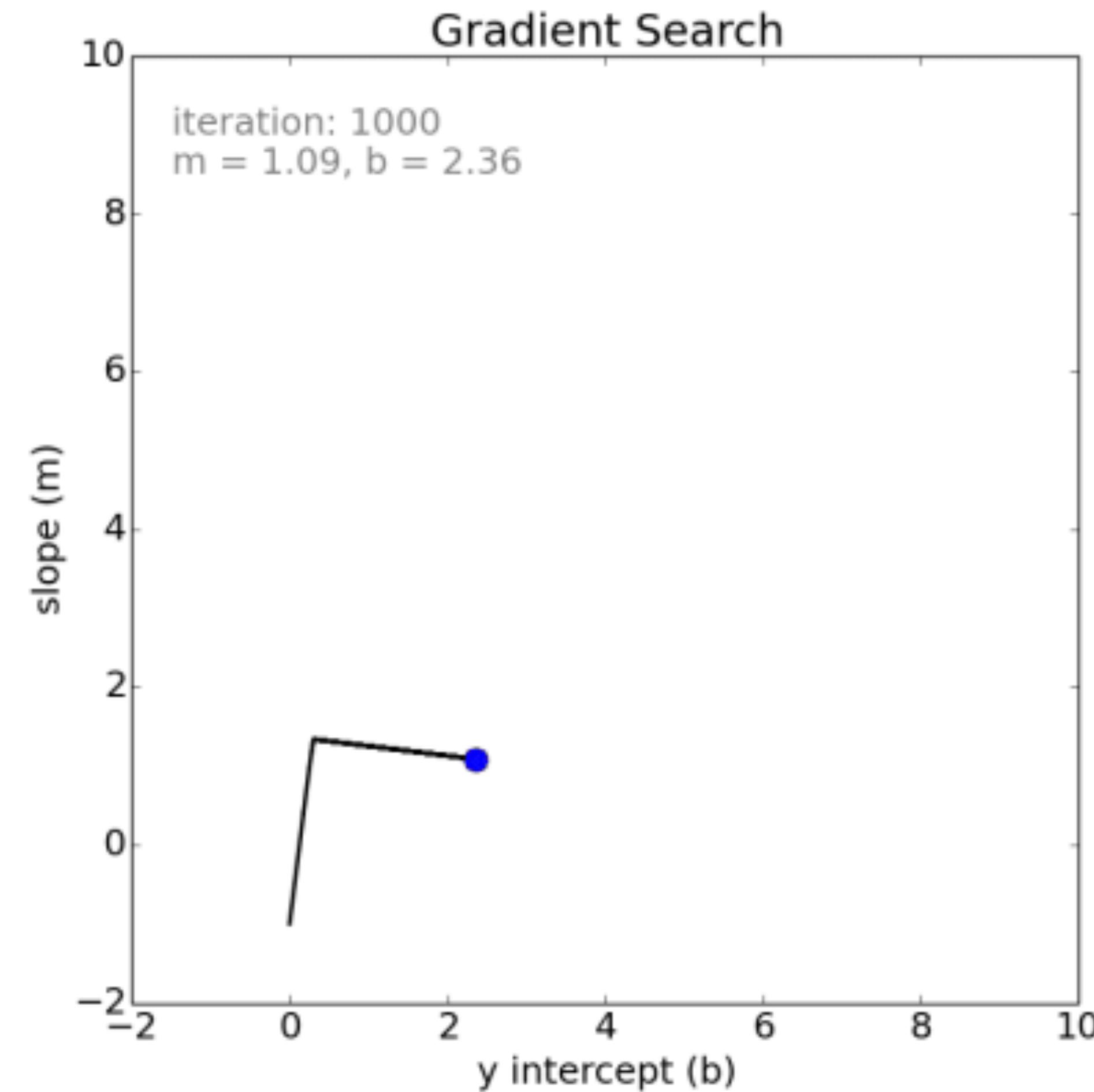
# Небольшой пример



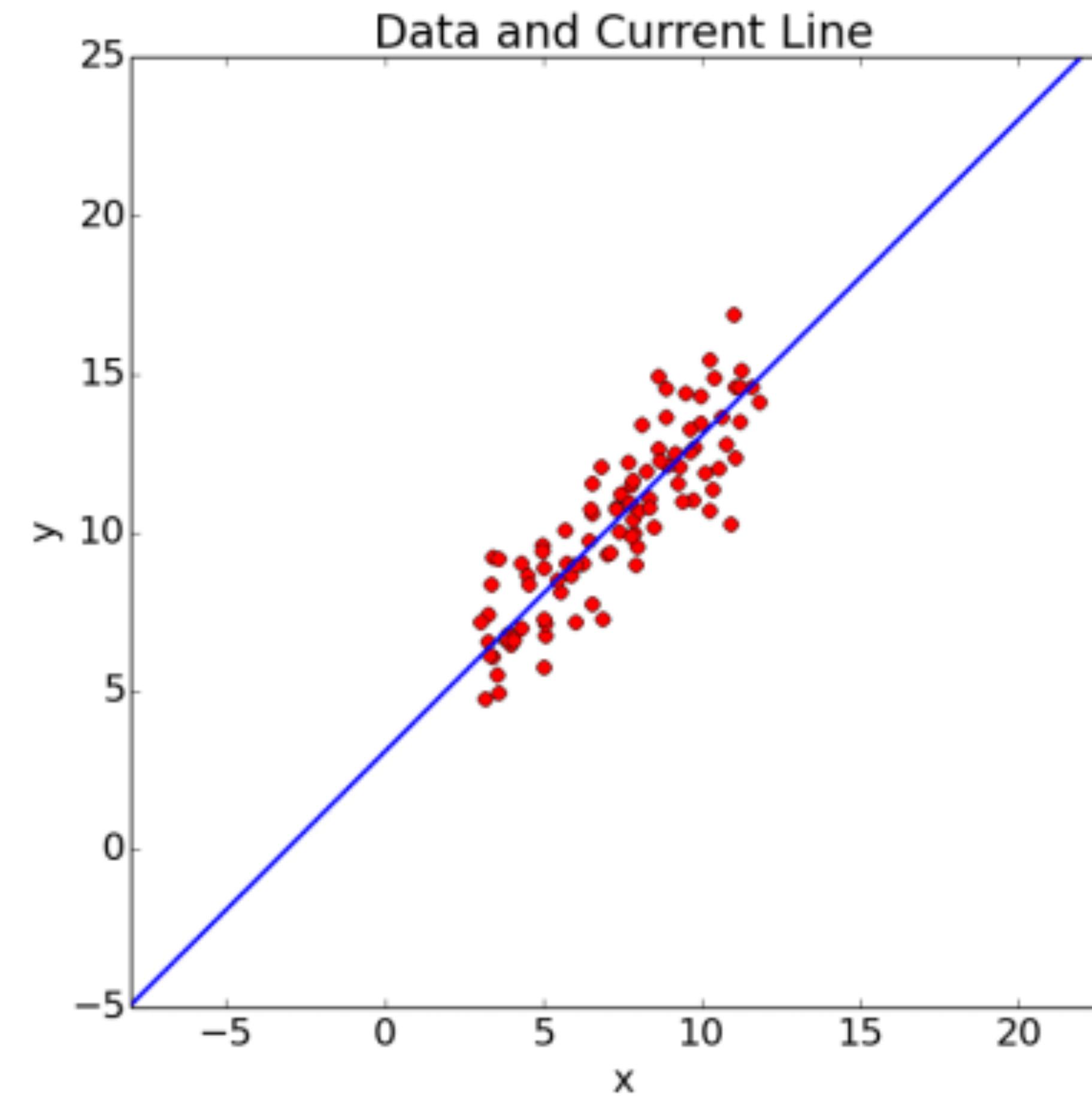
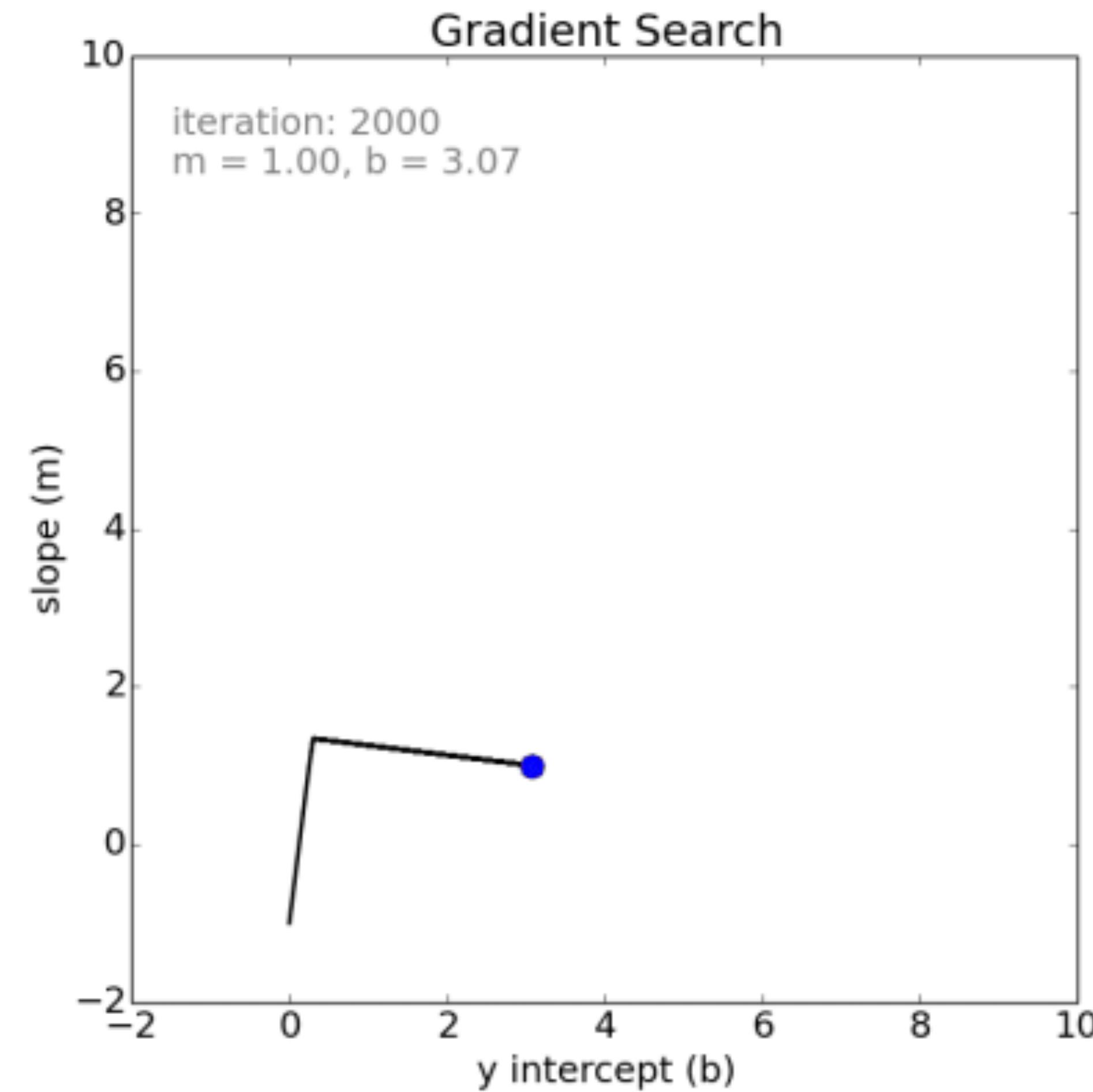
# Небольшой пример



# Небольшой пример

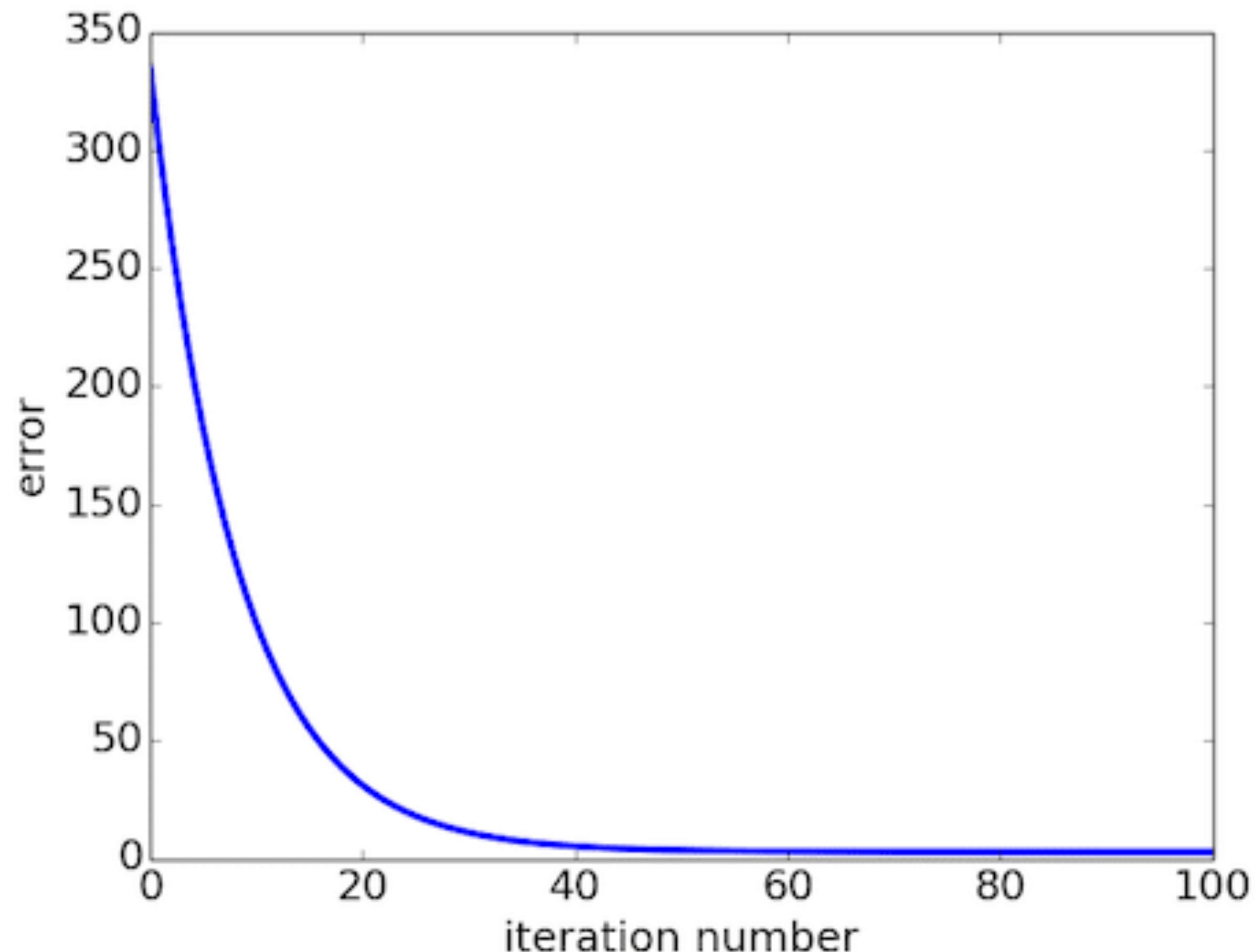


# Небольшой пример



# Небольшой пример

Поведение функционала ошибки — MSE



# MSE и общий случай

$$Q_{\text{MSE}}(w) = \frac{1}{l} \sum_{i=1}^l (\langle w, x_i \rangle - y_i)^2$$

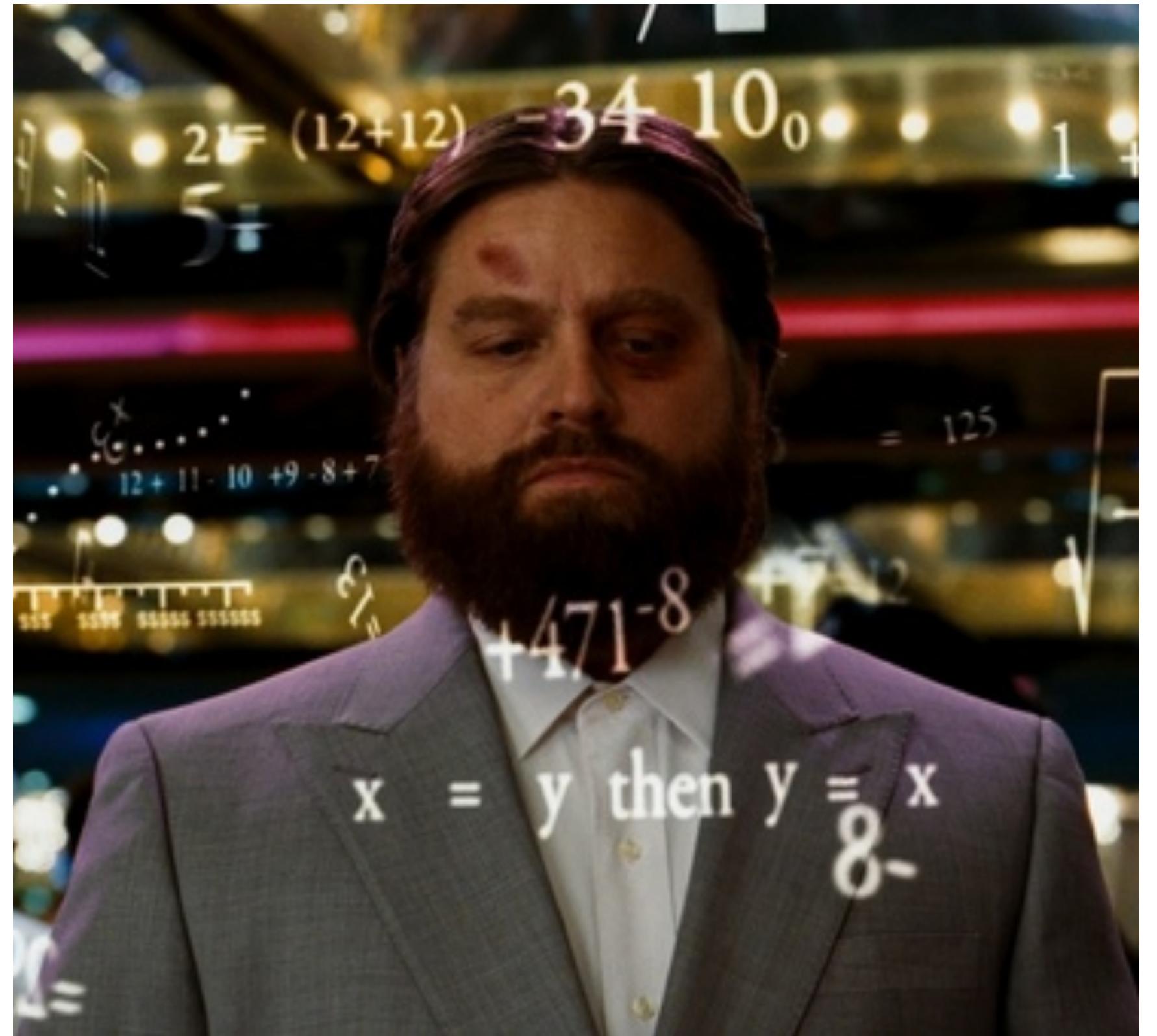
$$1. \frac{\partial Q}{\partial w_1} = \frac{2}{l} \sum_{i=1}^l x_{i1}(\langle w, x_i \rangle - y_i)$$

$$2. \frac{\partial Q}{\partial w_2} = \frac{2}{l} \sum_{i=1}^l x_{i2}(\langle w, x_i \rangle - y_i)$$

3. ...

$$4. \frac{\partial Q}{\partial w_d} = \frac{2}{l} \sum_{i=1}^l x_{id}(\langle w, x_i \rangle - y_i)$$

$$\nabla Q_{\text{MSE}}(w) = \frac{2}{l} X^T(Xw - y)$$



# Локальные минимумы

Инициализируем веса —  $w_0$

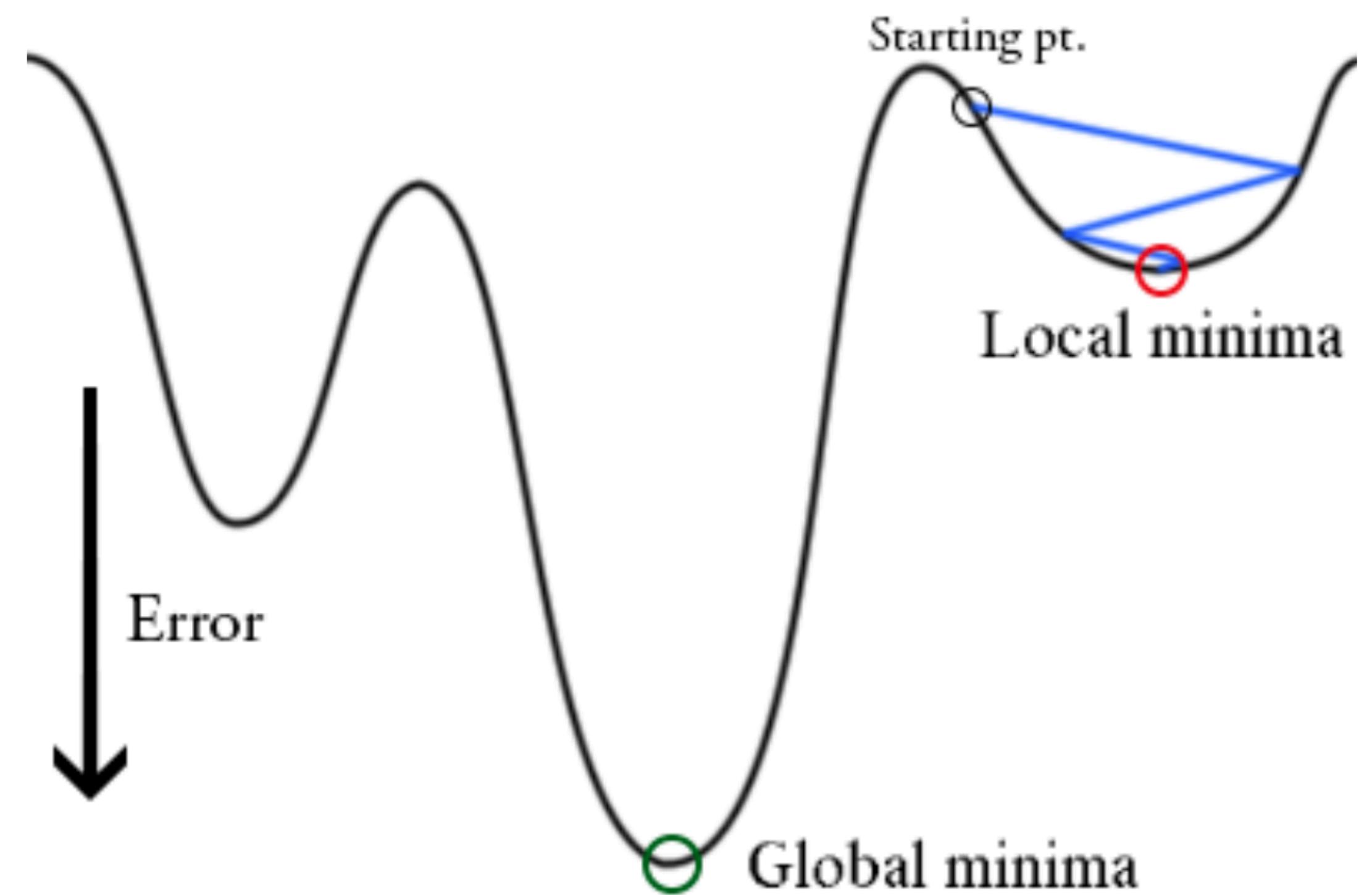
Повторяем до сходимости

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

Останавливаемся, если

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

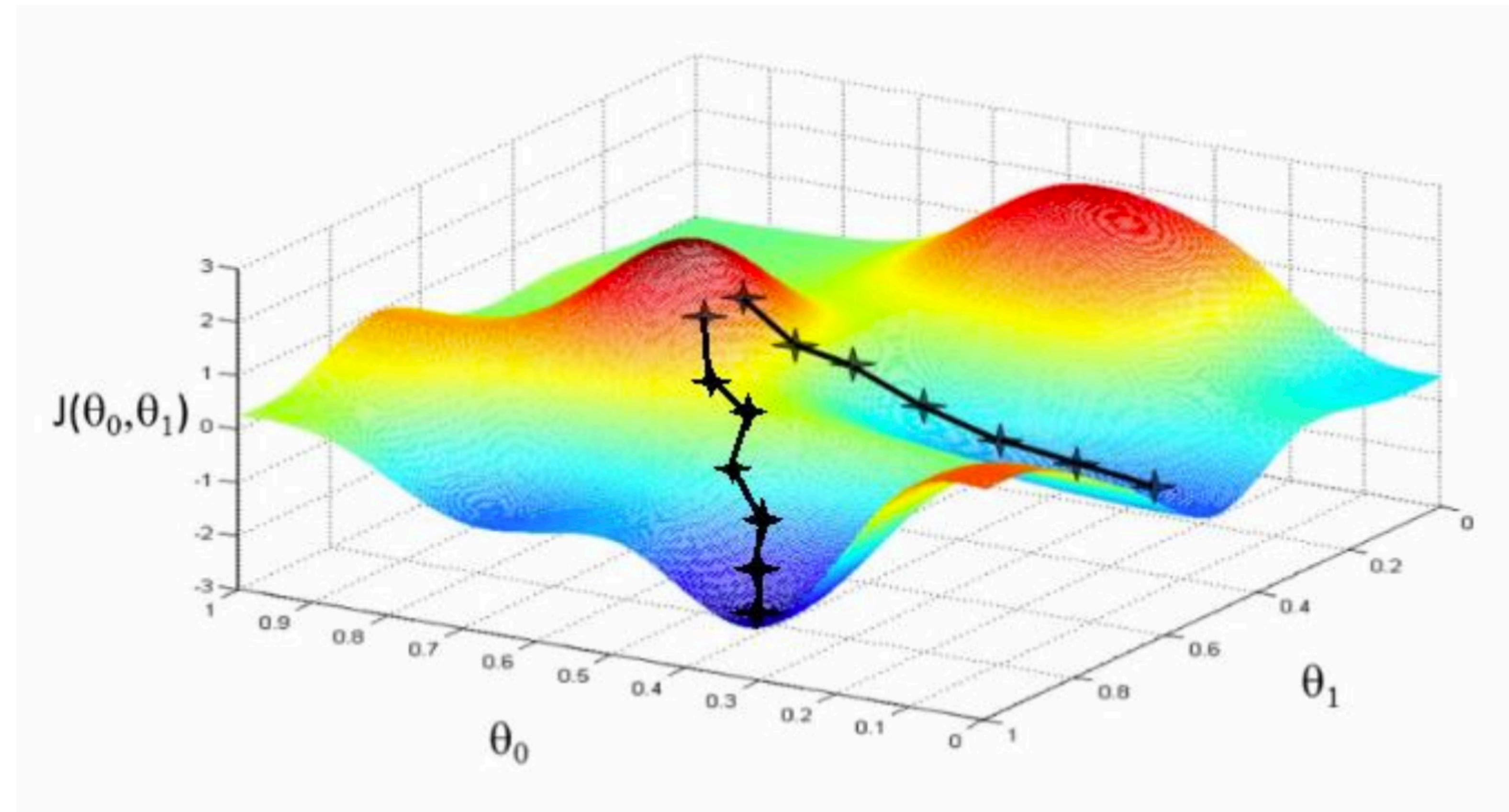
По итогу найдем локальный минимум!



# Локальные минимумы

Начиная с разных точек можно оказаться в разных местах

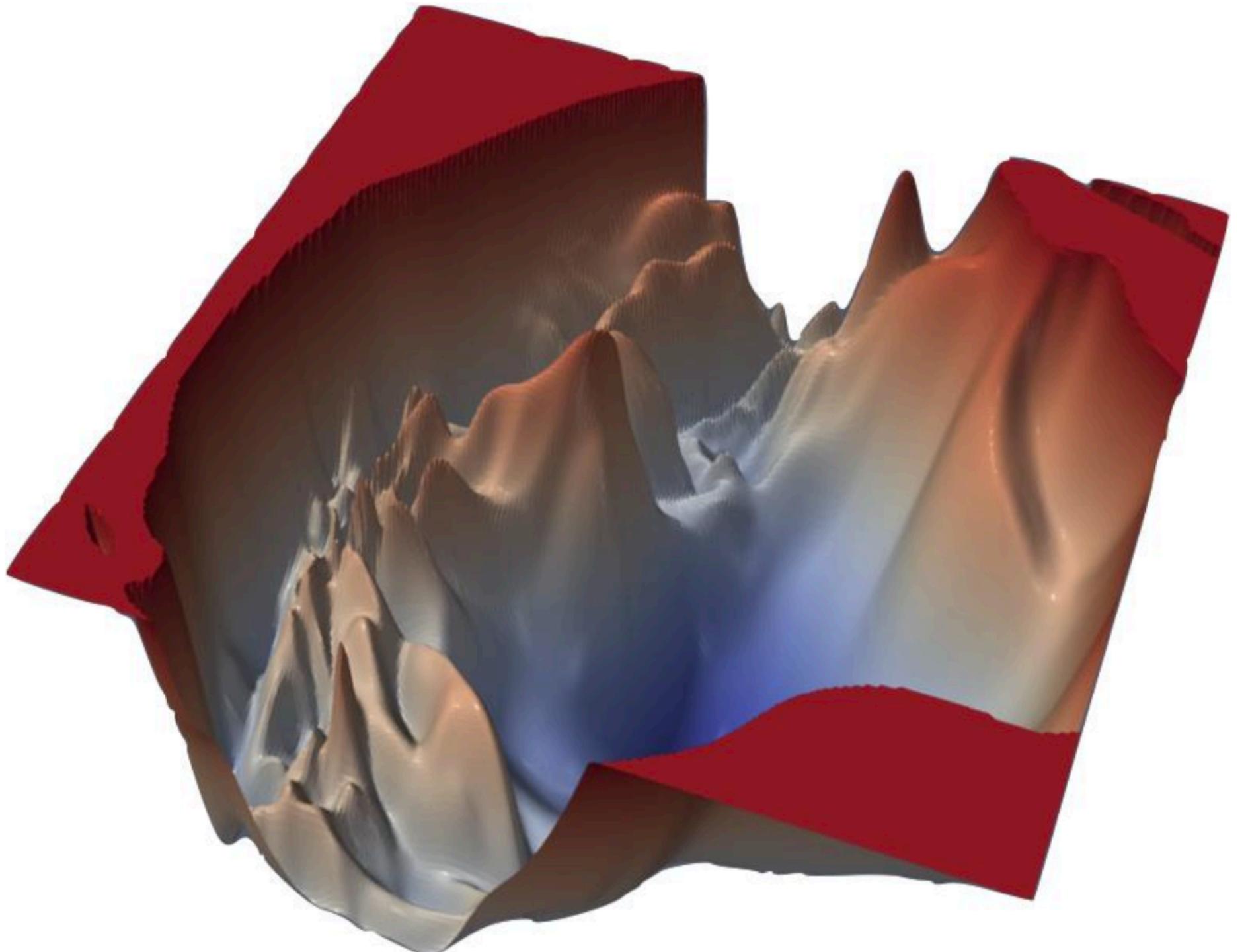
Мультистарт — запускаем обучение несколько раз из разных точек, оставляем лучший



## Локальные минимумы

Поверхность функционала ошибки (Loss Landscape) имеет странную и непонятную форму, необходимо

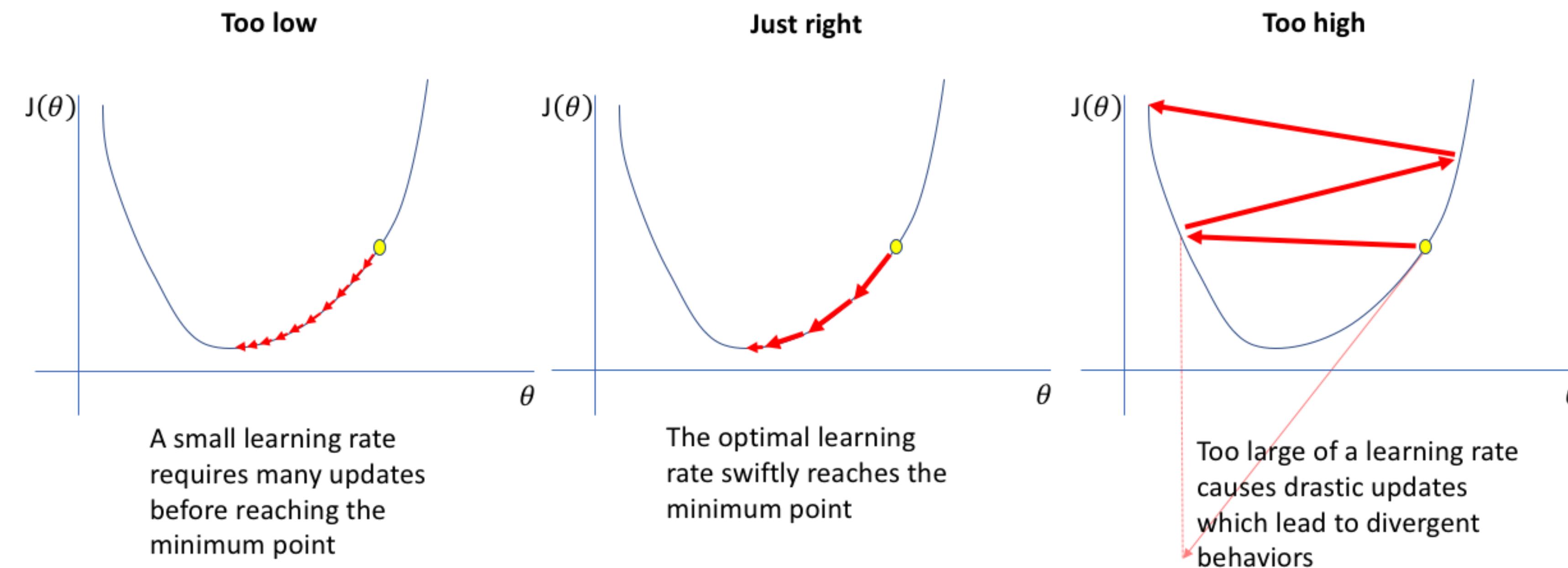
- 👉 Понимать размах значений loss-функции
- 👉 Уметь выбираться из локальных минимумов
- 👉 Адаптировать обучение под данные/функционал ошибки



# Learning rate

$\eta$  — длина шага/learning rate

- 👉 Позволяет контролировать скорость обучения
- 👉 Слишком маленькое значение — “топчемся” на месте, нет сходимости
- 👉 Слишком большое значение — “перешагиваем” минимум, нет сходимости
- 👉 Гиперпараметр, который надо всегда подбирать



# Learning rate

Длину шага можно адаптировать под обучение

- 👉 Начинаем из случайной точки пространства — надо “осмотреться”
- 👉 Чем ближе к минимуму, тем аккуратнее надо делать шаги

Переменная длина шага — меняем  $\eta$  в зависимости от шага

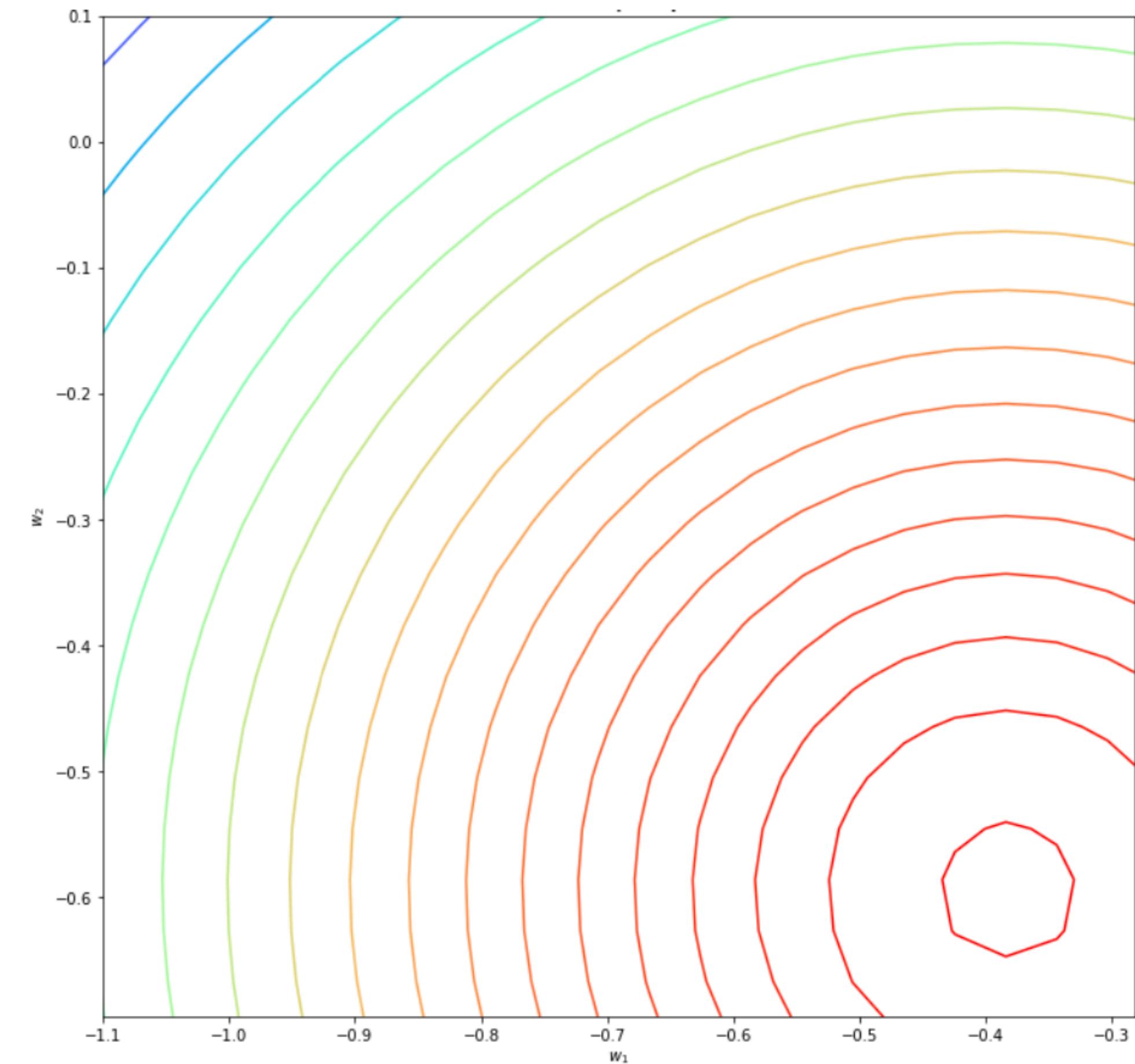
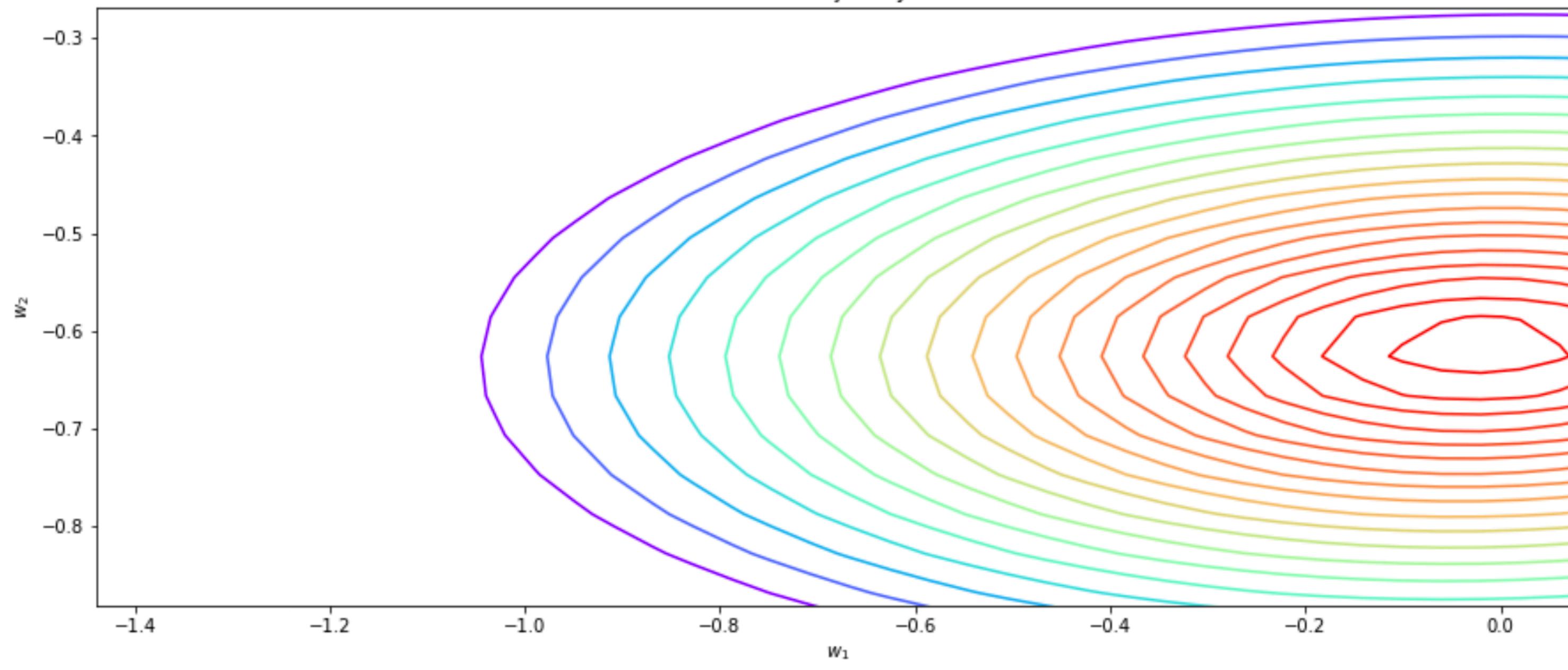
$$1. \eta_t = \frac{1}{t}$$

$$2. \eta_t = \lambda \left( \frac{s}{s+t} \right)^p$$
 — общий случай (из библиотеки [vwarpal wabbit](#))

3. Любая монотонно-убывающая функция

# Масштабирование признаков

Позволяет “упростить” поверхность функционала ошибки



# Сложности градиентного спуска

Для каждого шага необходимо посчитать

$$\frac{1}{l} \sum_{i=1}^l q_i(w)$$

- 👉 Датасет большой — может быть трудоемко
- 👉 В каждой точке не нужен точный градиент —  
делаем небольшой шаг в сторону антиградиента
- 👉 Небольшие неточности не критичны!



# Стochastic градиентный спуск

💡 Оценим градиент одним слагаемым

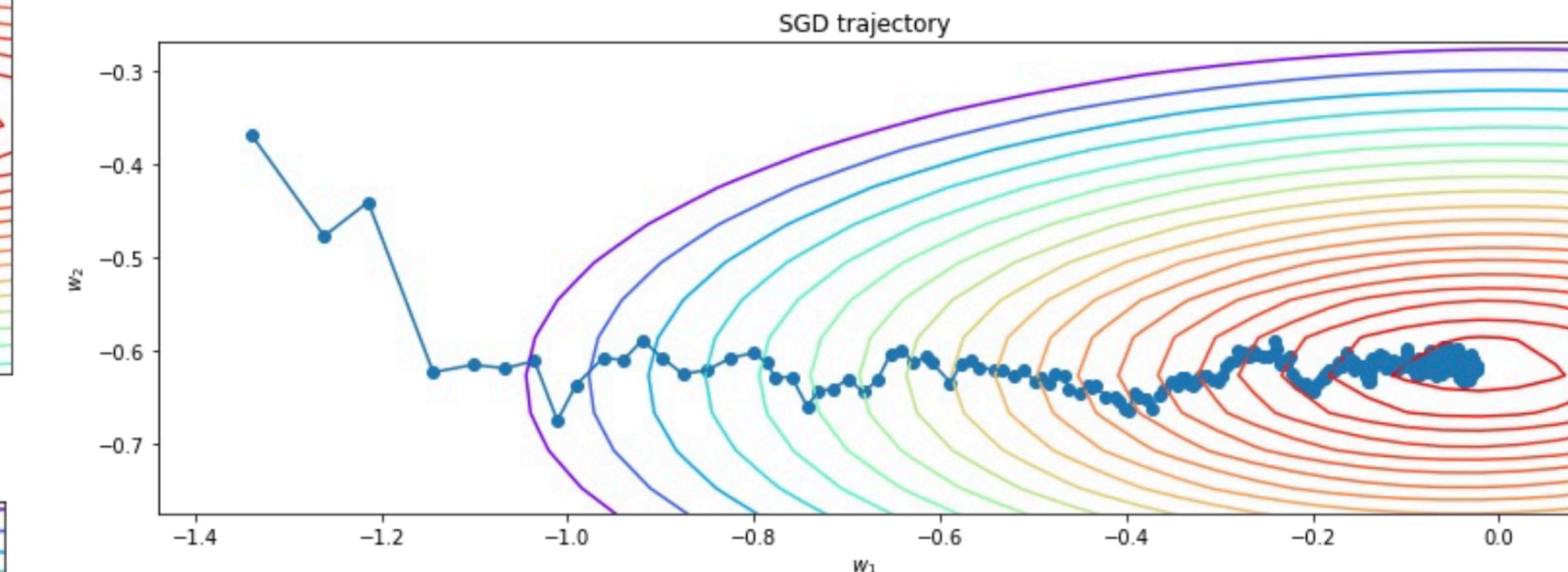
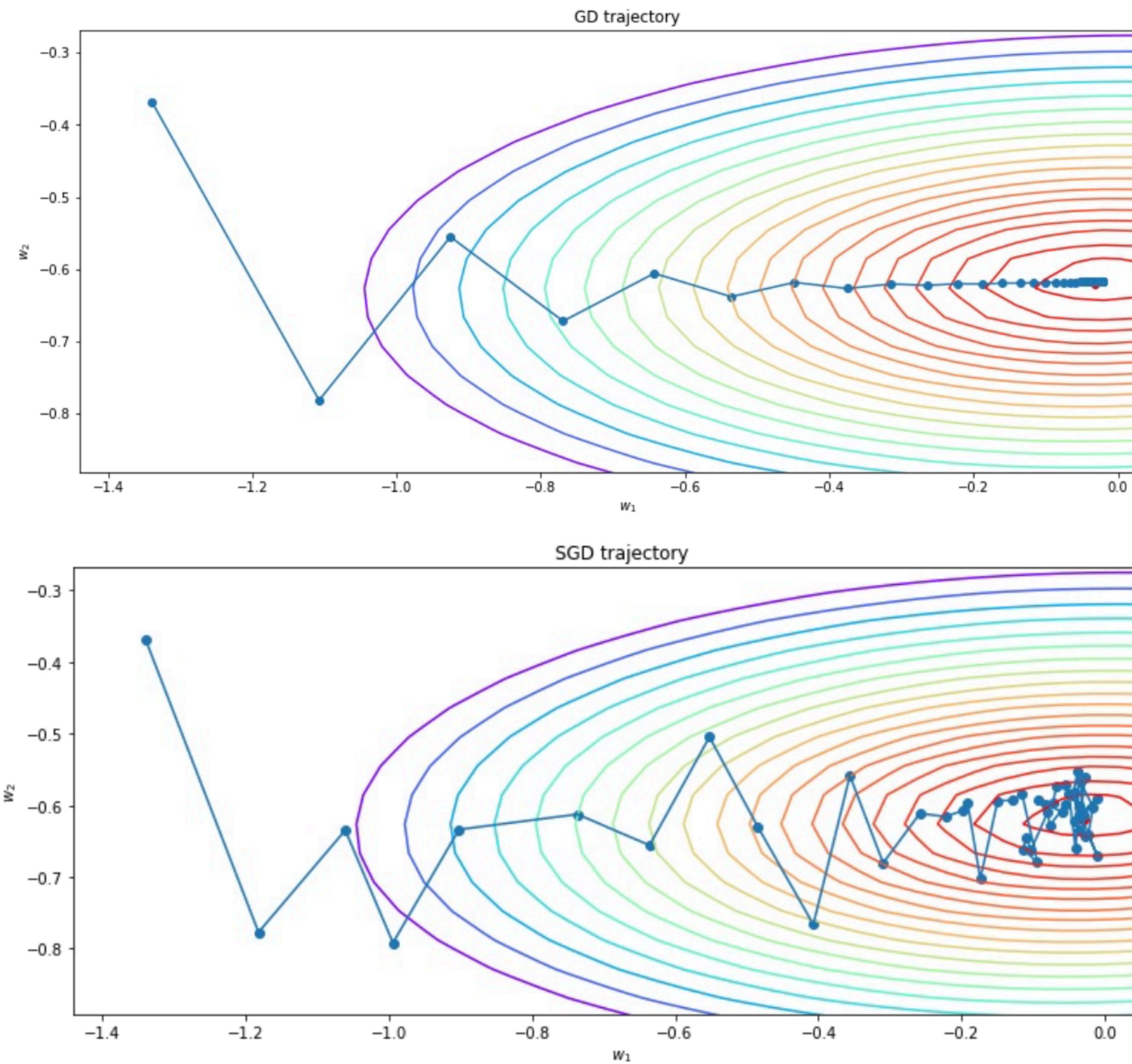
$$\nabla Q(w) \approx \nabla q_i(w)$$

## Stochastic Gradient Descent (SGD):

- 👉 Градиентный спуск — чем ближе к минимуму, тем меньше градиент SGD — это свойство теряется
- 👉 Важно использовать адаптивный LR и уменьшат его по мере обучения
- 👉 **Mini-batch GD** — модификация, оценка по нескольким слагаемым

$$\nabla Q(w) = \frac{1}{k} \sum_{j=1}^k \nabla q_j(w)$$

# SGD и шаг обучения



$$\eta_t = \frac{0.1}{t^{0.3}}$$

# Stochastic Average Gradient (SAG)

В 2013 году был предложен метод среднего стохастического градиента

1. Инициализируются  $w_0$
2. Инициализируются вспомогательные переменные  $z_i^0 - z_i^0 = \nabla q_i(w_0)$
3. На  $k$ -ой итерации выбирается случайное слагаемое  $Q(w^k)$  и обновляется соответствующая вспомогательная переменная  $z_{i_k}^0$
4. Оценка градиента — среднее вспомогательных переменных

В качестве оптимизации можно инициализировать  $z$  нулями и постепенно заполнять

# Модификации градиентного спуска

**Инерции (momentum)** — направлении антиградиента может сильно меняться от шага к шагу, вносит сильный шум в движение  $\Rightarrow$  учтываем антиградиент с нескольких последних шагов,  $h$  — вектор инерции

1.  $h_0 = 0$
2.  $h_k = \alpha h_{k-1} + \eta_k \nabla Q(w_{k-1})$
3.  $w_k = w_{k-1} - h_k$

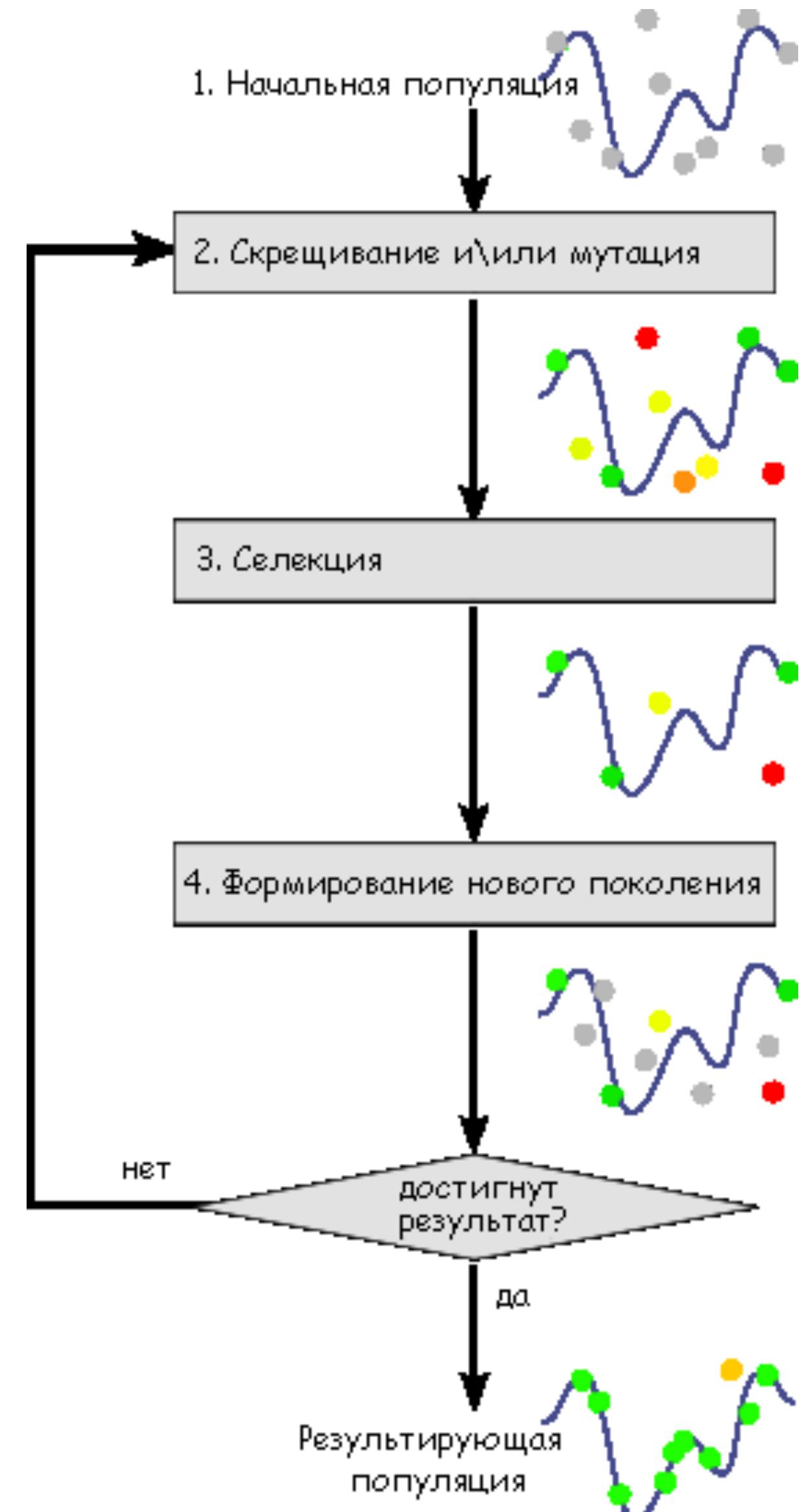
**Адаптивные (adaptive)** — длина шага варьируется в зависимости от размеров градиента по каждому направлению

1.  $G_{kj} = \alpha G_{k-1,j} + (1 - \alpha)(\nabla Q(w_{k-1}))_j^2$
2.  $w_j^k = w_j^{k-1} - \frac{\eta_t}{\sqrt{G_{kj} + \epsilon}} (\nabla Q(w^{k-1}))_j$

# Только ли градиентный спуск?

Существует большое количество оптимизационных алгоритмов...

- 👉 Нулевого порядка — золотое сечение, имитация отжига, генетические алгоритмы
- 👉 Первого порядка — градиентный спуск, метод сопряженных градиентов, квазиньютоновские методы
- 👉 Второго порядка — ньютоновские методы

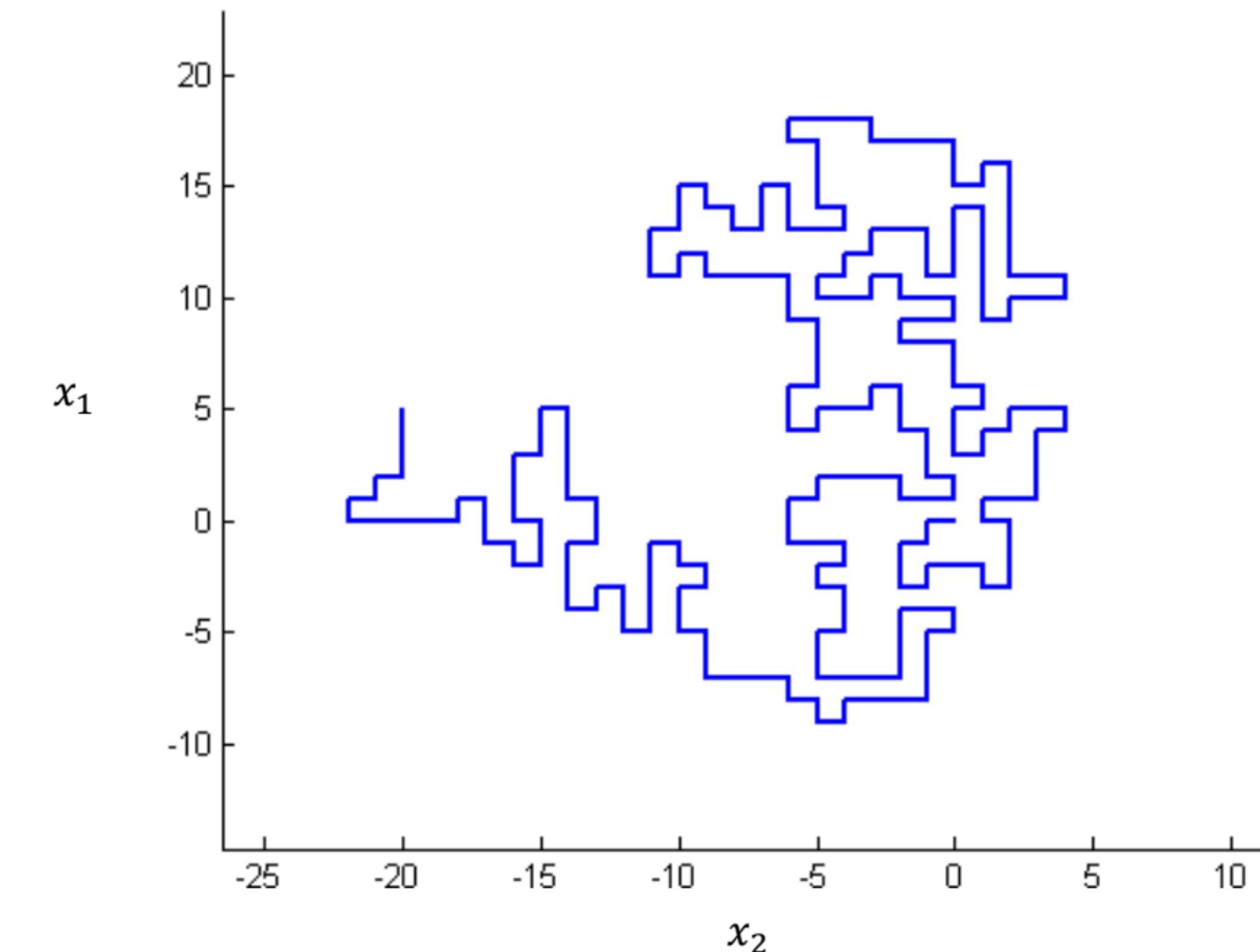
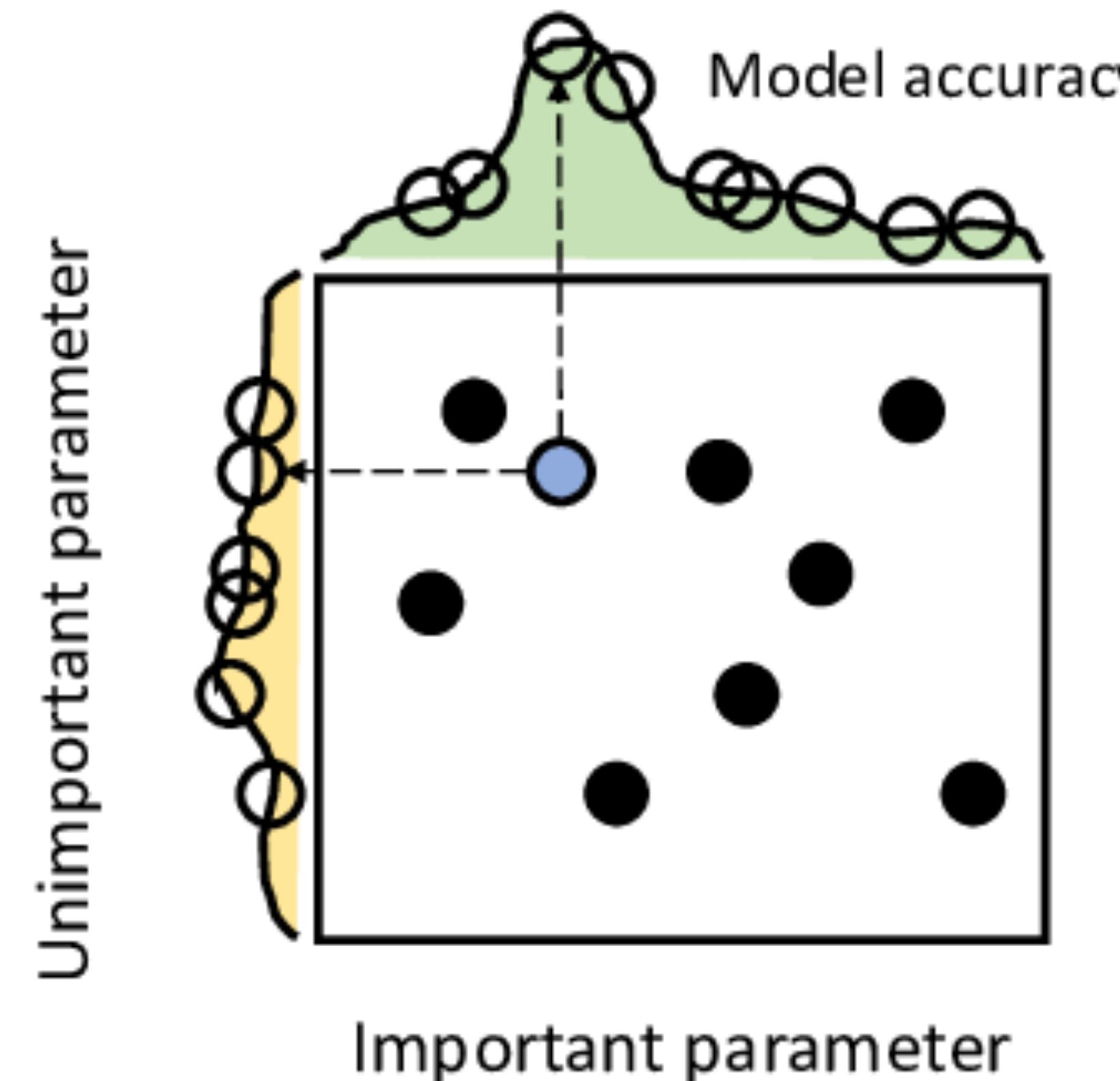


# **Подбор гипер-параметров**

# Random Search

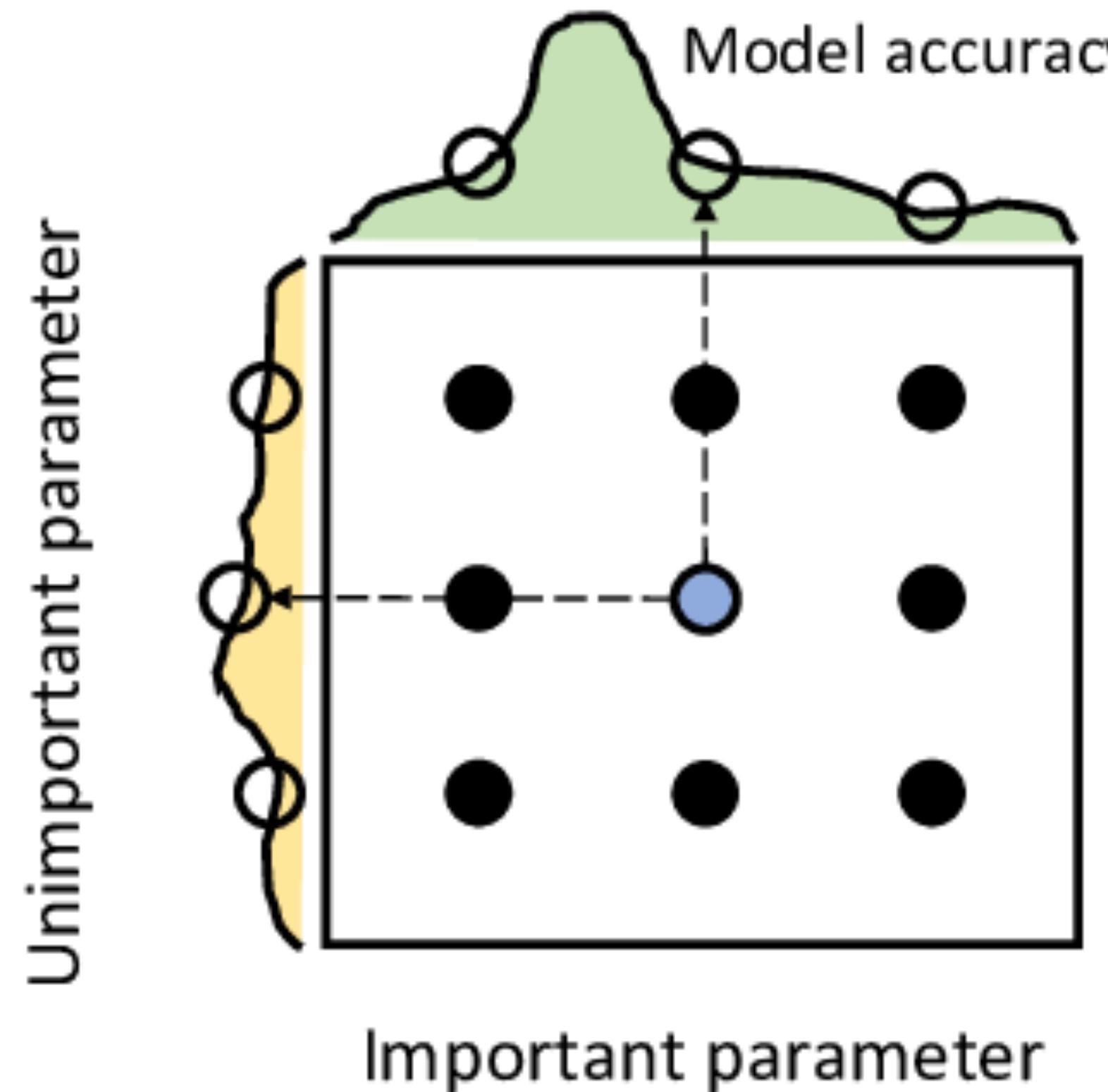
Пока не надоест — выбираем случайные гипер-параметры и запоминаем лучшие

Random Walk — если не можем семплировать независимо от предыдущей точки



# Grid Search

Поиск по сетке — задаем сетку / значения фичей, перебор всевозможных комбинаций

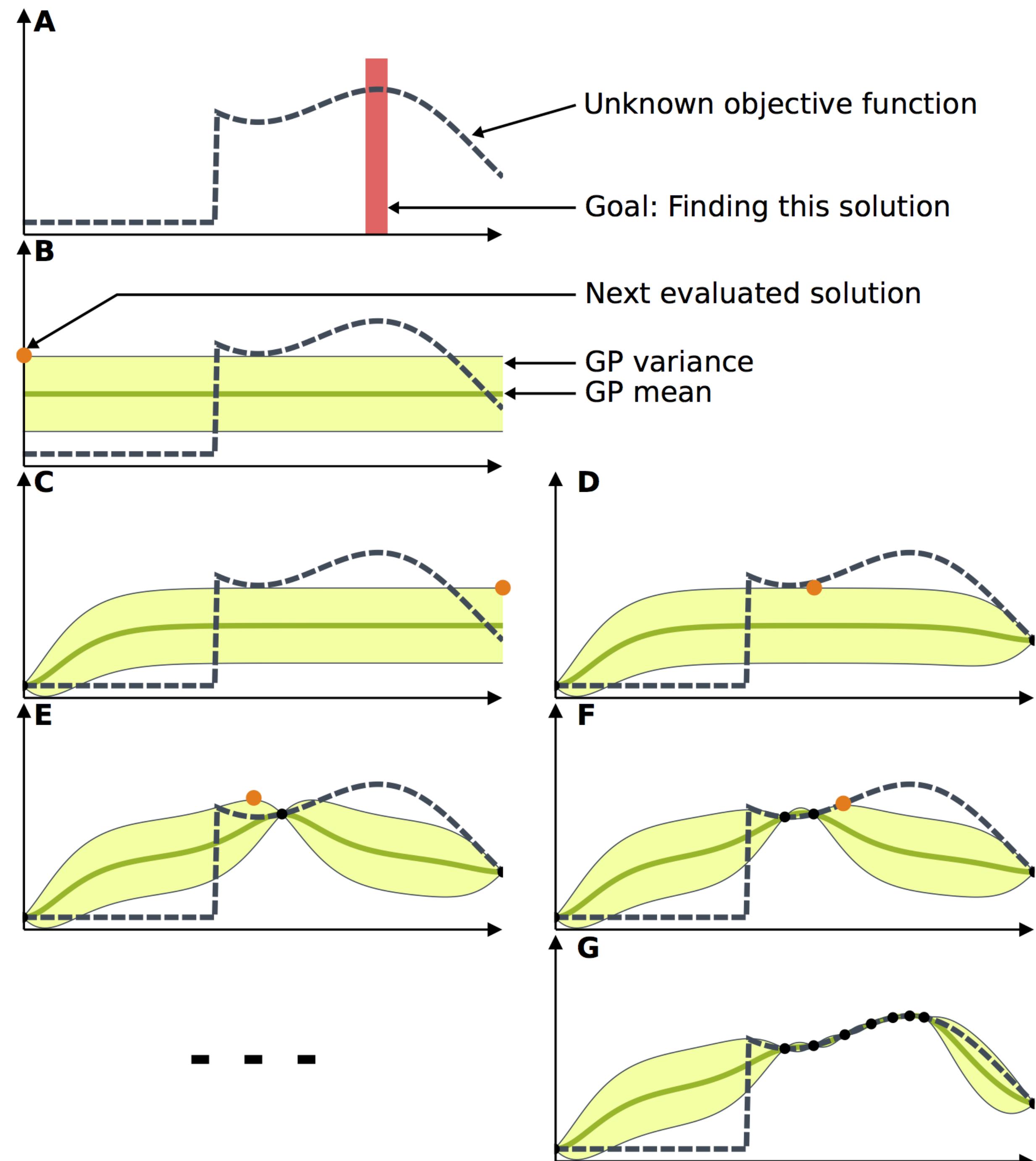


```
>>> from sklearn.model_selection import GridSearchCV
>>> params = {
...     "alpha": [0.01, 0.1, 1, 10, 100],
...     "penalty": ["l2", "l1", None],
...     ...
... }
>>> model = SGDRegressor()
>>> grid_search = GridSearchCV(
...     model, params, n_jobs=-1, cv=3
... )
>>> grid_search.fit(X, y)
```

# Bayesian Optimization

💡 Используем вспомогательную, суррогатную, функцию для аппроксимации целевой функции, метрики

1. Новая точка выбирается с помощью функции-приобретения (acquisition function)  
Баланс между исследованием (exploration) и использованием (exploitation)
2. Запускается обучение в выбранными гипер-параметрами
3. Обновляем суррогатную функцию



# Summary

1. Изучили градиентный спуск — итеративный метод оптимизации первого порядка
  1. Требует вычислений по всему датасету — SGD и SAD позволяют пересчитывать градиент по нескольким точкам
  2. Легко скатывается в локальные минимумы — “хитрый” пересчет размера шага позволяет вылезти из них
2. Обсудили подбор гипер-параметров для моделей
  1. GridSearch — база, когда надо перебрать заданные комбинации гипер-параметров
  2. Bayesian Optimization — для итеративного, но долгого поиска лучших гипер-параметров

Me: \*uses machine learning\*

Machine: \*learns\*

Me:

