Основы машинного обучения

Поляк Марк Дмитриевич

2025

Градиентный спуск

Лекция 4

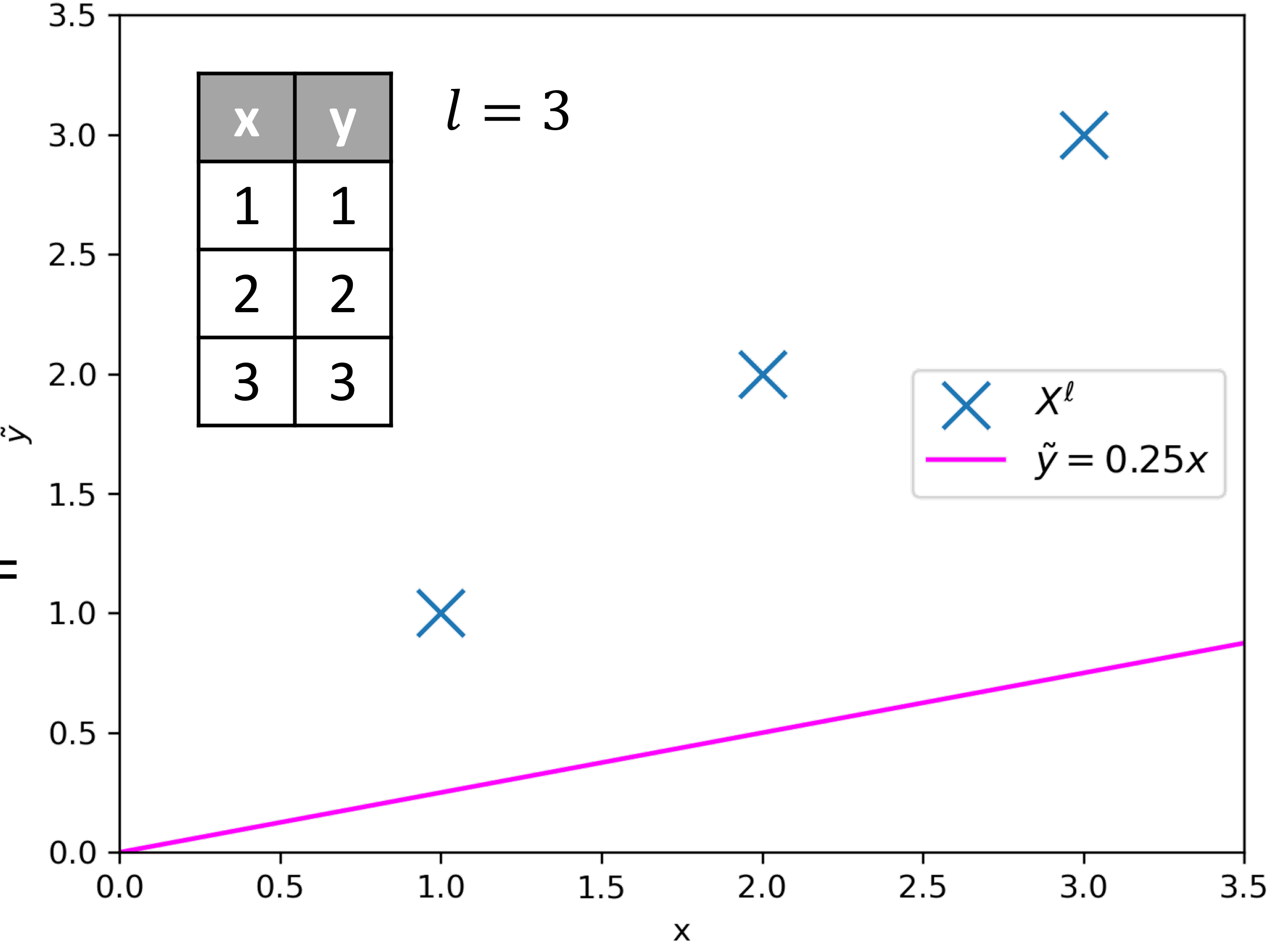
Линейная регрессия с одной переменной

- Модель:

$$g(x, \boldsymbol{\theta}) = \theta_1 x + \theta_0,$$
$$\tilde{y} = kx + b$$

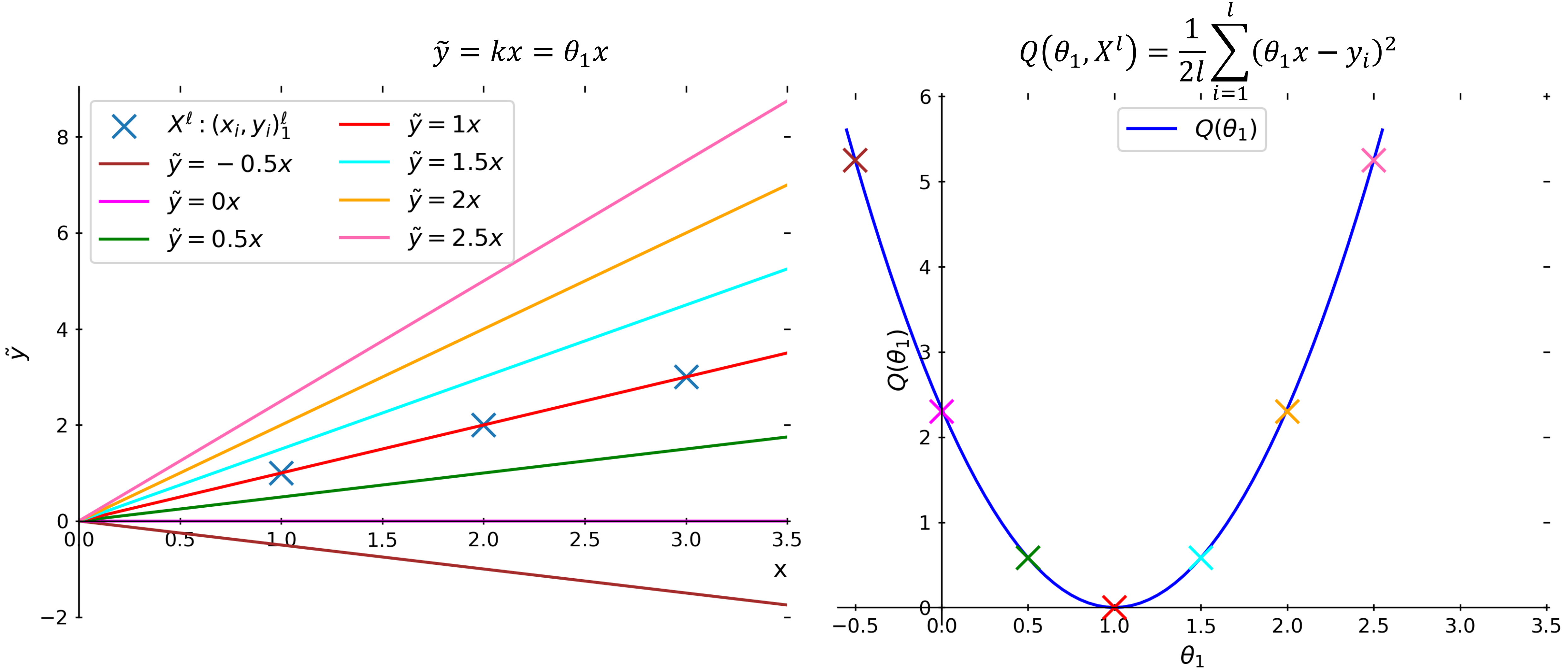
- Пусть $k = \theta_1, b = \theta_0 = 0$
- Тогда эмпирический риск:

$$Q(\boldsymbol{\theta}, X^l) = \frac{1}{2l} \sum_{i=1}^{l} (\tilde{y} - y_i)^2 =$$
$$= \frac{1}{2l} \sum_{i=1}^{l} (kx - y_i)^2$$



x	y
1	1
2	2
3	3

Пример функционала качества (эмпирический риск)



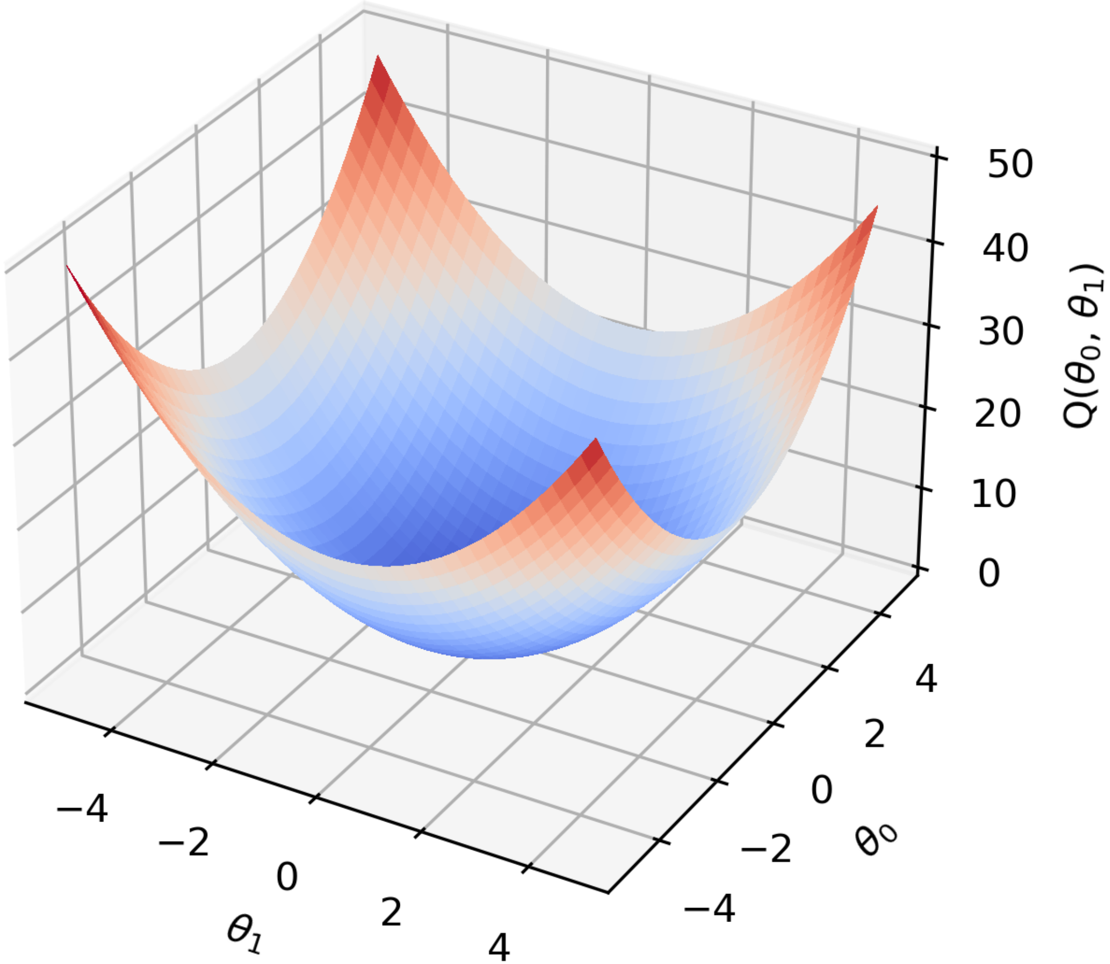
Пример функционала качества (эмпирический риск)

- Модель:

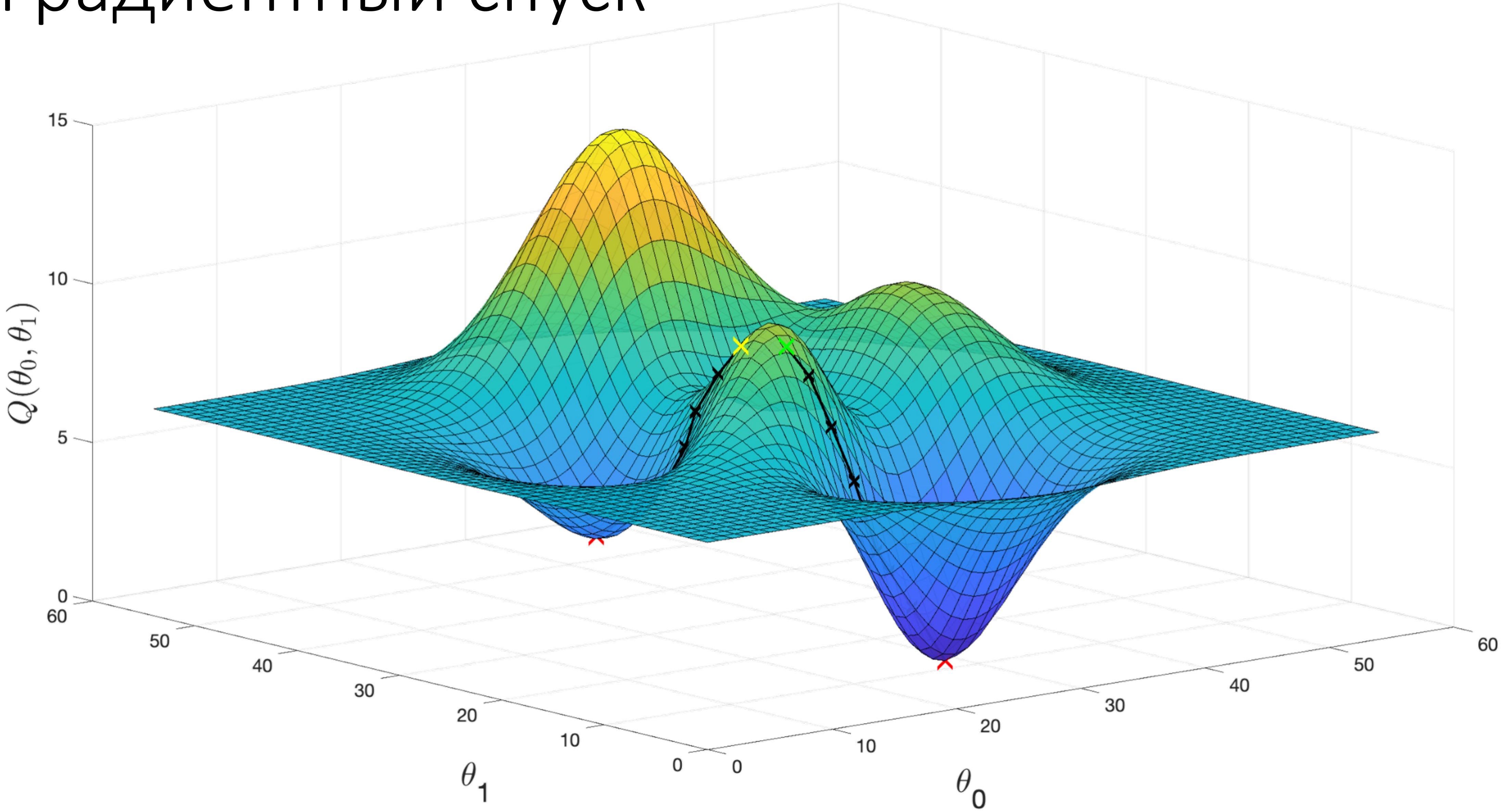
$$g(x, \boldsymbol{\theta}) = \theta_1 x + \theta_0,$$
$$\tilde{y} = kx + b$$

- Пусть $k = \theta_1, b = \theta_0 \neq 0$
- Тогда эмпирический риск:

$$Q(\boldsymbol{\theta}, X^l) = \frac{1}{2l} \sum_{i=1}^{l} (\tilde{y} - y_i)^2 =$$
$$= \frac{1}{2l} \sum_{i=1}^{l} (kx + b - y_i)^2$$



Градиентный спуск



Градиентный спуск

- **Дано**: функционал качества $Q(\boldsymbol{\theta})$.
- **Найти**: вектор параметров $\boldsymbol{\theta}$, при котором $Q(\boldsymbol{\theta}) \to \min$.

  Пример: $\min\limits_{\theta_0, \theta_1} Q(\theta_0, \theta_1)$, $\boldsymbol{\theta} = \{\theta_0, \theta_1\}$

- **Алгоритм**:
  1. Задаем начальное значение для вектора $\boldsymbol{\theta}$ (инициализируем веса)

     Пример: $\theta_0 = \theta_1 = 0$
  2. Пошагово изменяем значения элементов вектора θ, чтобы уменьшить $Q(\boldsymbol{\theta})$, до тех пор, пока не достигнем (окажемся вблизи) минимума.

     Пример:
     $$\theta_0^{\text{след.}} = \theta_0^{\text{пред.}} - \alpha \frac{\partial}{\partial \theta_0} Q\left(\theta_0^{\text{пред.}}, \theta_1^{\text{пред.}}\right),$$
     $$\theta_1^{\text{след.}} = \theta_1^{\text{пред.}} - \alpha \frac{\partial}{\partial \theta_1} Q\left(\theta_0^{\text{пред.}}, \theta_1^{\text{пред.}}\right).$$

Алгоритм градиентного спуска

Повторять:

$$\theta_j^{\text{след.}} := \theta_j^{\text{пред.}} - \alpha \frac{\partial}{\partial \theta_j} Q\left(\theta_0^{\text{пред.}}, \theta_1^{\text{пред.}}, \dots, \theta_n^{\text{пред.}}\right)$$

… до тех пор, пока не выполнится $\left|\theta_j^{\text{след.}} - \theta_j^{\text{пред.}}\right| < \delta_\theta$ для всех j,

либо $\left|Q\left(\theta_0^{\text{пред.}}, \dots, \theta_n^{\text{пред.}}\right) - Q\left(\theta_0^{\text{след.}}, \dots, \theta_n^{\text{след.}}\right)\right| < \delta_Q$.

Важно: обновлять θ_j необходимо одновременно для всех j

α – скорость обучения (learning rate):

- Если α слишком мало, требуется большое количество итераций для сходимости
- Если α слишком велико, значение функции потерь может не уменьшаться на каждой итерации и алгоритм может не сойтись к устойчивому минимуму

Градиентный спуск и линейная регрессия с одной переменной

- Найдем частные производные функционала качества:

$$\frac{\partial}{\partial\theta_0}Q(\theta_0,\theta_1) = \frac{\partial}{\partial\theta_0}\frac{1}{2l}\sum_{i=1}^{l}(g(x_i,\theta_0,\theta_1) - y_i)^2 = \frac{1}{2l}\sum_{i=1}^{l}\frac{\partial}{\partial\theta_0}(\theta_1 x_i + \theta_0 - y_i)^2 =$$

$$= \frac{1}{l}\sum_{i=1}^{l}(\theta_1 x_i + \theta_0 - y_i)$$

$$\frac{\partial}{\partial\theta_1}Q(\theta_0,\theta_1) = \frac{\partial}{\partial\theta_1}\frac{1}{2l}\sum_{i=1}^{l}(g(x_i,\theta_0,\theta_1) - y_i)^2 = \frac{1}{l}\sum_{i=1}^{l}(\theta_1 x_i + \theta_0 - y_i)\cdot x_i$$

- Итоговые формулы для градиентного спуска:

$$\theta_0 \coloneqq \theta_0 - \alpha\frac{\partial}{\partial\theta_0}Q(\theta_0,\theta_1) = \theta_0 - \alpha\frac{1}{l}\sum_{i=1}^{l}(g(x_i,\theta_0,\theta_1) - y_i)$$

$$\theta_1 \coloneqq \theta_1 - \alpha\frac{\partial}{\partial\theta_1}Q(\theta_0,\theta_1) = \theta_1 - \alpha\frac{1}{l}\sum_{i=1}^{l}(g(x_i,\theta_0,\theta_1) - y_i)\,x_i$$

Градиентный спуск и многомерная линейная регрессия

Повторять, пока алгоритм не сойдется: {

$$\theta_0 \coloneqq \theta_0 - \alpha\frac{1}{l}\sum_{i=1}^{l}(g(\mathbf{x}_i, \boldsymbol{\theta}) - y_i)\cdot x_{i,0}$$

$$\theta_1 \coloneqq \theta_1 - \alpha\frac{1}{l}\sum_{i=1}^{l}(g(\mathbf{x}_i, \boldsymbol{\theta}) - y_i)\cdot x_{i,1}$$

$$\theta_2 \coloneqq \theta_2 - \alpha\frac{1}{l}\sum_{i=1}^{l}(g(\mathbf{x}_i, \boldsymbol{\theta}) - y_i)\cdot x_{i,2}$$

…

}

$\mathbf{x}_i$ – i-ый объект (вектор) из обучающей выборки, $i = 1..l$

$x_{i,j}$ – j-ый элемент вектора $\mathbf{x}_i$, $j = 0..n$, $x_{i,0} = 1$ при $\forall i$

$\boldsymbol{\theta}$ – вектор параметров, $\boldsymbol{\theta}=\{\theta_0, \theta_1, \dots, \theta_n\}$

Повторять, пока алгоритм не сойдется: {

$$\theta_j \coloneqq \theta_j - \alpha\frac{1}{l}\sum_{i=1}^{l}(g(\mathbf{x}_i, \boldsymbol{\theta}) - y_i)\cdot x_{i,j}\text{, для } j = 0..n$$

}

Стохастический градиентный спуск

- **Проблема**: если обучающая выборка X^ℓ велика ($\ell \gg 0$), то каждый шаг градиентного спуска будет требовать большого количества вычислений, а сам алгоритм будет работать медленно
- **Решение**: Stochastic Gradient Descent, SGD
  - берем по одной паре «объект-ответ» (x_i, y_i) и сразу обновляем вектор $\boldsymbol{\theta}$
  - градиент вычисляем по функции ошибки $\mathcal{L}$, а не по функционалу качества Q
  - функционал качества оцениваем по приближенной формуле
- **Алгоритм**:
  1. выбрать объект x_i из X^ℓ случайным образом;
  2. вычислить потерю: $\varepsilon_i = \mathcal{L}_i(g, x)$; (напр., $\varepsilon_i = (g(x_i, \boldsymbol{\theta}) - y_i)^2$)
  3. сделать градиентный шаг: $\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathcal{L}_i(g, x)$;
  4. оценить функционал: $\bar{Q} = \lambda \varepsilon_i + (1 - \lambda)\bar{Q}$, где λ – скорость забывания;
  5. повторять шаги 1-4, пока значение $\bar{Q}$ и/или параметры $\boldsymbol{\theta}$ не сойдутся.

Рекуррентная оценка функционала качества

- **Проблема**: вычисление оценки Q по всей выборке $x_1, \dots, x_\ell$ намного дольше градиентного шага по одному объекту x_i.
- **Решение**: использовать приближенную рекуррентную формулу.

Среднее арифметическое:

$$\bar{Q}_m = \frac{1}{m}\,\varepsilon_m + \frac{1}{m}\,\varepsilon_{m-1} + \frac{1}{m}\,\varepsilon_{m-2} + \cdots$$
$$\bar{Q}_m = \frac{1}{m}\,\varepsilon_m + \left(1 - \frac{1}{m}\right)\bar{Q}_{m-1}$$

Экспоненциальное скользящее среднее:

$$\bar{Q}_m = \lambda\varepsilon_m + (1-\lambda)\lambda\varepsilon_{m-1} + (1-\lambda)^2\lambda\,\varepsilon_{m-2} + \cdots$$
$$\bar{Q}_m = \lambda\varepsilon_m + (1-\lambda)\bar{Q}_{m-1}$$

Параметр λ – скорость забывания предыстории ряда.

Варианты инициализации весов

1. $\theta_j = 0$ для всех $j = 0, \dots, n$.
2. Небольшие случайные значения:
$$\theta_j = \mathrm{random}\left(-\frac{1}{2n}, \frac{1}{2n}\right)$$
3. $\theta_j = \frac{\langle y, f_j \rangle}{\langle f_j, f_j \rangle}, f_j = \left(f_j(x_i)\right)_{i=1}^{\ell}$ – вектор значений признака.

   Эта оценка θ оптимальна, если:

   1) функция потерь квадратична и
   2) признаки некоррелированы, $\langle f_j, f_k \rangle = 0,\ j \neq k$.
4. Обучение по небольшой случайной подвыборке объектов.
5. Мультистарт: многократные запуски из разных случайных начальных приближений и выбор лучшего решения.

Варианты выбора скорости обучения

1. Постоянное значение $\alpha = const$.
2. Убывающее значение. Сходимость гарантируется (для выпуклых функций) при
$$\alpha_t \to 0, \qquad \sum_{t=1}^{\infty} \alpha_t = \infty,\ \sum_{t=1}^{\infty} \alpha_t^2 < \infty\,,$$
в частности можно положить $\alpha_t = 1/t$, где t – номер шага.
3. Метод наискорейшего градиентного спуска:
$$\mathcal{L}_i\big(\boldsymbol{\theta} - \alpha \nabla \mathcal{L}_i(\boldsymbol{\theta})\big) \to \min_{\alpha}$$
позволяет найти адаптивную скорость α^*.
При квадратичной функции потерь $\alpha^* = \|x_i\|^{-2}$.
4. Пробные случайные шаги для «выбивания» итерационного процесса из локальных минимумов.
5. Метод Левенберга-Марквардта (второго порядка).

Диагональный метод Левенберга-Марквардта

- Метод Ньютона-Рафсона, $\mathcal{L}_i(\boldsymbol{\theta}) \equiv \mathcal{L}(\langle\boldsymbol{\theta}, x_i\rangle y_i)$:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha\left(\mathcal{L}_i''(\boldsymbol{\theta})\right)^{-1}\nabla\mathcal{L}_i(\boldsymbol{\theta}),$$

где $\mathcal{L}_i''(\boldsymbol{\theta}) = \left(\frac{\partial^2\mathcal{L}_i(\boldsymbol{\theta})}{\partial\theta_j\partial\theta_{j'}}\right)$ – гессиан, матрица $n{\times}n$.

- Эвристика. Считаем, что гессиан диагонален

$$\theta_j = \theta_j - \alpha\left(\frac{\partial^2\mathcal{L}_i(\boldsymbol{\theta})}{\partial\theta_j^2} + \mu\right)^{-1}\frac{\partial\mathcal{L}_i(\boldsymbol{\theta})}{\partial\theta_j},$$

  α – скорость обучения, можно полагать $\alpha = 1$;
  μ – параметр, предотвращающий обнуление знаменателя.
  Отношение α/μ есть скорость обучения на ровных участках функционала $\mathcal{L}_i(\boldsymbol{\theta})$, где вторая производная обнуляется.

Проблема переобучения

- Возможные причины переобучения:
  - слишком мало объектов;
  - слишком много признаков;
  - линейная зависимость (мультиколлинеарность) признаков.
- Проявления переобучения:
  - слишком большие значения параметров $|\theta_j|$ разных знаков;
  - неустойчивость дискриминантной функции $\langle \boldsymbol{\theta}, x \rangle$;
  - $Q(X^\ell) \ll Q(X^k)$.
- Основной способ уменьшить переобучение:
  - **регуляризация** – уменьшение значений параметров (сокращение весов, weight decay).

Регуляризация (сокращение значений параметров)

- Штраф за увеличение нормы вектора весов:

$$\tilde{\mathcal{L}}_i(\boldsymbol{\theta}) = \mathcal{L}_i(\boldsymbol{\theta}) + \frac{\tau}{2}\|\boldsymbol{\theta}\|^2 = \mathcal{L}_i(\boldsymbol{\theta}) + \frac{\tau}{2}\sum_{j=1}^{n}\theta_j^2 \to \min_{\boldsymbol{\theta}}$$

- Градиент:

$$\nabla\tilde{\mathcal{L}}_i(\boldsymbol{\theta}) = \nabla\mathcal{L}_i(\boldsymbol{\theta}) + \tau\boldsymbol{\theta}.$$

- Модификация градиентного шага:

$$\boldsymbol{\theta} = \boldsymbol{\theta}(1 - \alpha\tau) - \alpha\nabla\mathcal{L}_i(\boldsymbol{\theta}).$$

- Методы подбора коэффициента регуляризации τ:
    1. скользящий контроль;
    2. стохастическая адаптация;
    3. двухуровневый байесовский вывод.

Достоинства и недостатки стохастического градиента

- **Достоинства:**
  1. легко реализуется;
  2. легко обобщается на любые $g(\mathbf{x}, \boldsymbol{\theta}), \mathcal{L}(\mathrm{a}, y)$;
  3. легко добавить регуляризацию;
  4. возможно динамическое (потоковое) обучение;
  5. на сверхбольших выборках можно получить неплохое решение, даже не обработав все (x_i, y_i);
  6. подходит для задач с большими данными.
- **Недостатки:**
  1. подбор комплекса эвристик является искусством
     (не забыть про переобучение, застревание, расходимость)

Векторизация

NumPy

LAPACK

Linear Algebra Package

BLAS

Basic Linear Algebra Subprograms

- Регрессионная модель:

$$g(\mathbf{x}_i, \boldsymbol{\theta}) = \theta_0 + \sum_{j=1}^{n} \theta_j x_{i,j} = \sum_{j=0}^{n} \theta_j x_{i,j}$$

где $\mathbf{x}_i$ – вектор признаков i-го объекта, $\boldsymbol{\theta}$ – вектор параметров, $x_{i,0} = 1$ – фиктивный признак.

```
for j in range(0, n):
    g = g + theta[j] * x[j]
```

- В векторном виде:

$$g(\mathbf{x}_i, \boldsymbol{\theta}) = \boldsymbol{\theta} \cdot \mathbf{x}_i$$

быстрее на порядки!

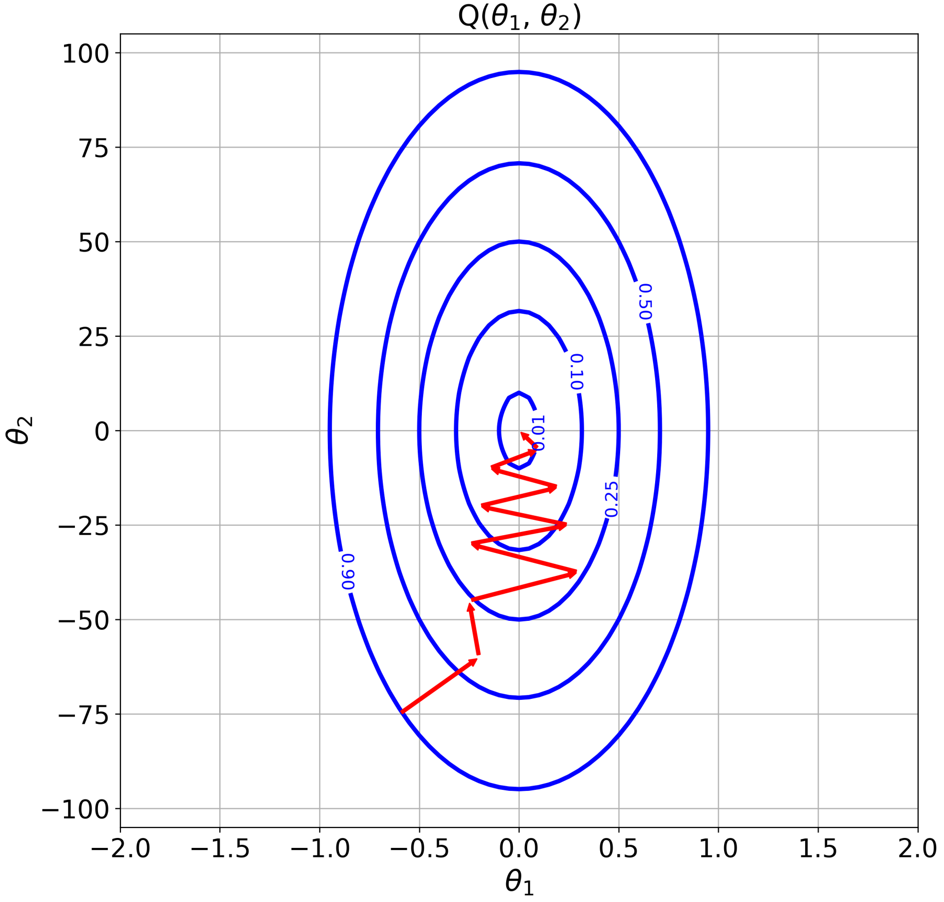
```
import numpy as np
g = np.dot(theta, x)
```

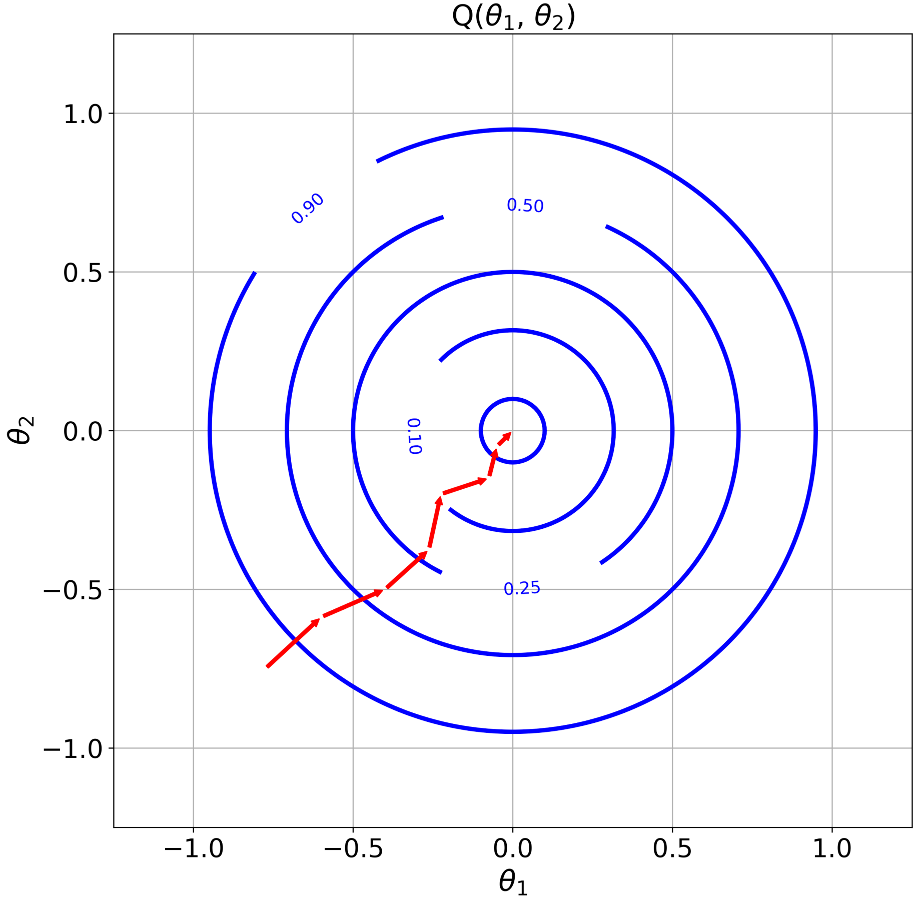
theta[0]	theta[1]	…	theta[n]
*	*	…	*
x[0]	x[1]	…	x[n]

theta[0]*x[0] + theta[1]*x[1] … + … theta[n]*x[n]

Нормализация данных

$$\tilde{y} = \theta_1 x_1 + \theta_2 x_2 + \theta_0$$





Нормализация данных

- Нормирование значений признаков (нормализация средним):
    - вычесть математическое ожидание μ_{x_j} признака x_j;
    - поделить на СКО σ_{x_j};

$$x_j^* := \frac{x_j - \mu_{x_j}}{\sigma_{x_j}}.$$

- Минимаксная нормализация:
    - значения признака приводятся к диапазону $[0,1]$ или $[-1,1]$, например:

$$x_j^* := \frac{x_j - \min x_j}{\max x_j - \min x_j}.$$

    - вместо $\min x_j$ **в числителе** можно использовать среднее μ_{x_j} или медиану.

Когда нужна нормализация данных

- Необходимо стремиться: $-1 \le x_j \le 1$ для каждого признака x_j
- Эвристика: нормализация не обязательна, если диапазон признака отличается от $-1 \le x_j \le 1$ менее, чем на 2 порядка
- Нормализация не нужна:

$$-3 \le x_1 \le 3$$
$$-0.3 \le x_2 \le 0.3$$
$$0 \le x_3 \le 3$$
$$-2 \le x_4 \le 0.5$$

Нормализация обязательна:

$$-100 \le x_5 \le 100$$
$$-0.001 \le x_6 \le 0.001$$
$$98.6 \le x_7 \le 105$$

- Лучше провести нормализацию, чем от нее отказаться