

Занятие 2

Линейные методы регрессии. Часть 1.

Елена Кантонистова

elena.kantonistova@yandex.ru

ВШЭ, 2020

ЛИНЕЙНАЯ РЕГРЕССИЯ

Пример (напоминание):

Предположим, что мы хотим предсказать *стоимость дома* у по его *площади* (x_1) и *количеству комнат* (x_2).

Линейная модель для предсказания стоимости:

$$a(x) = w_0 + w_1x_1 + w_2x_2,$$

где w_0, w_1, w_2 -

параметры модели (веса).



ЛИНЕЙНАЯ РЕГРЕССИЯ

Пример (напоминание):

Предположим, что мы хотим предсказать *стоимость дома* y по его *площади* (x_1) и *количеству комнат* (x_2).

Линейная модель для предсказания стоимости:

$$a(x) = w_0 + w_1x_1 + w_2x_2,$$

где w_0, w_1, w_2 -

параметры модели (*веса*).



Общий вид (линейная регрессия):

$$a(x) = w_0 + w_1x^1 + w_2x^2 + \dots + w_nx^n,$$

где x^1, \dots, x^d - признаки объекта x .

ЛИНЕЙНАЯ РЕГРЕССИЯ

Линейная регрессия:

$$a(x) = w_0 + w_1x^1 + w_2x^2 + \dots + w_nx^n$$

ЛИНЕЙНАЯ РЕГРЕССИЯ

Линейная регрессия:

$$a(x) = w_0 + w_1x^1 + w_2x^2 + \dots + w_nx^n$$

- сокращенная запись:

$$a(x) = w_0 + \sum_{j=1}^d w_j x^j$$

ЛИНЕЙНАЯ РЕГРЕССИЯ

Линейная регрессия:

$$a(x) = w_0 + w_1x^1 + w_2x^2 + \dots + w_nx^n$$

- сокращенная запись:

$$a(x) = w_0 + \sum_{j=1}^d w_j x^j$$

- запись через скалярное произведение (с добавлением признака $x^0 = 1$):

$$a(x) = w_0 \cdot 1 + \sum_{j=1}^d w_j x^j = \sum_{j=0}^d w_j x^j = (w, x)$$

ЛИНЕЙНАЯ РЕГРЕССИЯ

Линейная регрессия:

$$a(x) = w_0 + w_1x^1 + w_2x^2 + \dots + w_nx^n$$

- сокращенная запись:

$$a(x) = w_0 + \sum_{j=1}^d w_j x^j$$

- запись через скалярное произведение (с добавлением признака $x^0 = 1$):

$$a(x) = w_0 \cdot 1 + \sum_{j=1}^d w_j x^j = \sum_{j=0}^d w_j x^j = (w, x) \leftrightarrow a(x) = (w, x)$$

ЛИНЕЙНАЯ РЕГРЕССИЯ

Линейная регрессия:

$$a(x) = w_0 + \sum_{j=1}^d w_j x^j = (w, x)$$

**Обучение линейной регрессии - минимизация
среднеквадратичной ошибки:**

$$Q(a, X) = \frac{1}{l} \sum_{i=1}^l (a(x_i) - y_i)^2 = \frac{1}{l} \sum_{i=1}^l ((w, x_i) - y_i)^2 \rightarrow \min_w$$

АНАЛИТИЧЕСКОЕ РЕШЕНИЕ ЗАДАЧИ МНК

Задачу обучения линейной регрессии можно записать в матричной форме:

$$Q(a, X) = \frac{1}{l} \|Xw - y\|^2 \rightarrow \min_w$$

АНАЛИТИЧЕСКОЕ РЕШЕНИЕ ЗАДАЧИ МНК

Задачу обучения линейной регрессии можно записать в матричной форме:

$$Q(w, X) = \frac{1}{l} \|Xw - y\|^2 \rightarrow \min_w$$

Существует точное (аналитическое) решение этой задачи:

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

НЕДОСТАТКИ АНАЛИТИЧЕСКОЙ ФОРМУЛЫ

- Обращение матрицы – сложная операция ($O(N^3)$ от числа признаков)
- Матрица $X^T X$ может быть вырожденной или плохо обусловленной
- Если заменить среднеквадратичный функционал ошибки на другой, то скорее всего не найдем аналитическое решение

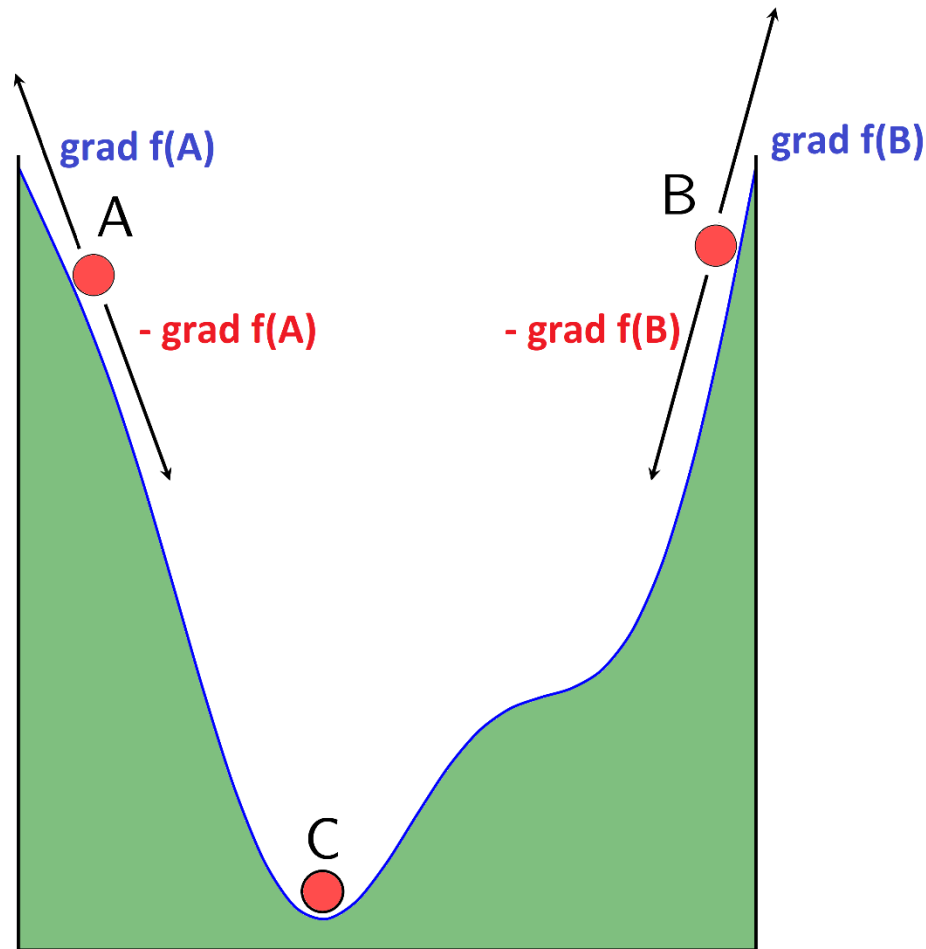
ТЕОРЕМА О ГРАДИЕНТЕ

Теорема. Градиент – это вектор, в направлении которого функция быстрее всего растёт.

Антиградиент (вектор, противоположный градиенту) – вектор, в направлении которого функция быстрее всего убывает.

ТЕОРЕМА О ГРАДИЕНТЕ

Антиградиент (вектор, противоположный градиенту) – вектор, в направлении которого функция быстрее всего убывает.



МЕТОД ГРАДИЕНТНОГО СПУСКА

- Наша задача при обучении модели – найти такие веса w , на которых достигается **минимум функции ошибки**.

МЕТОД ГРАДИЕНТНОГО СПУСКА

- Наша задача при обучении модели – найти такие веса w , на которых достигается минимум функции ошибки.
- Грубо говоря, график среднеквадратичной ошибки – это парабола.

МЕТОД ГРАДИЕНТНОГО СПУСКА

- Наша задача при обучении модели – найти такие веса w , на которых достигается минимум функции ошибки.
- Грубо говоря, график среднеквадратичной ошибки – это парабола.
- Идея метода градиентного спуска:

На каждом шаге (на каждой итерации метода) движемся в сторону антиградиента функции потерь!

То есть на каждом шаге движемся в направлении уменьшения ошибки.

МЕТОД ГРАДИЕНТНОГО СПУСКА

- Наша задача при обучении модели – найти такие веса w , на которых достигается минимум функции ошибки.
- Грубо говоря, график среднеквадратичной ошибки – это парабола.
- **Идея метода градиентного спуска:**

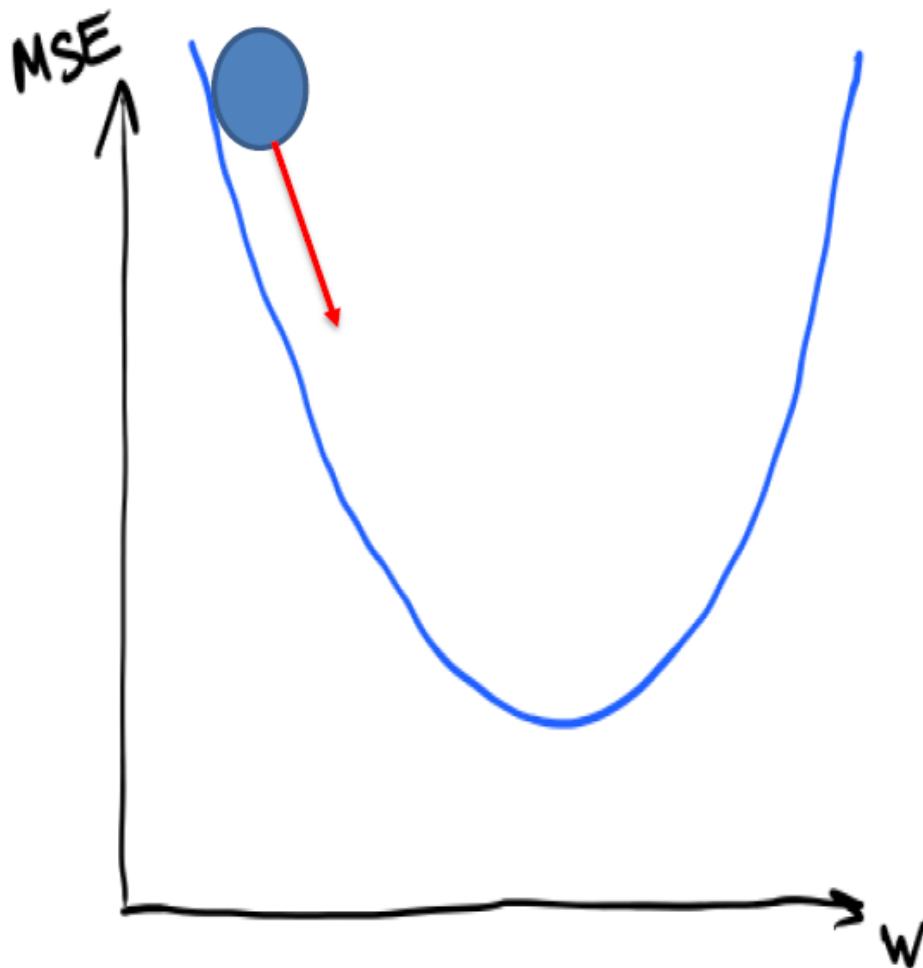
На каждом шаге (на каждой итерации метода) движемся в сторону антиградиента функции потерь!

То есть на каждом шаге движемся в направлении уменьшения ошибки.

Вектор градиента функции потерь обозначают ***grad Q*** или **∇Q** .

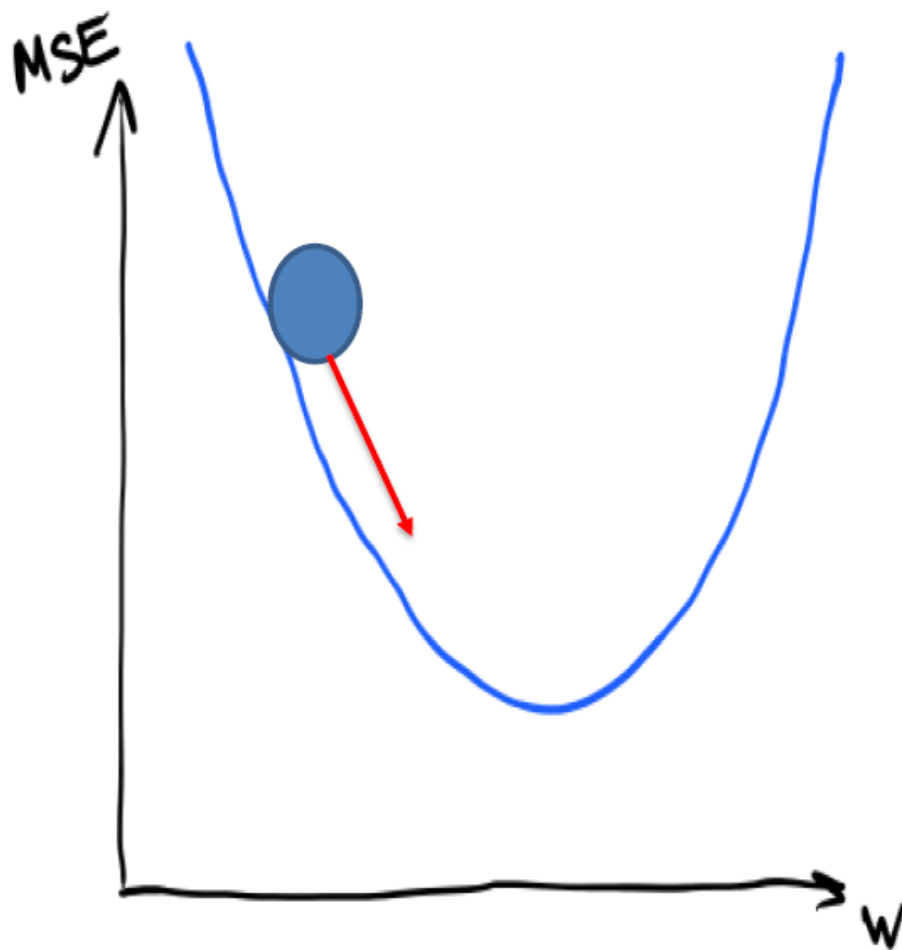
МЕТОД ГРАДИЕНТНОГО СПУСКА

На каждом шаге (на каждой итерации метода) движемся в сторону антиградиента функции потерь!



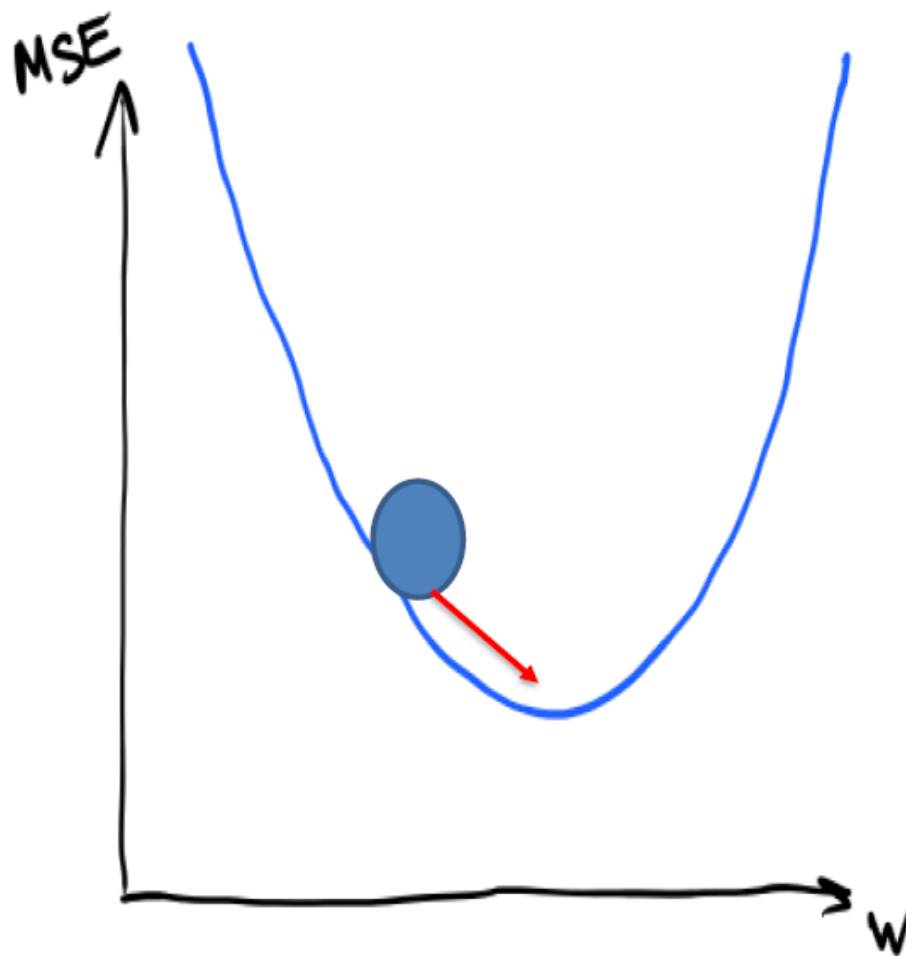
МЕТОД ГРАДИЕНТНОГО СПУСКА

На каждом шаге (на каждой итерации метода) движемся в сторону антиградиента функции потерь!



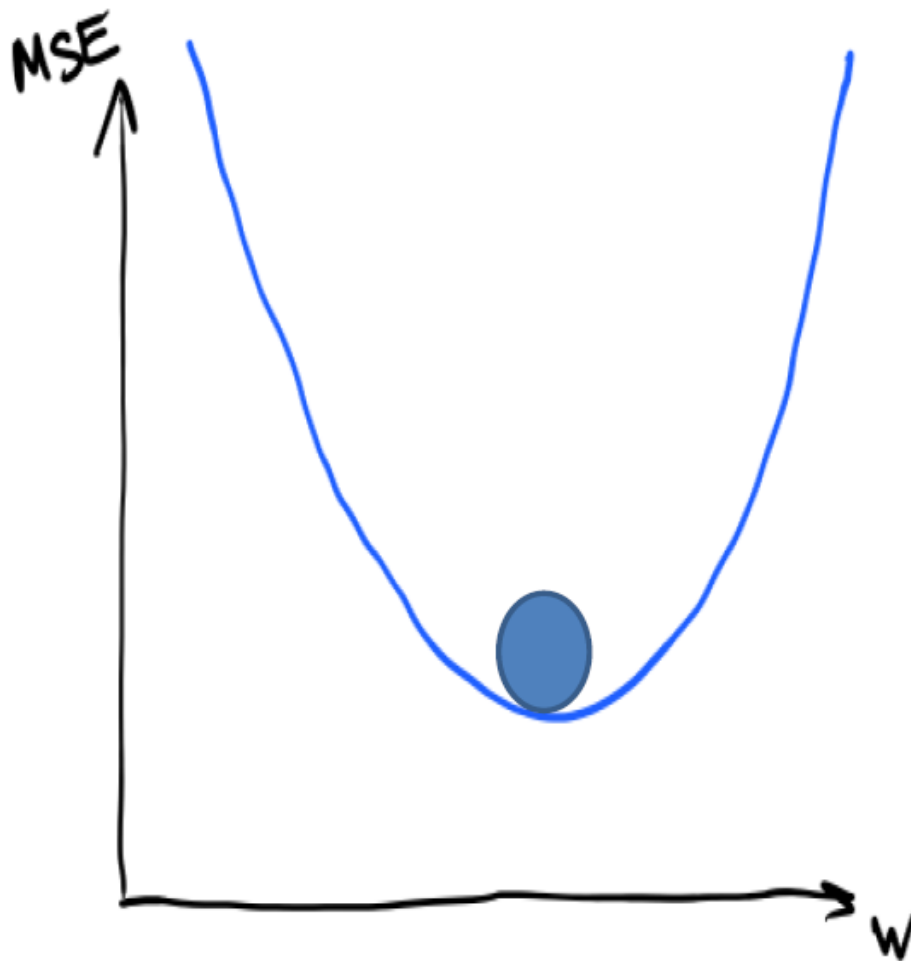
МЕТОД ГРАДИЕНТНОГО СПУСКА

На каждом шаге (на каждой итерации метода) движемся в сторону антиградиента функции потерь!



МЕТОД ГРАДИЕНТНОГО СПУСКА

На каждом шаге (на каждой итерации метода) движемся в сторону антиградиента функции потерь!



МЕТОД ГРАДИЕНТНОГО СПУСКА

На каждом шаге (на каждой итерации метода) движемся в сторону антиградиента функции потерь!

Метод градиентного спуска:

- Инициализируем веса $w_0^{(0)}, w_1^{(0)}, w_2^{(0)}, \dots, w_n^{(0)}$.

МЕТОД ГРАДИЕНТНОГО СПУСКА

На каждом шаге (на каждой итерации метода) движемся в сторону антиградиента функции потерь!

Метод градиентного спуска:

- Инициализируем веса $w_0^{(0)}, w_1^{(0)}, w_2^{(0)}, \dots, w_n^{(0)}$.
- На каждом следующем шаге обновляем веса, сдвигаясь в направлении антиградиента функции потерь Q :

$$w_0^{(k)} = w_0^{(k-1)} - \nabla Q(w_0^{(k-1)}),$$

МЕТОД ГРАДИЕНТНОГО СПУСКА

На каждом шаге (на каждой итерации метода) движемся в сторону антиградиента функции потерь!

Метод градиентного спуска:

- Инициализируем веса $w_0^{(0)}, w_1^{(0)}, w_2^{(0)}, \dots, w_n^{(0)}$.
- На каждом следующем шаге обновляем веса, сдвигаясь в направлении антиградиента функции потерь Q :

$$w_0^{(k)} = w_0^{(k-1)} - \nabla Q(w_0^{(k-1)}),$$

$$w_1^{(k)} = w_1^{(k-1)} - \nabla Q(w_1^{(k-1)}),$$

...

$$w_n^{(k)} = w_n^{(k-1)} - \nabla Q(w_n^{(k-1)}),$$

МЕТОД ГРАДИЕНТНОГО СПУСКА

На каждом шаге (на каждой итерации метода) движемся в сторону антиградиента функции потерь!

Метод градиентного спуска можно записать в векторном виде:

- Инициализируем веса $w^{(0)}$.
- На каждом следующем шаге обновляем веса по формуле:

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

МЕТОД ГРАДИЕНТНОГО СПУСКА

На каждом шаге (на каждой итерации метода) движемся в сторону антиградиента функции потерь!

Метод градиентного спуска можно записать в векторном виде:

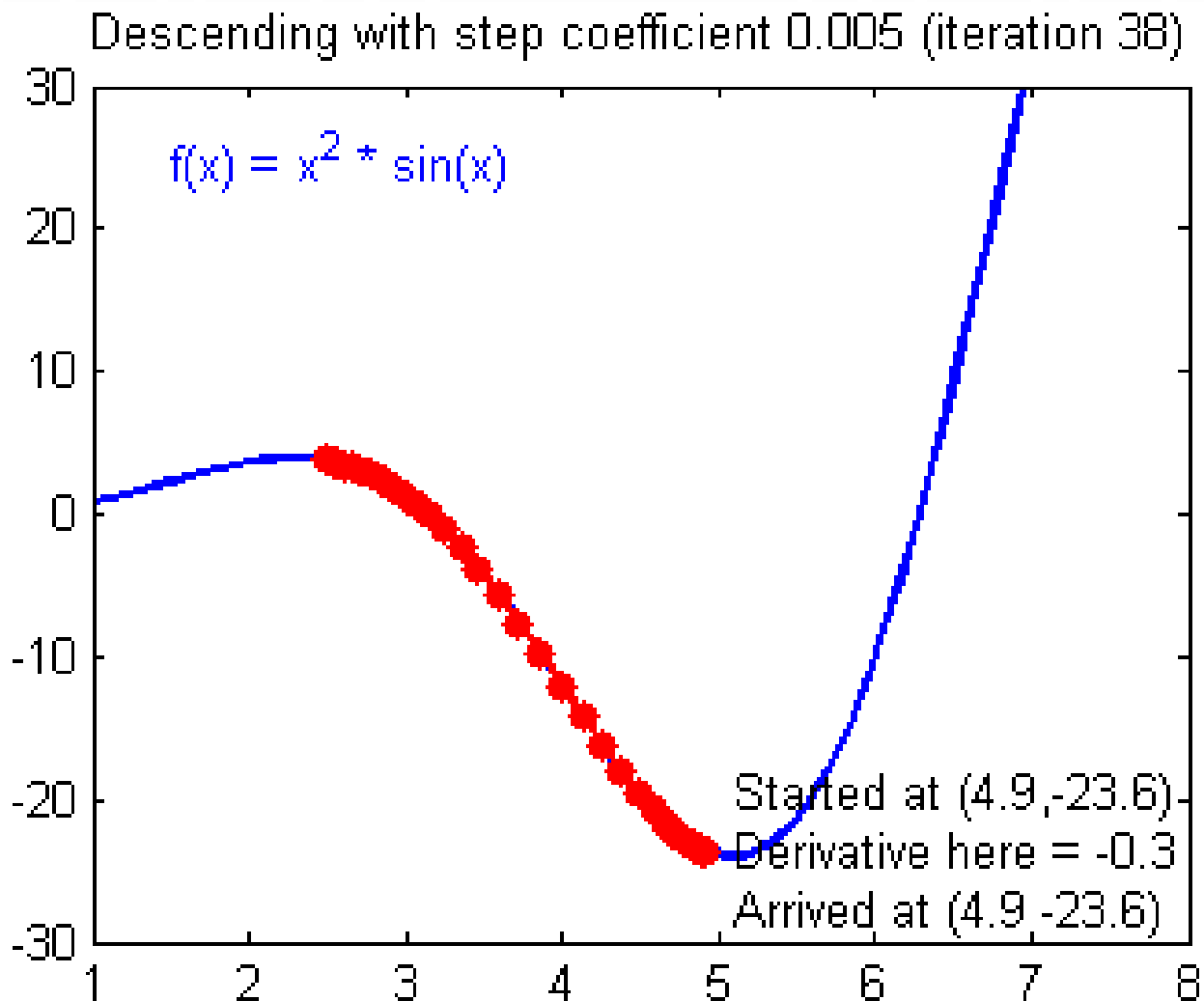
- Инициализируем веса $\mathbf{w}^{(0)}$.
- На каждом следующем шаге обновляем веса по формуле:

$$\mathbf{w}^{(k)} = \mathbf{w}^{(k-1)} - \nabla Q(\mathbf{w}^{(k-1)})$$

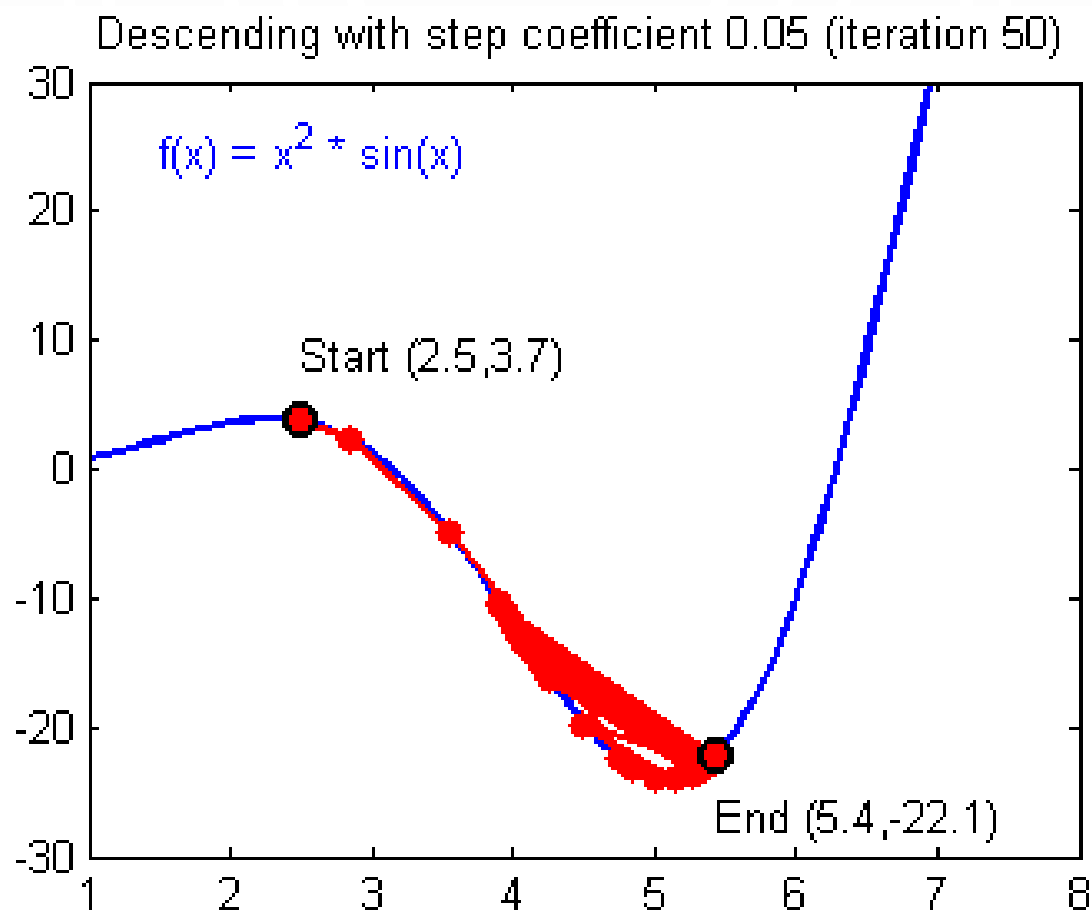
В формулу обычно добавляют параметр η – величина градиентного шага (learning rate). Он отвечает за скорость движения в сторону антиградиента:

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

ГРАДИЕНТНЫЙ СПУСК



ПРОБЛЕМА ВЫБОРА ГРАДИЕНТНОГО ШАГА



ВАРИАНТЫ ИНИЦИАЛИЗАЦИИ ВЕСОВ

- $w_j = 0, j = 1, \dots, n$
- Небольшие случайные значения:

$$w_j := \text{random}(-\varepsilon, \varepsilon)$$

- Обучение по небольшой случайной подвыборке объектов
- Мультистарт: многократный запуск из разных случайных начальных приближений и выбор лучшего решения

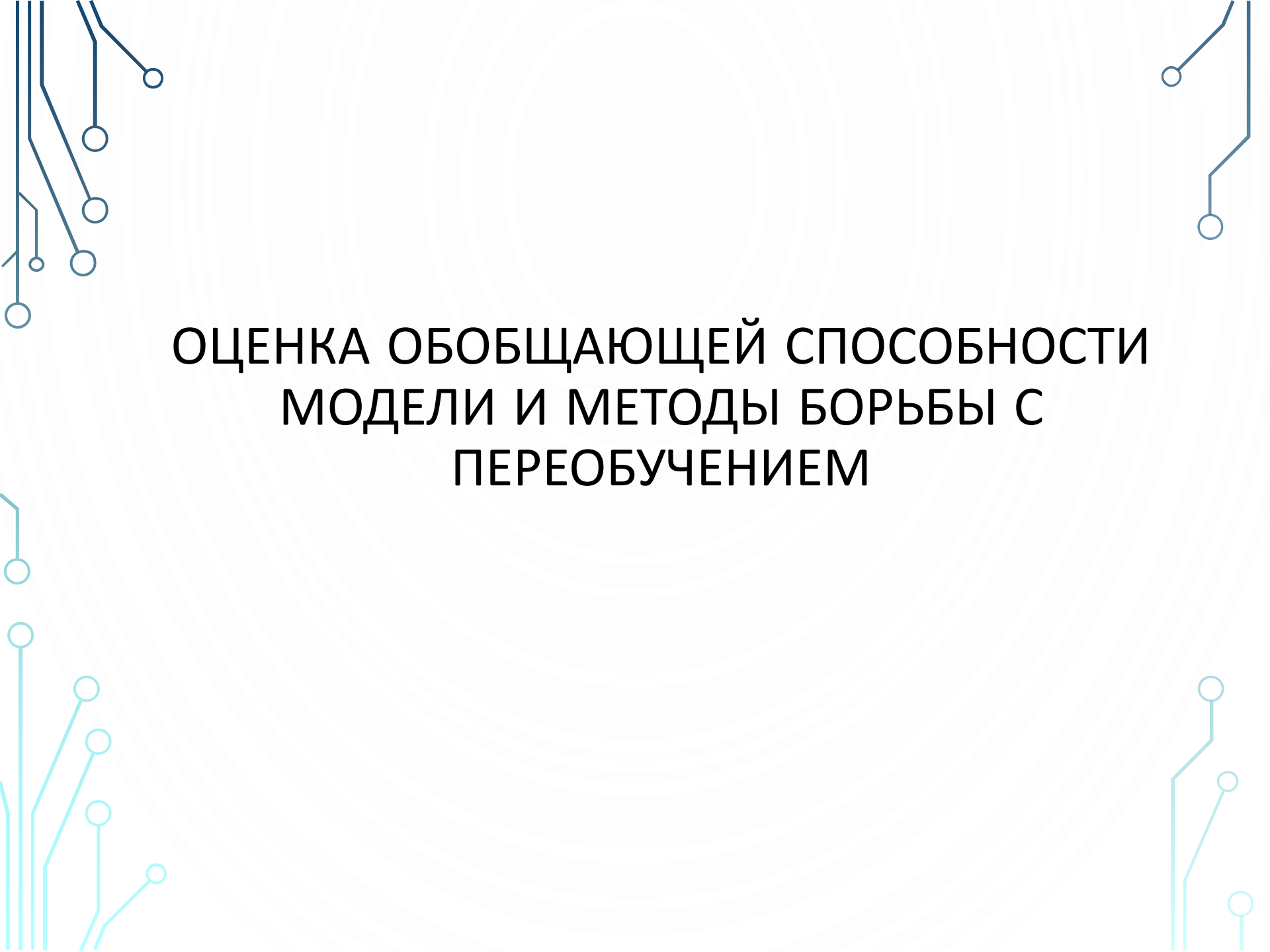
КРИТЕРИИ ОСТАНОВА

- $|\nabla Q(w^{(k-1)})| < \varepsilon$

- $\Delta w = |w^{(k)} - w^{(k-1)}| < \varepsilon$

ГРАДИЕНТНЫЙ ШАГ

- $\eta_k = c$
- $\eta_k = \frac{1}{k}$
- $\eta_k = \lambda \left(\frac{s_0}{s_0+k} \right)^p$, λ, s_0, p - параметры

The image features a light blue background with a subtle pattern of concentric circles. In the four corners, there are decorative elements resembling circuit board traces or neural network connections, consisting of thin blue lines and small circles.

ОЦЕНКА ОБОБЩАЮЩЕЙ СПОСОБНОСТИ МОДЕЛИ И МЕТОДЫ БОРЬБЫ С ПЕРЕОБУЧЕНИЕМ

ОЦЕНКА ОБОБЩАЮЩЕЙ СПОСОБНОСТИ МОДЕЛИ

Переобучение (overfitting) – явление, при котором качество модели на новых данных сильно хуже, чем качество на тренировочных данных.

Fitting training data

Degree = 1

— True function
— Model
• Training data (MSE = 1.37)

Underfitting

Degree = 2

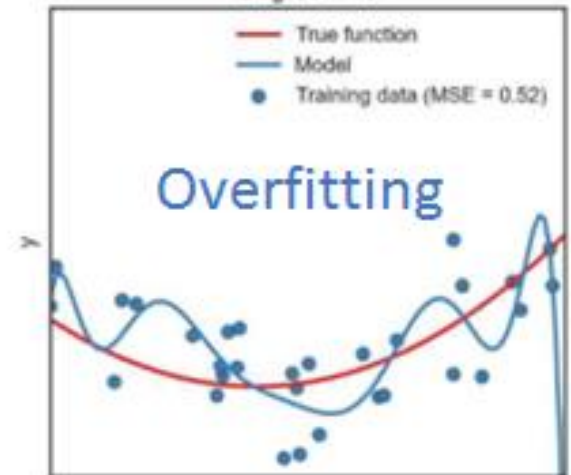
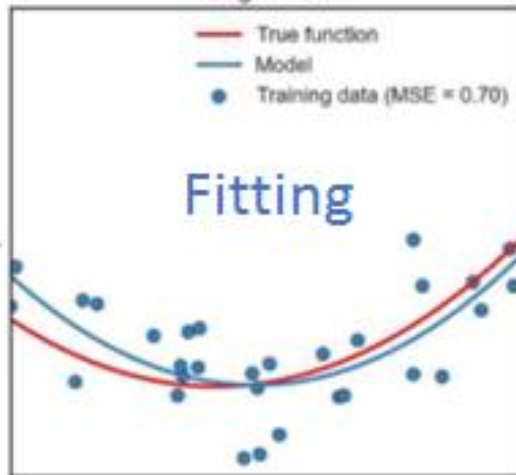
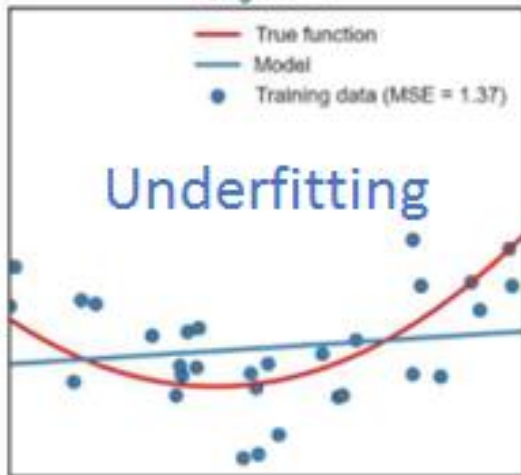
— True function
— Model
• Training data (MSE = 0.70)

Fitting

Degree = 10

— True function
— Model
• Training data (MSE = 0.52)

Overfitting



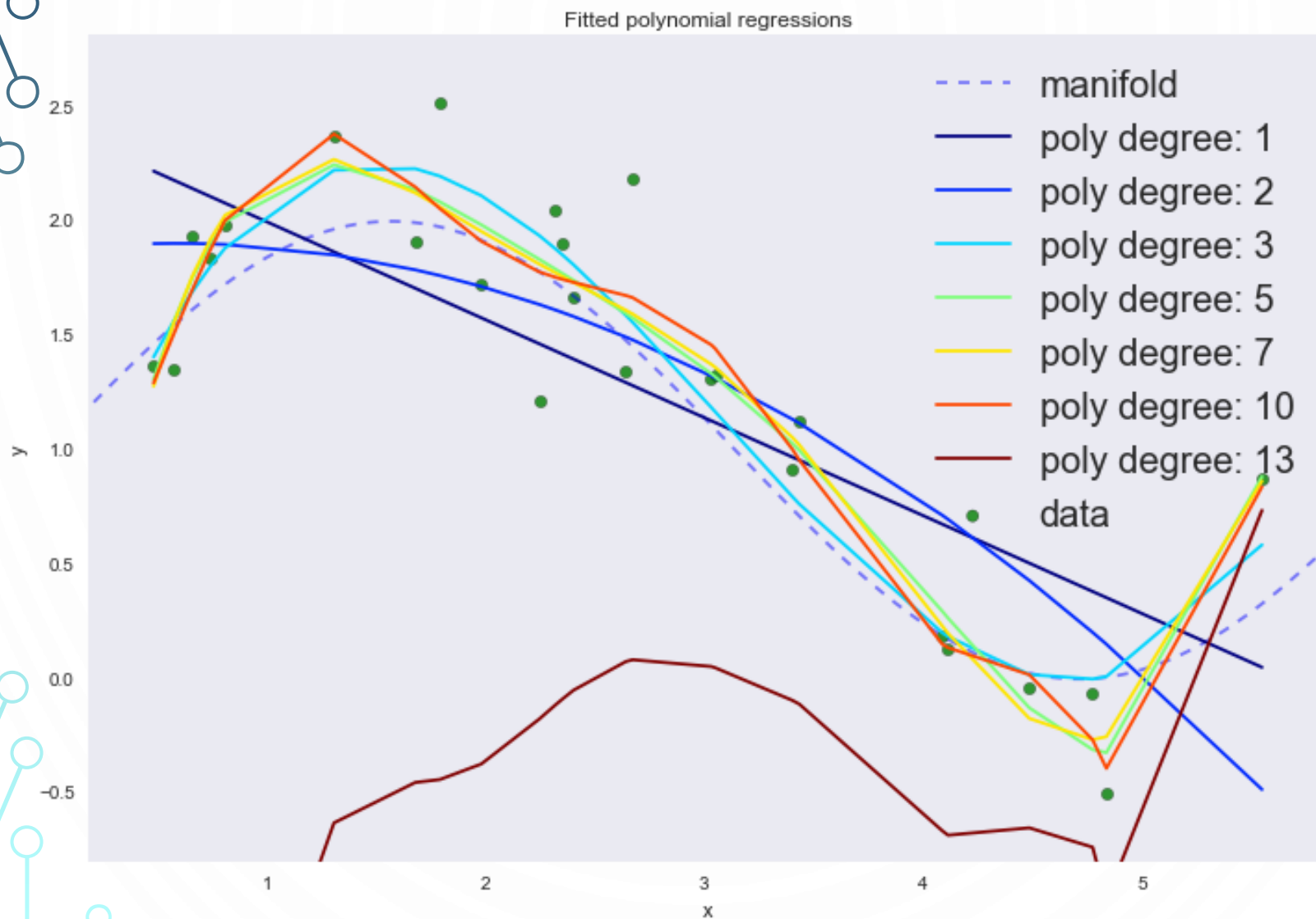
ПРИЗНАКИ ПЕРЕОБУЧЕННОЙ МОДЕЛИ

- Большая разница в качестве на тренировочных и тестовых данных (модель подгоняется под тренировочные данные и не может найти истинную зависимость)

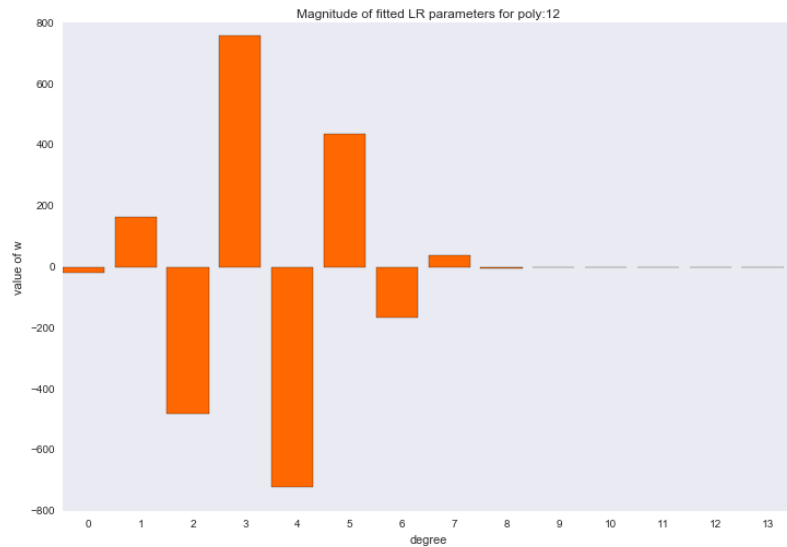
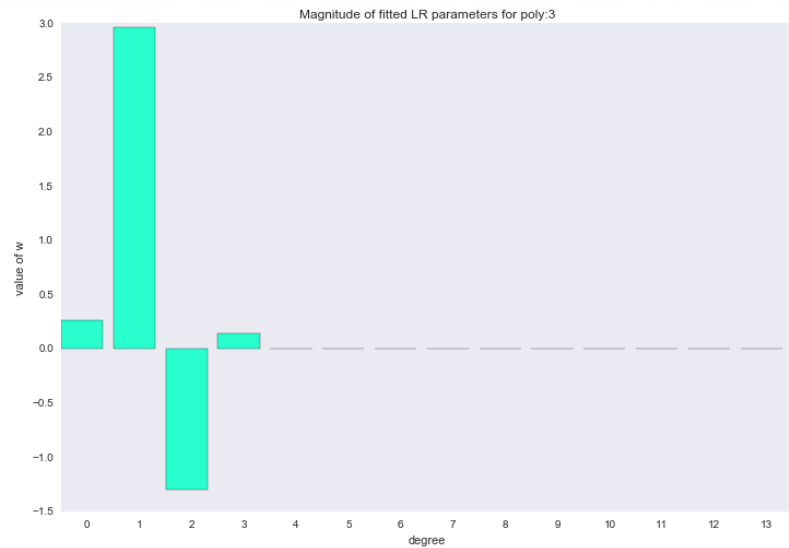
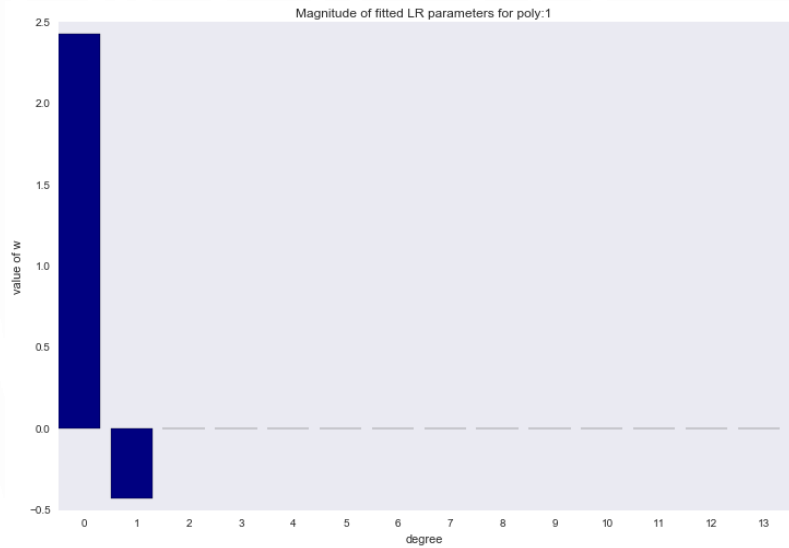
ПРИЗНАКИ ПЕРЕОБУЧЕННОЙ МОДЕЛИ

- Большая разница в качестве на тренировочных и тестовых данных (модель подгоняется под тренировочные данные и не может найти истинную зависимость)
- Большие значения параметров (весов) w_j модели

ПЕРЕОБУЧЕНИЕ: ПРИМЕР



ПЕРЕОБУЧЕНИЕ: ПРИМЕР



ОЦЕНИВАНИЕ КАЧЕСТВА МОДЕЛИ

- Отложенная выборка
- Кросс-валидация

ОТЛОЖЕННАЯ ВЫБОРКА

Делим тренировочную выборку на две части:

- По первой части обучаем модель (train)
- По оставшимся данным – оцениваем качество (test)



Какой недостаток?

ОТЛОЖЕННАЯ ВЫБОРКА

Делим тренировочную выборку на две части:

- По первой части обучаем модель (train)
- По оставшимся данным – оцениваем качество (test)

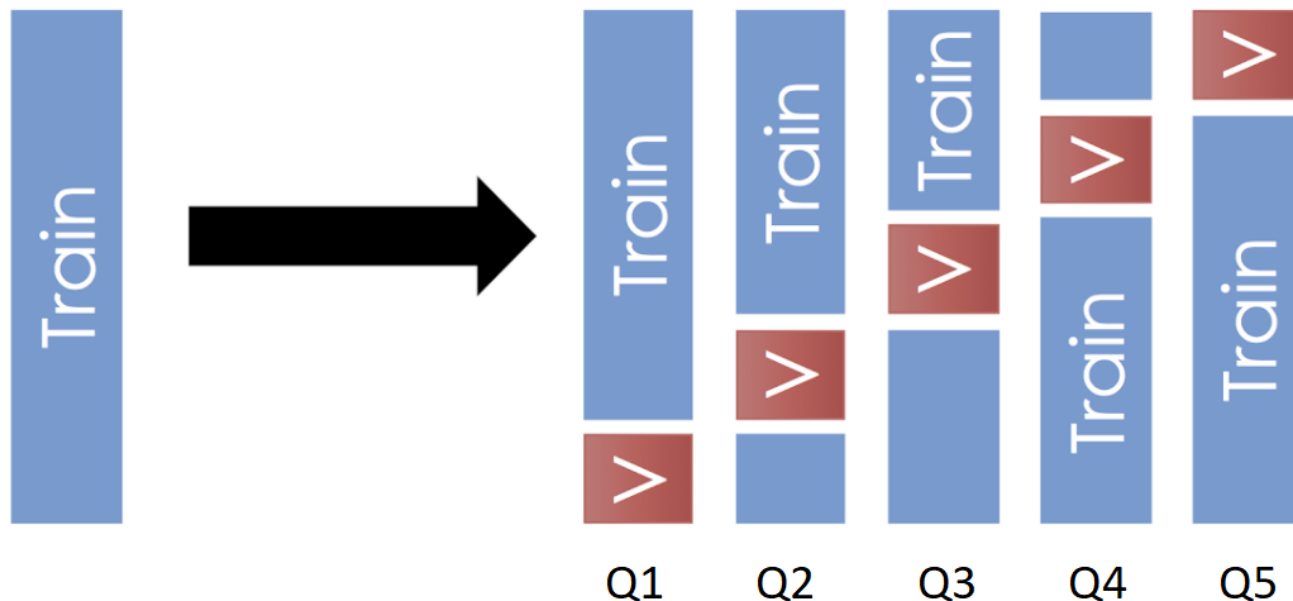


Недостаток:

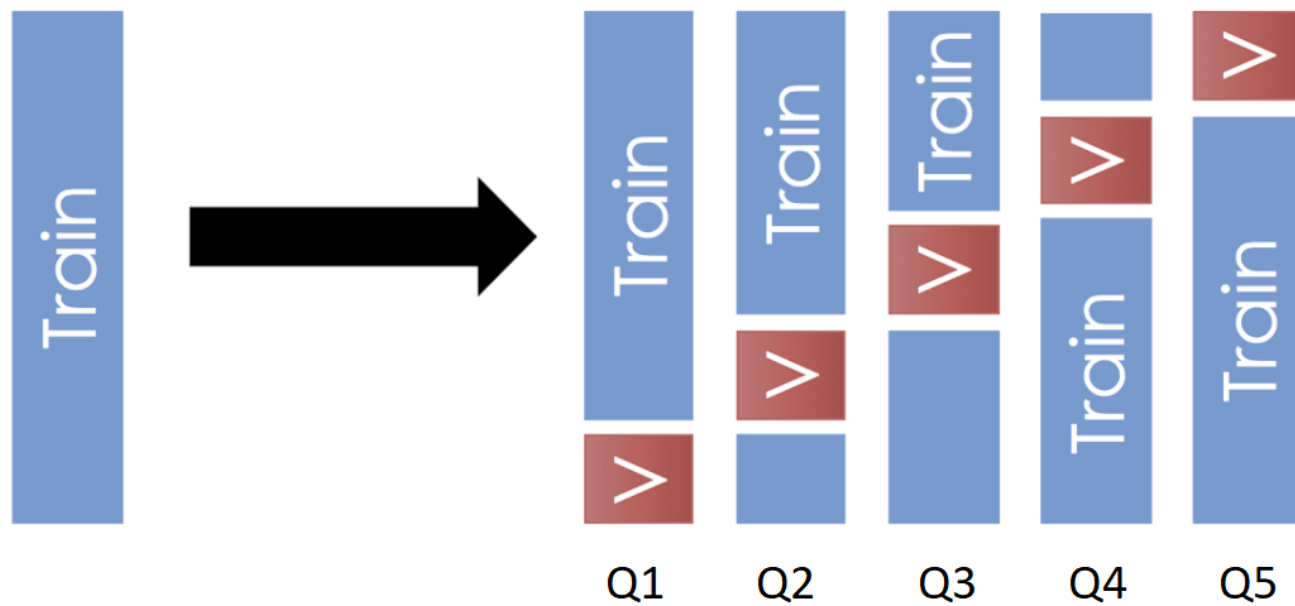
- Результат сильно зависит от разбиения на train и test

КРОСС-ВАЛИДАЦИЯ

- Разбиваем объекты на тренировку (train) и валидацию (validation) **несколько раз** (при разбиении k раз получаем k -fold кросс-валидацию)
- Для каждого разбиения вычисляем качество на валидационной части
- Усредняем полученные результаты



КРОСС-ВАЛИДАЦИЯ



$$CV = \frac{1}{k} \sum_{i=1}^k Q_i$$

ВИДЫ КРОСС-ВАЛИДАЦИИ

- **k-fold cross-validation** – разбиваем данные на k блоков, каждый из которых по очереди становится контрольным (валидационным)
- **Complete cross-validation** – перебираем ВСЕ разбиения
- **Leave-one-out cross-validation** – каждый блок состоит из одного объекта (число блоков = числу объектов)

ВЫБОР КОЛИЧЕСТВА БЛОКОВ В K-FOLD КРОСС-ВАЛИДАЦИИ



- Маленькое k – оценка может быть пессимистично занижена из-за маленького размера тренировочной части
- Большое k – оценка может быть неустойчивой из-за маленького размера валидационной части