

Занятие 2

Линейные методы

регрессии. Часть 1

Дубенюк Анна Антоновна

anya.dubenyuk@yandex.ru

@andu192

ВШЭ, 2023

ОЦЕНКА ПРЕДСКАЗАТЕЛЬНОЙ СПОСОБНОСТИ АЛГОРИТМА

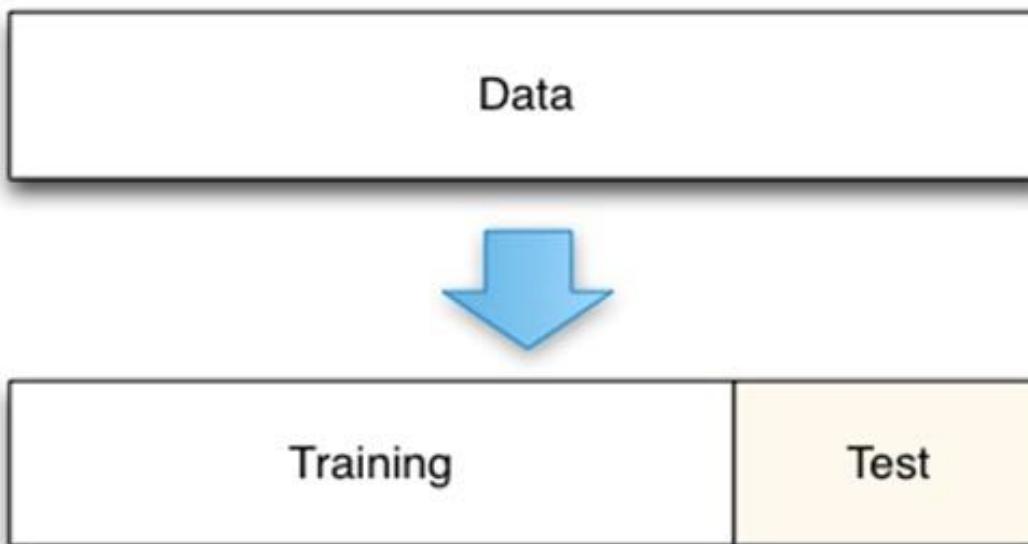
- Пусть мы решаем задачу *предсказания стоимости дома* по его признакам.



- В обучающей выборке 1000 домов.
- Мы обучаем алгоритм по имеющимся 1000 домам. На каких объектах будем проверять качество алгоритма?

ОЦЕНКА ПРЕДСКАЗАТЕЛЬНОЙ СПОСОБНОСТИ АЛГОРИТМА

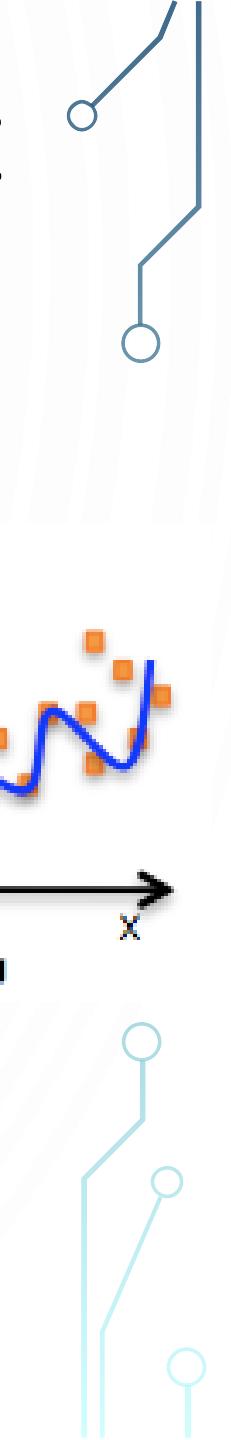
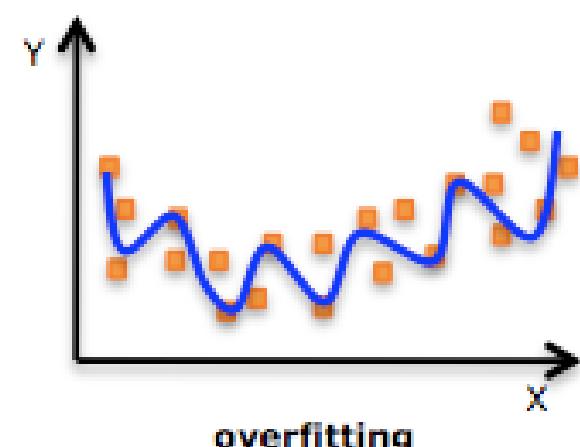
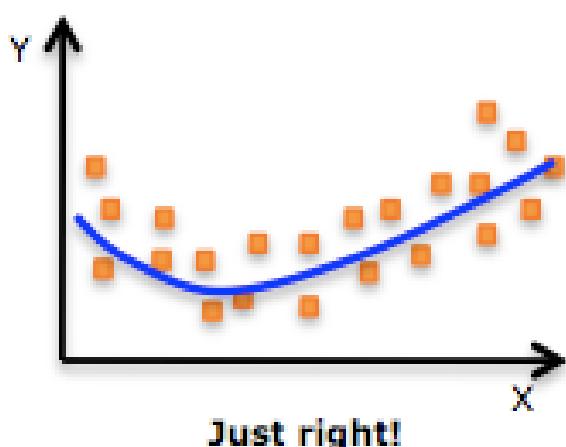
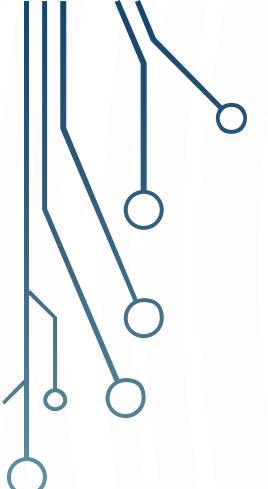
- Перед началом обучения отложим часть обучающих объектов и не будем использовать их для построения модели (отложенная выборка).



ОТЛОЖЕННАЯ ВЫБОРКА

- Перед началом обучения отложим часть обучающих объектов и не будем использовать их для построения модели (отложенная выборка).
- Тогда можно измерить качество построенной модели на отложенной выборке и оценить ее предсказательную силу.

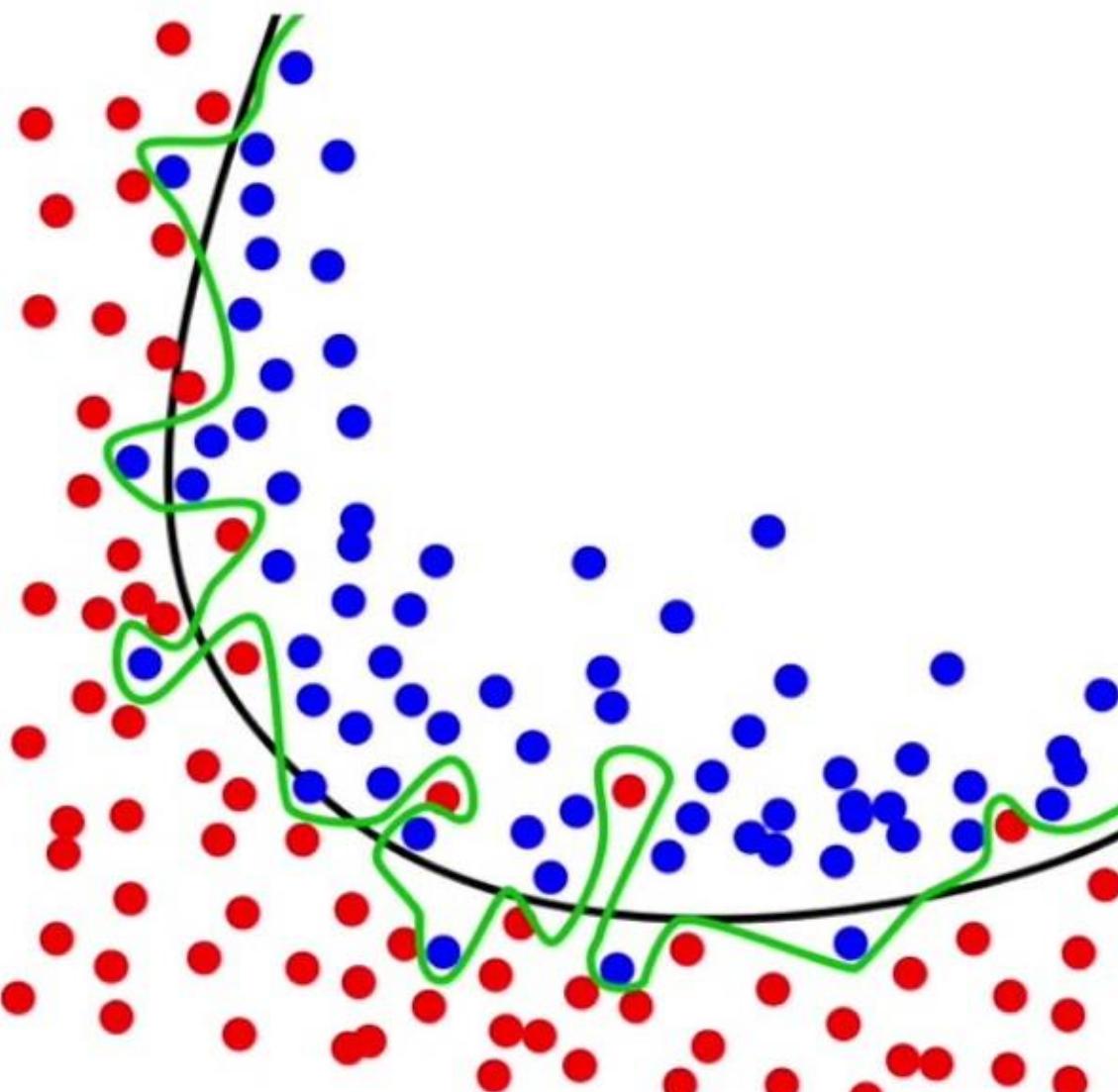
ПЕРЕОБУЧЕНИЕ И НЕДООБУЧЕНИЕ



ИЗ-ЗА ЧЕГО ВОЗНИКАЕТ ПЕРЕОБУЧЕНИЕ

- Избыточная сложность модели (большое количество весов). В этом случае лишние степени свободы в модели “тратятся” на чрезмерно точную подгонку под обучающую выборку.
- Переобучение есть всегда, когда есть оптимизация параметров по конечной (заведомо неполной) выборке.

ПРИМЕР ПЕРЕОБУЧЕНИЯ В ЗАДАЧЕ КЛАССИФИКАЦИИ



ПРИЗНАК ПЕРЕОБУЧЕНИЯ

- *Если качество на отложенной выборке сильно ниже качества на обучающих данных, то происходит переобучение*

ЛИНЕЙНАЯ РЕГРЕССИЯ

Пример (напоминание):

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

Линейная модель для предсказания стоимости:

$$a(x) = w_0 + w_1 x_1 + w_2 x_2,$$

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 *количество 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_n - признаки объекта x .

ЛИНЕЙНАЯ РЕГРЕССИЯ

Линейная регрессия:

$$a(x) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

ЛИНЕЙНАЯ РЕГРЕССИЯ

Линейная регрессия:

$$a(x) = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n$$

- сокращенная запись:

$$a(x) = w_0 + \sum_{j=1}^n w_jx_j$$

ЛИНЕЙНАЯ РЕГРЕССИЯ

Линейная регрессия:

$$a(x) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

- сокращенная запись:

$$a(x) = w_0 + \sum_{j=1}^n w_jx_j$$

- запись через скалярное произведение (с добавлением признака $x_0 = 1$):

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

ЛИНЕЙНАЯ РЕГРЕССИЯ

Линейная регрессия:

$$a(x) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

- сокращенная запись:

$$a(x) = w_0 + \sum_{j=1}^n w_jx_j$$

- запись через скалярное произведение (с добавлением признака $x_0 = 1$):

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

ЛИНЕЙНАЯ РЕГРЕССИЯ

Линейная регрессия:

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

Обучение линейной регрессии - минимизация
среднеквадратичной ошибки:

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

(здесь l – количество объектов)

О ПРИМЕНИМОСТИ ЛИНЕЙНОЙ МОДЕЛИ

Пример:

Предположим, что мы хотим предсказать *стоимость дома* y по его *площади* (x_1) и *количество комната* (x_2), *району* (x_3) и *удаленности от МКАД* (x_4).

$$a(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4.$$



О ПРИМЕНИМОСТИ ЛИНЕЙНОЙ МОДЕЛИ

Пример:

Предположим, что мы хотим предсказать *стоимость дома* y по его *площади* (x_1) и *количество комната* (x_2), *району* (x_3) и *удаленности от МКАД* (x_4).

$$a(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4.$$

Проблема №1: район (x_3) – это не число, а название района. Например, Мамыри, Дудкино, Барвиха... Что с этим делать?



О ПРИМЕНИМОСТИ ЛИНЕЙНОЙ МОДЕЛИ

Пример:

Предположим, что мы хотим предсказать *стоимость дома* у него *площади* (x_1) и *количество комнат* (x_2), *району* (x_3) и *удаленности от МКАД* (x_4).

$$a(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4.$$

Проблема №1: район (x_3) – это не число, а название района. Например, Мамыри, Дудкино, Барвиха... Что с этим делать?

Решение – one-hot encoding (ОНЕ): создаем новые числовые столбцы, каждый из которых является индикатором района.



ONE-HOT ENCODING



Район
Дудкино
Барвиха
Мамыри
...
Барвиха



Мамыри	Дудкино	Барвиха
0	1	0
0	0	1
1	0	0
...
0	0	1

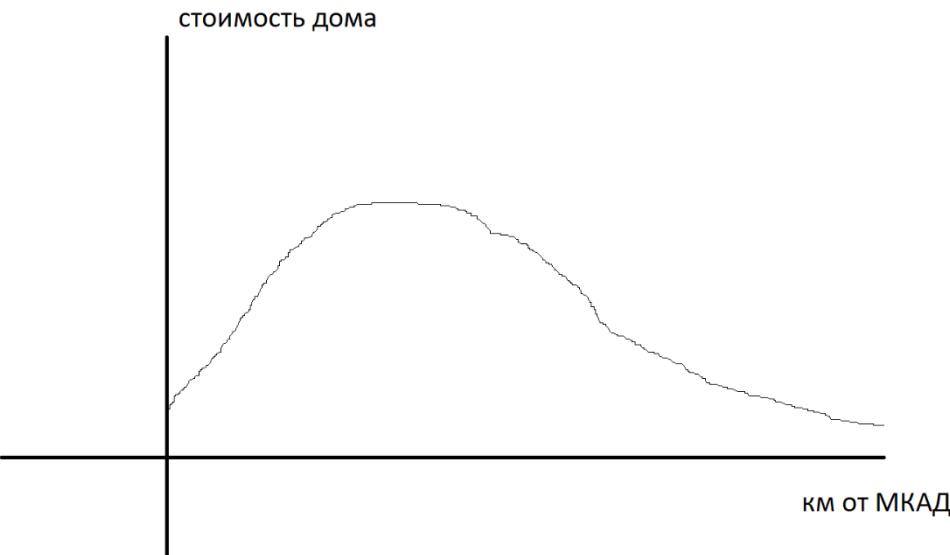
$$\begin{aligned}a(x) &= \\&= w_0 + w_1 x_1 + w_2 x_2 + w_{31} x_{\text{Мамыри}} + w_{32} x_{\text{Дудкино}} + w_{33} x_{\text{Барвиха}} + w_4 x_4.\end{aligned}$$

О ПРИМЕНИМОСТИ ЛИНЕЙНОЙ МОДЕЛИ

Пример:

Предположим, что мы хотим предсказать *стоимость дома* y по его *площади* (x_1) и *количество комната* (x_2), *району* (x_3) и *удаленности от МКАД* (x_4).

Проблема №2: удаленность от МКАД (x_4) не монотонно влияет на стоимость дома.



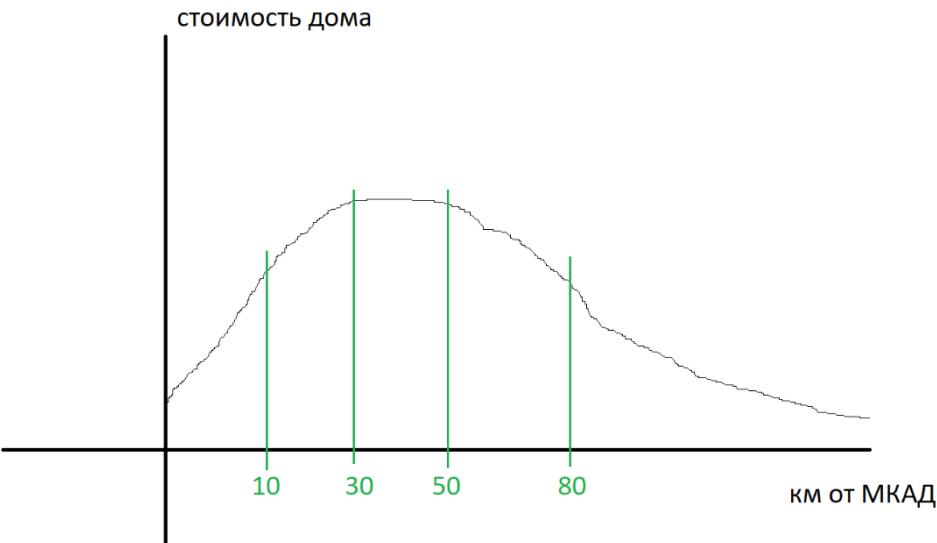
О ПРИМЕНИМОСТИ ЛИНЕЙНОЙ МОДЕЛИ

Проблема №2: удаленность от МКАД (x_4) не монотонно влияет на стоимость дома.

Решение – бинаризация (разбиение на бины).

Новые признаки:

- $x_{[0;10)}$ - равен 1, если дом находится в пределах 10 км от МКАД, и 0 иначе
- $x_{[10;30)}$ - равен 1, если дом находится в пределах от 10 км до 30 км МКАД, и 0 иначе. И т.д.



О ПРИМЕНИМОСТИ ЛИНЕЙНОЙ МОДЕЛИ

Проблема №2: удаленность от МКАД (x_4) не монотонно влияет на стоимость дома.

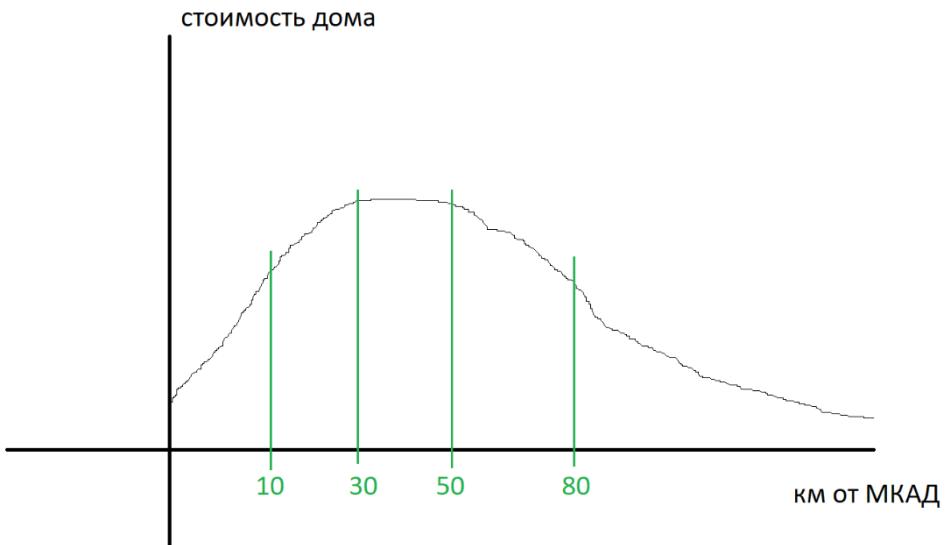
Решение – бинаризация (разбиение на бины).

Новые признаки:

- $x_{[0;10)}$ - равен 1, если дом находится в пределах 10 км от МКАД, и 0 иначе

- $x_{[10;30)}$ - равен 1, если

дом находится в пределах от 10 км до 30 км МКАД, и 0 иначе. И т.д.



$$\begin{aligned}a(x) = & \\= & w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_{41} x_{[0;10)} + w_{42} x_{[10;30)} + w_{43} x_{[30;50)} \\+ & w_{44} x_{\geq 50}\end{aligned}$$

АНАЛИТИЧЕСКОЕ РЕШЕНИЕ ЗАДАЧИ МЕТОДА НАИМЕНЬШИХ КВАДРАТОВ (МНК)

Задача обучения линейной регрессии (в матричной форме):

$$\frac{1}{\ell} \|Xw - y\|^2 \rightarrow \min_w$$

Точное (аналитическое) решение:

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

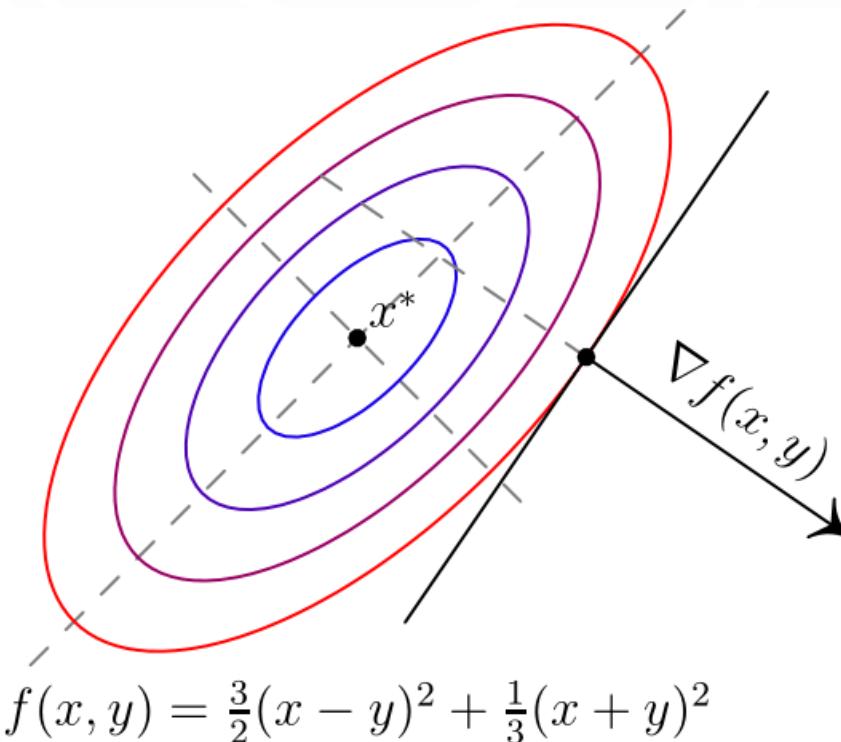
НЕДОСТАТКИ АНАЛИТИЧЕСКОЙ ФОРМУЛЫ

- 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

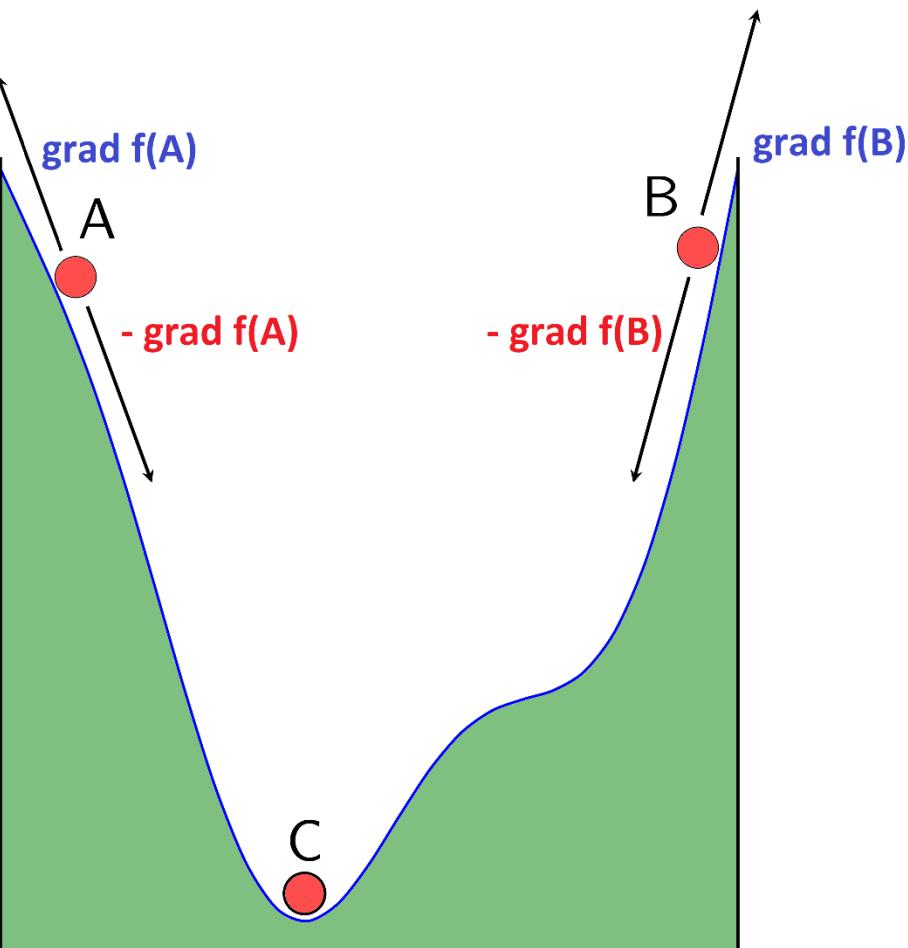
Теорема. Градиент – это вектор, в направлении которого функция быстрее всего растёт.

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.
- В простейшем случае, если ошибка среднеквадратичная, то её график – это парабола.

МЕТОД ГРАДИЕНТНОГО СПУСКА

- Наша задача при обучении модели – найти такие веса w , на которых достигается минимум функции ошибки.
- В простейшем случае, если ошибка среднеквадратичная, то её график – 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.
- В простейшем случае, если ошибка среднеквадратичная, то её график – это парабола.
- Идея метода градиентного спуска:

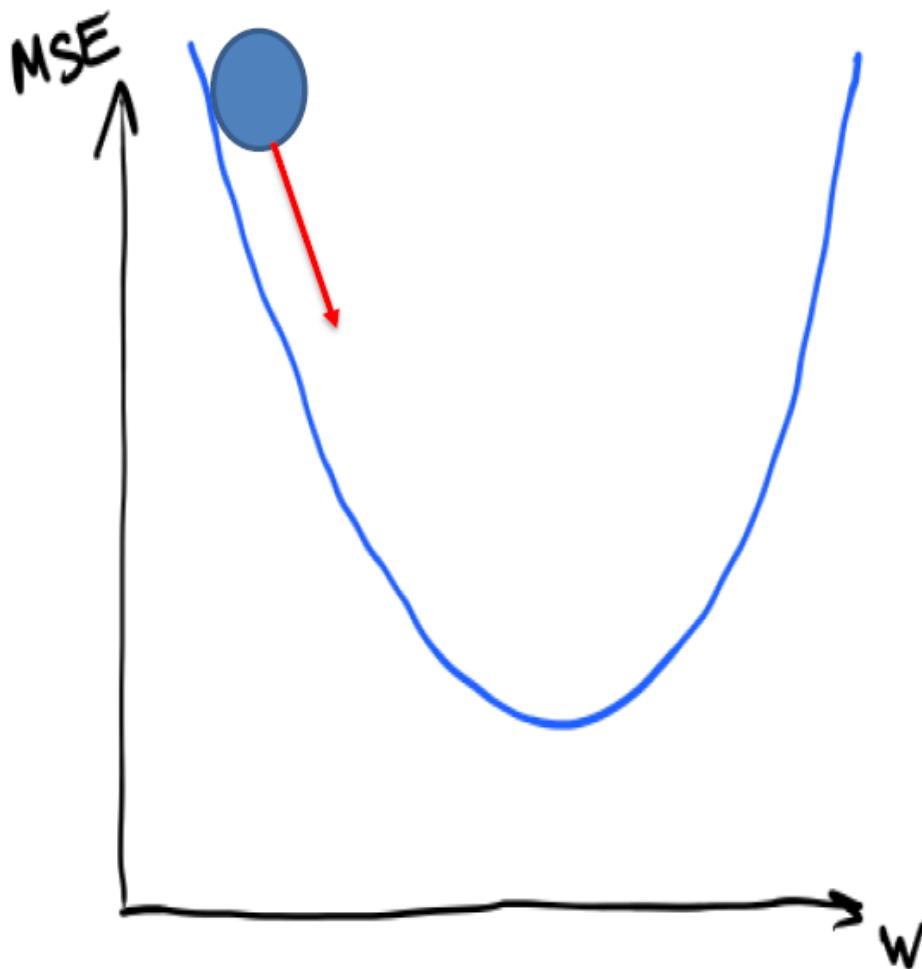
На каждом шаге (на каждой итерации метода) движемся в сторону антиградиента функции потерь!

То есть на каждом шаге движемся в направлении уменьшения ошибки.

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

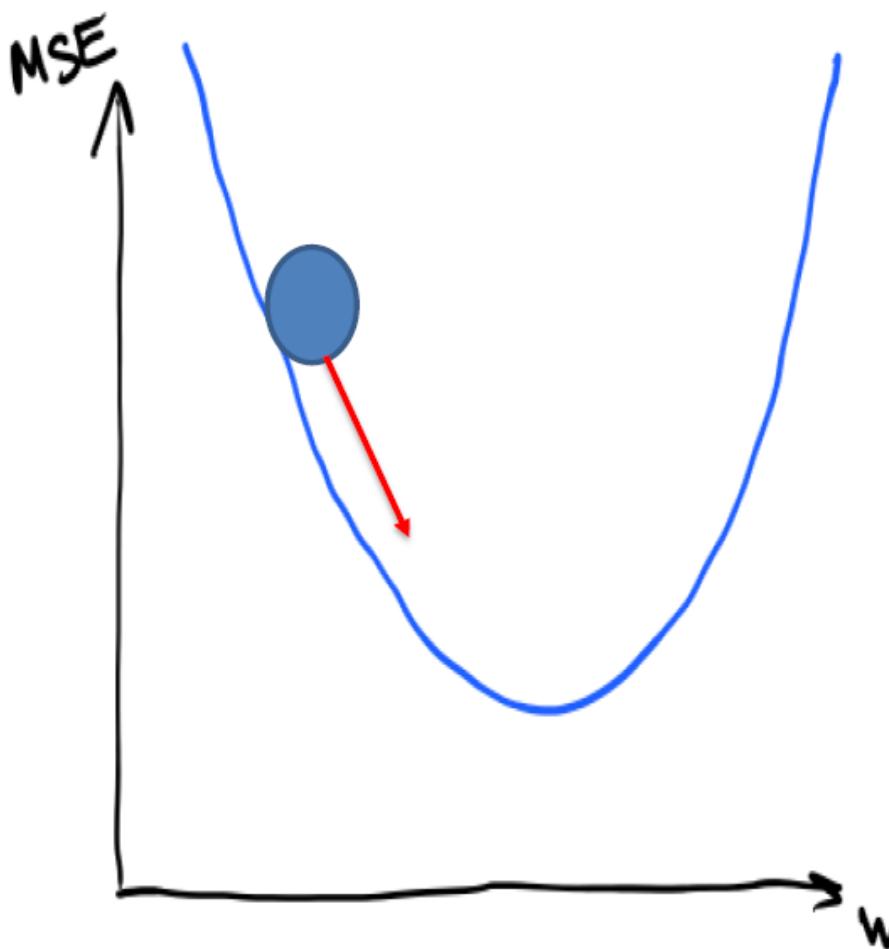
МЕТОД ГРАДИЕНТНОГО СПУСКА

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



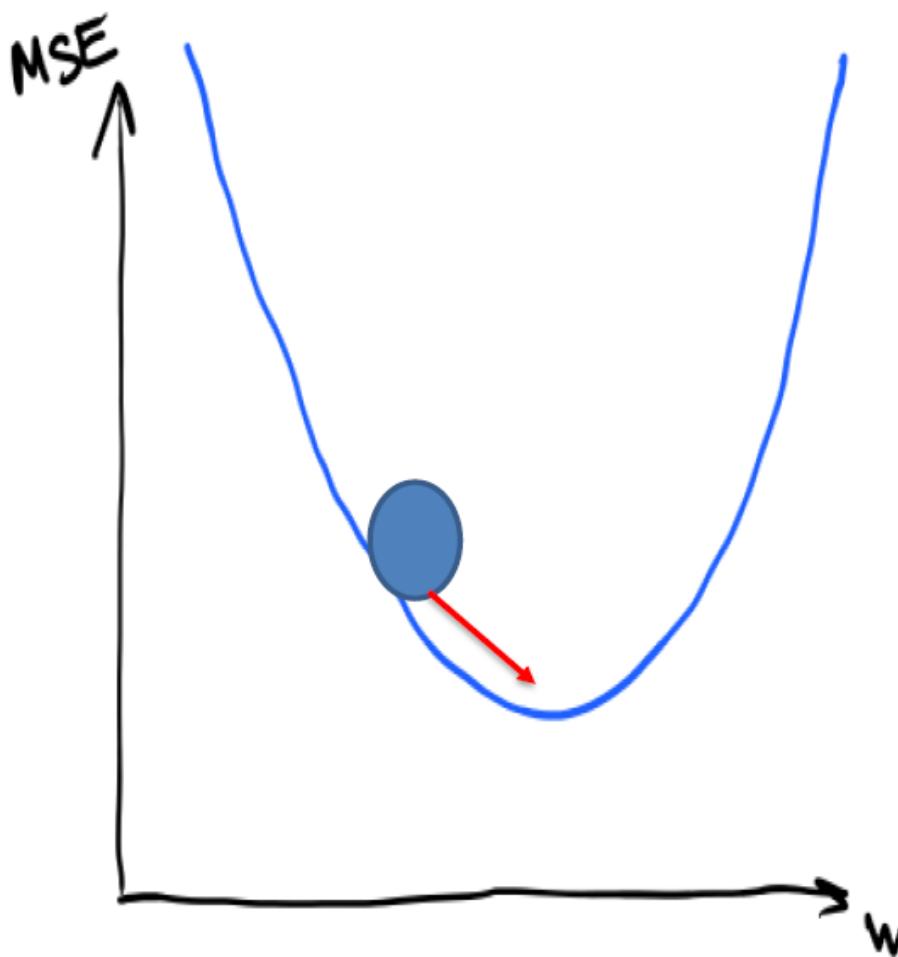
МЕТОД ГРАДИЕНТНОГО СПУСКА

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



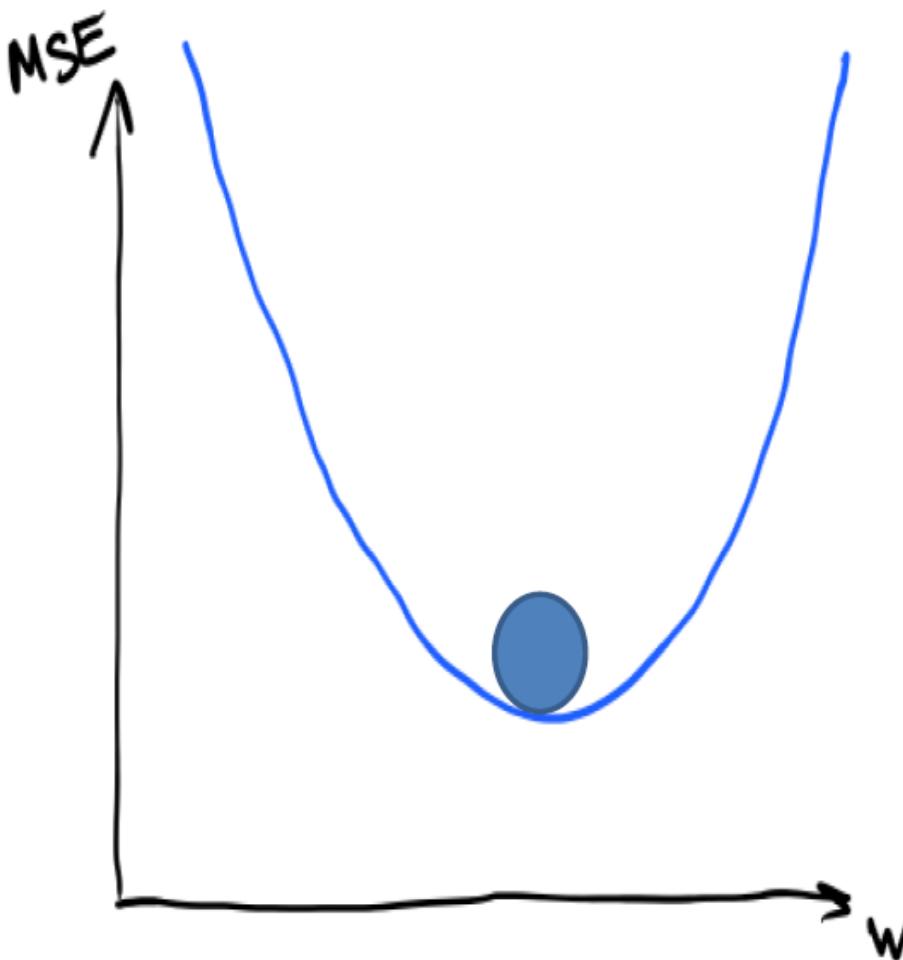
МЕТОД ГРАДИЕНТНОГО СПУСКА

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

Метод градиентного спуска (одномерный случай):

Пусть у нас только один вес - w .

Тогда при добавлении к весу w слагаемого $-\frac{\partial Q}{\partial w}$ функция $Q(w)$ убывает.

МЕТОД ГРАДИЕНТНОГО СПУСКА

Метод градиентного спуска (одномерный случай):

Пусть у нас только один вес - w .

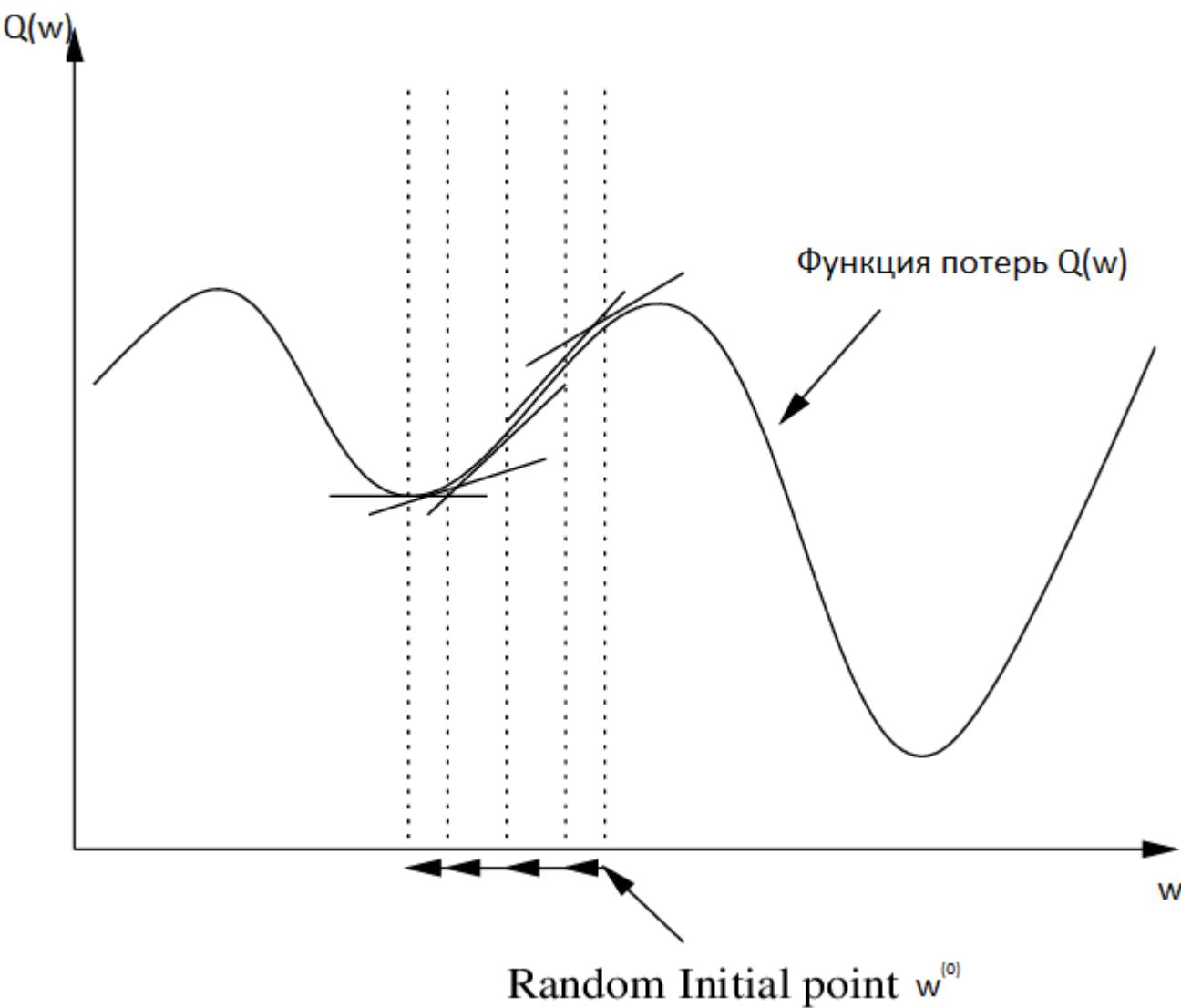
Тогда при добавлении к весу w слагаемого $-\frac{\partial Q}{\partial w}$ функция $Q(w)$ убывает.

- Инициализируем вес $w^{(0)}$.
- На каждом следующем шаге обновляем вес, добавляя

$$-\frac{\partial Q}{\partial w}(w^{(k-1)}) :$$

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

МЕТОД ГРАДИЕНТНОГО СПУСКА



МЕТОД ГРАДИЕНТНОГО СПУСКА

Метод градиентного спуска (общий случай случай):

Пусть w_0, w_1, \dots, w_n - веса, которые мы ищем.

Тогда $\nabla Q(w) = \left\{ \frac{\partial Q}{\partial w_0}, \frac{\partial Q}{\partial w_1}, \dots, \frac{\partial Q}{\partial w_n} \right\}$

МЕТОД ГРАДИЕНТНОГО СПУСКА

Метод градиентного спуска (общий случай случай):

Пусть w_0, w_1, \dots, w_n - веса, которые мы ищем.

Тогда $\nabla Q(w) = \left\{ \frac{\partial Q}{\partial w_0}, \frac{\partial Q}{\partial w_1}, \dots, \frac{\partial Q}{\partial w_n} \right\}$

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

МЕТОД ГРАДИЕНТНОГО СПУСКА

Метод градиентного спуска (общий случай случай):

Пусть w_0, w_1, \dots, w_n - веса, которые мы ищем.

Тогда $\nabla Q(w) = \left\{ \frac{\partial Q}{\partial w_0}, \frac{\partial Q}{\partial w_1}, \dots, \frac{\partial Q}{\partial w_n} \right\}$

- Инициализируем веса $w_0^{(0)}, w_1^{(0)}, w_2^{(0)}, \dots, w_n^{(0)}$.
- На каждом следующем шаге обновляем веса:

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

...

$$w_n^{(k)} = w_n^{(k-1)} - \frac{\partial Q}{\partial w_n} (w_n^{(k-1)}).$$

МЕТОД ГРАДИЕНТНОГО СПУСКА

Формулу для обновления весов 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

Формулу для обновления весов можно записать в векторном виде:

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

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

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

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

МЕТОД ГРАДИЕНТНОГО СПУСКА

Формулу для обновления весов можно записать в векторном виде:

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

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

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

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

Если функция $Q(w)$ выпуклая и гладкая, а также имеет минимум в точке w^* , то метод градиентного спуска при аккуратно подобранном η через некоторое число шагов гарантированно попадет в малую окрестность точки w^* .

МЕТОД ГРАДИЕНТНОГО СПУСКА

Пример (решение на доске):

выписать формулы обновления весов методом градиентного спуска.

$$y = w_0 + w_1 x$$

$$Q(w) = \sum_{i=1}^l (w_0 + w_1 x_i - y_i)^2$$

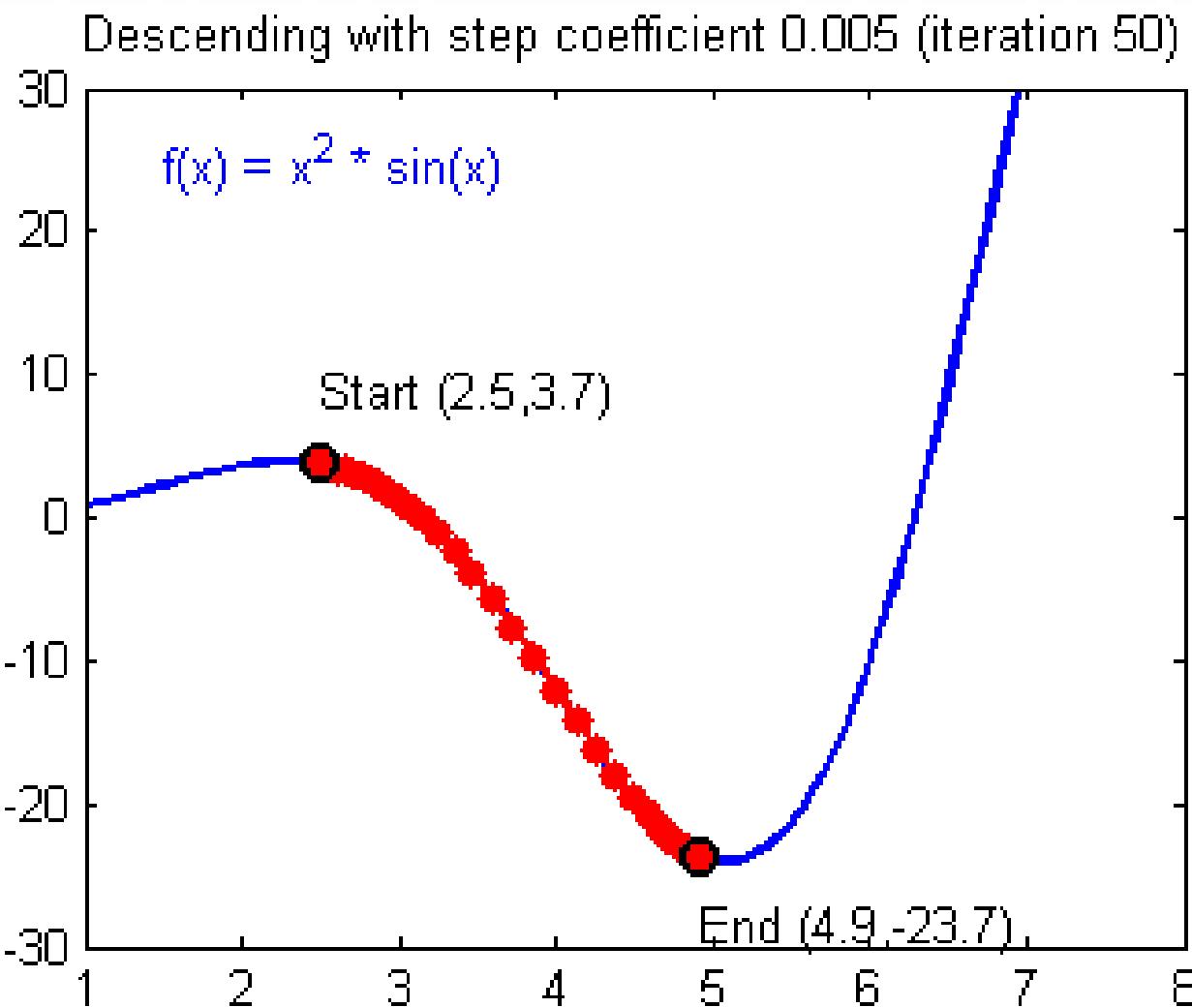
ВАРИАНТЫ ИНИЦИАЛИЗАЦИИ ВЕСОВ

- $w_j = 0, j = 1, \dots, n$
- 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

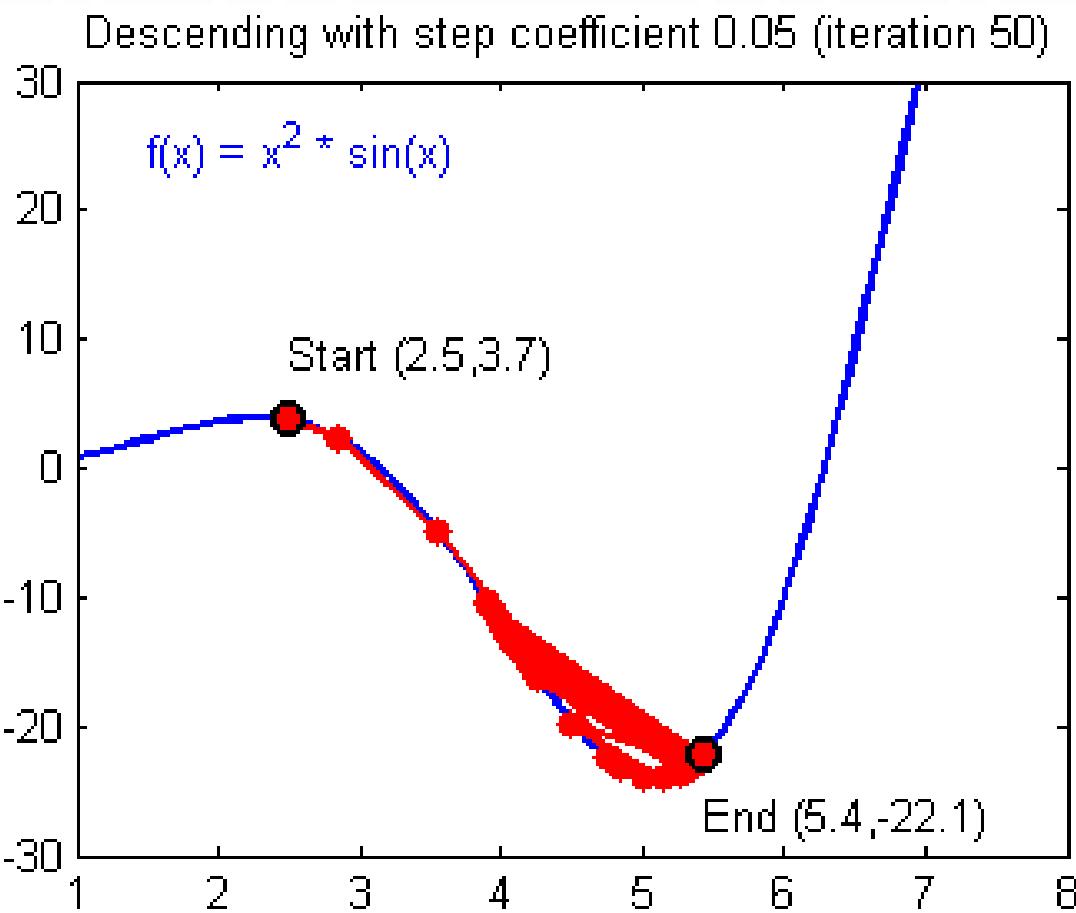
КРИТЕРИИ ОСТАНОВА

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

ГРАДИЕНТНЫЙ СПУСК



ПРОБЛЕМА ВЫБОРА ГРАДИЕНТНОГО ШАГА



ГРАДИЕНТНЫЙ ШАГ

В общем случае градиентный шаг может зависеть от номера итерации, тогда будем писать не η , а η_k .

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

ОДИН ИЗ НЕДОСТАТКОВ ГРАДИЕНТНОГО СПУСКА

(с точки зрения реализации)

- На каждом шаге для вычисления $\nabla Q(w)$ мы вычисляем производную по каждому весу от каждого объекта. То есть вычисляем целую матрицу производных – это затратно и по времени, и по памяти.

СТОХАСТИЧЕСКИЙ ГРАДИЕНТНЫЙ СПУСК

Stochastic gradient descent (SGD):

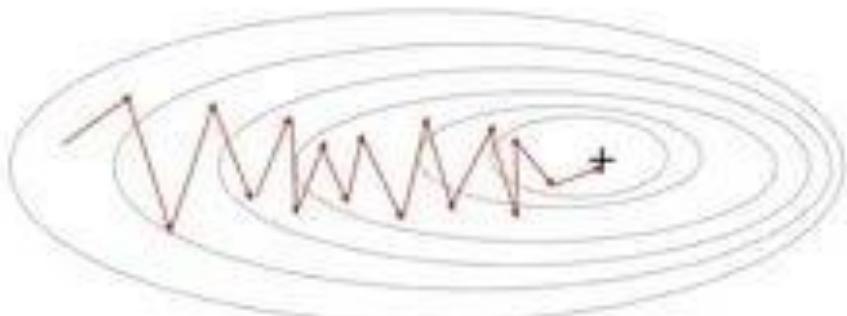
- на каждом шаге выбираем **один случайный объект** и сдвигаемся в сторону антиградиента по этому объекту:

$$w^{(k)} = w^{(k-1)} - \eta_k \cdot \nabla q_{i_k}(w^{(k-1)}),$$

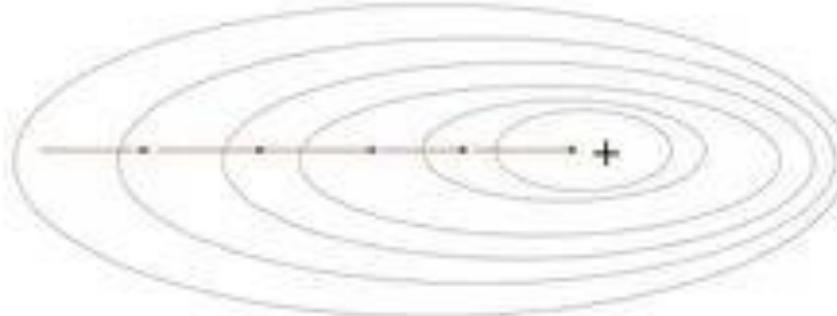
где $\nabla q_{i_k}(w^{(k-1)})$ - градиент функции потерь, вычисленный только по объекту с номером i_k (а не по всей обучающей выборке).

СТОХАСТИЧЕСКИЙ ГРАДИЕНТНЫЙ СПУСК

Stochastic Gradient Descent



Gradient Descent



Если функция $Q(w)$ выпуклая и гладкая, а также имеет минимум в точке w^* , то метод стохастического градиентного спуска при аккуратно подобранном η через некоторое число шагов гарантированно попадет в малую окрестность точки w^* . Однако, сходится метод медленнее, чем обычный градиентный спуск

MINI-BATCH GRADIENT DESCENT

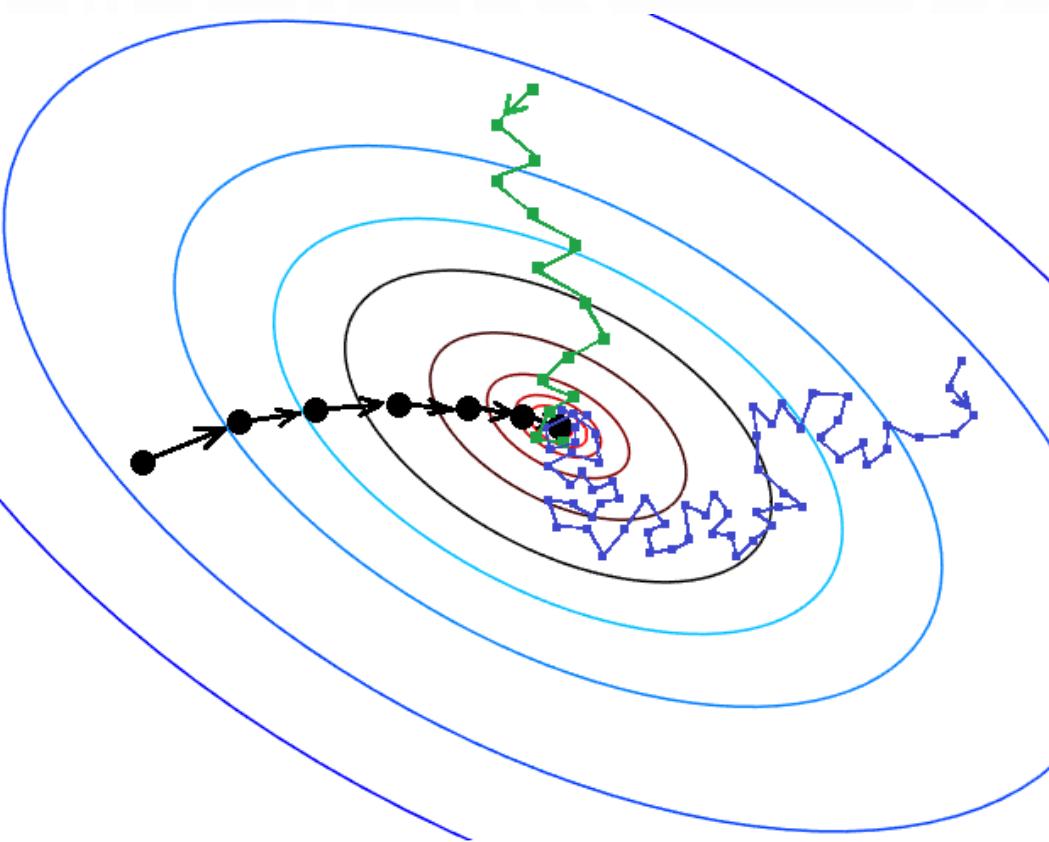
Промежуточное решение между классическим градиентным спуском и стохастическим вариантом.

- Выбираем `batch size` (например, 32, 64 и т.д.). Разбиваем все пары объект-ответ на группы размера `batch size`.
- На i -й итерации градиентного спуска вычисляем $\nabla Q(w)$ только по объектам i -го батча:

$$w^{(k)} = w^{(k-1)} - \eta_k \cdot \nabla Q_i(w^{(k-1)}),$$

где $\nabla Q_i(w^{(k-1)})$ - градиент функции потерь, вычисленный по объектам из i -го батча.

ВАРИАНТЫ ГРАДИЕНТНОГО СПУСКА



Batch GD

- Slowest
- Perfect gradient

Stochastic GD

- Fastest
- Rough-estimate grad

Mini-batch GD

- Compromise