Метрики качества регрессии

Елена Кантонистова

МЕТРИКИ КАЧЕСТВА И ФУНКЦИОНАЛЫ ОШИБКИ В ЗАДАЧАХ РЕГРЕССИИ

МЕТРИКИ КАЧЕСТВА И ФУНКЦИИ ОШИБКИ

- **Функционал (функция) ошибки –** функция, которую минимизируют в процессе обучения модели для нахождения неизвестных параметров (весов).
- **Метрика качества –** функция, которую используют для оценки качества построенной (уже обученной) модели.

МЕТРИКИ КАЧЕСТВА И ФУНКЦИИ ОШИБКИ

- **Функционал (функция) ошибки –** функция, которую минимизируют в процессе обучения модели для нахождения неизвестных параметров (весов).
- **Метрика качества –** функция, которую используют для оценки качества построенной (уже обученной) модели.

Иногда одна и та же функция может использоваться и для обучения модели (функция ошибки), и для оценки качества модели (метрика качества).

ЛИНЕЙНАЯ РЕГРЕССИЯ

Линейная регрессия:

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

Обучение линейной регрессии - минимизация среднеквадратичной ошибки:

$$\frac{1}{l}\sum_{i=1}^{l}(a(x_i) - y_i)^2 \to \min_w$$

СРЕДНЕКВАДРАТИЧНОЕ ОТКЛОНЕНИЕ: MSE (MEAN SQUARED ERROR)

Среднеквадратичное отклонение:

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

СРЕДНЕКВАДРАТИЧНОЕ ОТКЛОНЕНИЕ: MSE (MEAN SQUARED ERROR)

Среднеквадратичное отклонение:

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

Плюсы:

- Позволяет сравнивать модели
- Подходит для контроля качества во время обучения

СРЕДНЕКВАДРАТИЧНОЕ ОТКЛОНЕНИЕ: MSE

Среднеквадратичное отклонение:

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

Плюсы:

- Позволяет сравнивать модели
- Подходит для контроля качества во время обучения

Минусы:

- Плохо интерпретируется, т.к. не сохраняет единицы измерения (если целевая переменная – кг, то MSE измеряется в кг в квадрате)
- Тяжело понять, насколько хорошо данная модель решает задачу, так как MSE не ограничена сверху.

RMSE (ROOT MEAN SQUARED ERROR)

Корень из среднеквадратичной ошибки:

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

Плюсы:

- Все плюсы MSE
- **Сохраняет единицы измерения (в отличие от MSE)**

Минусы:

- Тяжело понять, насколько хорошо данная модель решает задачу, так как RMSE не ограничена сверху.

КОЭФФИЦИЕНТ ДЕТЕРМИНАЦИИ (R^2)

Коэффициент детерминации:

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

где $\overline{y} = \frac{1}{l}\sum_{i=1}^{l} y_i$.

КОЭФФИЦИЕНТ ДЕТЕРМИНАЦИИ (R^2)

Коэффициент детерминации:

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

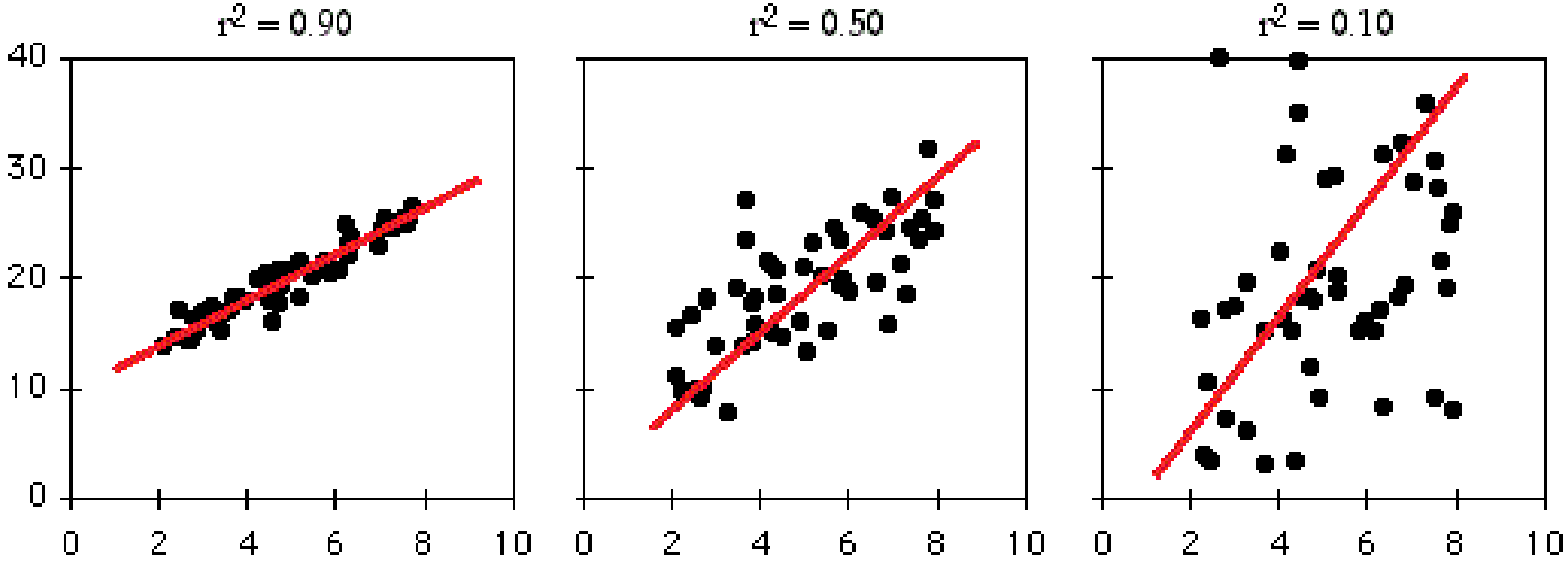
где $\overline{y} = \frac{1}{l}\sum_{i=1}^{l} y_i$.

Коэффициент детерминации это доля дисперсии целевой переменной, объясняемая моделью.

- Чем ближе R^2 к 1, тем лучше модель объясняет данные
- Чем ближе R^2 к 0, тем ближе модель к константному предсказанию
- Отрицательный R^2 говорит о том, что модель плохо решает задачу

КОЭФФИЦИЕНТ ДЕТЕРМИНАЦИИ (R^2)

$$R^2 \leq 1$$



MAE (MEAN ABSOLUTE ERROR)

Средняя абсолютная ошибка:

$$MAE(a, X) = \frac{1}{l}\sum_{i=1}^{l} |\boldsymbol{a(x_i) - y_i}|$$

MAE (MEAN ABSOLUTE ERROR)

Средняя абсолютная ошибка:

$$MAE(a, X) = \frac{1}{l}\sum_{i=1}^{l} |\boldsymbol{a(x_i) - y_i}|$$

Плюсы:

- Менее чувствителен к выбросам, чем MSE

MAE (MEAN ABSOLUTE ERROR)

Средняя абсолютная ошибка:

$$MAE(a, X) = \frac{1}{l}\sum_{i=1}^{l} |\boldsymbol{a(x_i) - y_i}|$$

Плюсы:

- Менее чувствителен к выбросам, чем MSE

Минусы:

- MAE - не дифференцируемый функционал

MSLE (MEAN SQUARED LOGARITHMIC ERROR)

Среднеквадратичная логарифмическая ошибка:

$$\mathrm{M}SLE(a, X) = \frac{1}{l}\sum_{i=1}^{l}\left(\mathbf{\log(a(x_i) + 1) - \log(y + 1)}\right)^{\mathbf{2}}$$

- Подходит для задач с неотрицательной целевой переменной ($y \geq 0$)
- Штрафует за отклонения в порядке величин
- Штрафует заниженные прогнозы сильнее, чем завышенные

MAPE

MAPE – Mean Absolute Percentage Error:

$$MAPE(a, X) = \frac{1}{l}\sum_{i=1}^{l}\frac{|y_i - a(x_i)|}{|y_i|}$$

MAPE измеряет относительную ошибку.

MAPE

$$MAPE(a, X) = \frac{1}{l}\sum_{i=1}^{l}\frac{|y_i - a(x_i)|}{|y_i|}$$

Плюсы:

- $MAPE \geq 0$
- Хорошо интерпретируема: например, MAPE=0.16 означает, что ошибка модели в среднем составляет 16% от фактических значений.

MAPE

$$MAPE(a, X) = \frac{1}{l}\sum_{i=1}^{l}\frac{|\boldsymbol{y_i - a(x_i)}|}{|\boldsymbol{y_i}|}$$

Плюсы:

- $MAPE \geq 0$
- Хорошо интерпретируема: например, MAPE=0.16 означает, что ошибка модели в среднем составляет 16% от фактических значений.

Минусы:

- По-разному относится к недо- и перепрогнозу. Например, если правильный ответ $y = 10$, а прогноз $a(x) = 20$, то ошибка $\frac{|10-20|}{|10|} = \mathbf{1}$, а если ответ $y = 30$, то ошибка $\frac{|30-20|}{|30|} = \frac{1}{3} \approx \mathbf{0.33}$.

SMAPE

SMAPE – *Symmetric Mean Absolute Percentage Error* (симметричный вариант MAPE):

$$SMAPE(a, X) = \frac{1}{l}\sum_{i=1}^{l}\frac{|\boldsymbol{y_i - a(x_i)}|}{(|\boldsymbol{y_i}| + |\boldsymbol{a(x_i)}|)/\boldsymbol{2}}$$

SMAPE – попытка сделать симметричным прогноз (то есть дать одинаковую ошибку для недо- и перепрогноза).

SMAPE

SMAPE – *Symmetric Mean Absolute Percentage Error* (симметричный вариант MAPE):

$$SMAPE(a, X) = \frac{1}{l}\sum_{i=1}^{l}\frac{|y_i - a(x_i)|}{(|y_i| + |a(x_i)|)/2}$$

SMAPE – попытка сделать симметричным прогноз (то есть дать одинаковую ошибку для недо- и перепрогноза).

<u>Проверим:</u>

Пусть правильный ответ $y = 10$, а прогноз $a(x) = 20$, то ошибка $\frac{|10-20|}{|10+20|/2} = \frac{2}{3} \approx 0.67$, а если ответ $y = 30$, то ошибка $\frac{|30-20|}{|30+20|/2} = \frac{2}{5} = 0.4$.

SMAPE

SMAPE – попытка сделать симметричным прогноз (то есть дать одинаковую ошибку для недо- и перепрогноза).

<u>Проверим:</u>

Пусть правильный ответ $\boldsymbol{y = 10}$, а прогноз $\boldsymbol{a(x) = 20}$, то ошибка $\frac{|10-20|}{|10+20|/2} = \frac{2}{3} \approx \mathbf{0.67}$, а если ответ $\boldsymbol{y = 30}$, то ошибка $\frac{|30-20|}{|30+20|/2} = \frac{2}{5} = \mathbf{0.4}$.

Ошибки стали меньше отличаться друг от друга, но всё-таки не равны.

SMAPE

SMAPE – попытка сделать симметричным прогноз (то есть дать одинаковую ошибку для недо- и перепрогноза).

“Сейчас уже в среде прогнозистов сложилось более-менее устойчивое понимание, что SMAPE не является хорошей ошибкой. Тут дело не только в завышении прогнозов, но ещё и в том, что наличие прогноза в знаменателе позволяет манипулировать результатами оценки.” (см. источник)

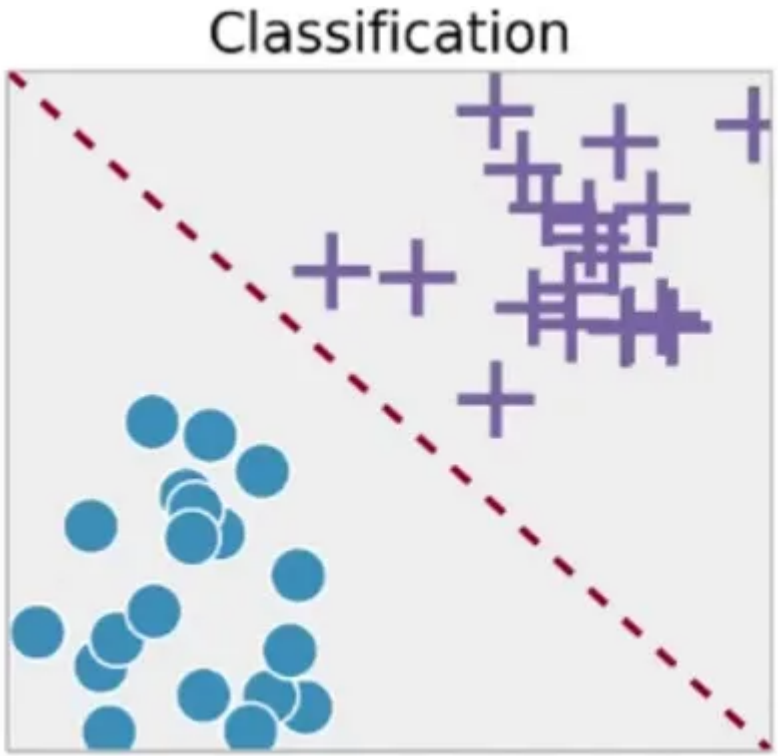
Линейная классификация

КЛАССИФИКАЦИЯ

Хотим предсказывать классы – как?

- Линейная регрессия:

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



КЛАССИФИКАЦИЯ

Хотим предсказывать классы.

- Линейная регрессия:

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

- Линейный классификатор:

$$a(x, w) = sign(w_0 + w_1 x_1 + w_2 x_2 + \ \ldots)$$

ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

Хотим предсказывать не только классы, но и ***вероятности классов***.

- Линейная регрессия:

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

- Линейный классификатор:

$$a(x, w) = sign(w_0 + w_1 x_1 + w_2 x_2 + \ \ldots)$$

- Логистическая регрессия:

$$\boldsymbol{a(x, w) = \sigma(w_0 + w_1 x_1 + w_2 x_2 + \ \ldots) = \sigma(w, x),}$$

где $\sigma(z) = \frac{1}{1+e^{-z}}$ - сигмоида (логистическая функция)

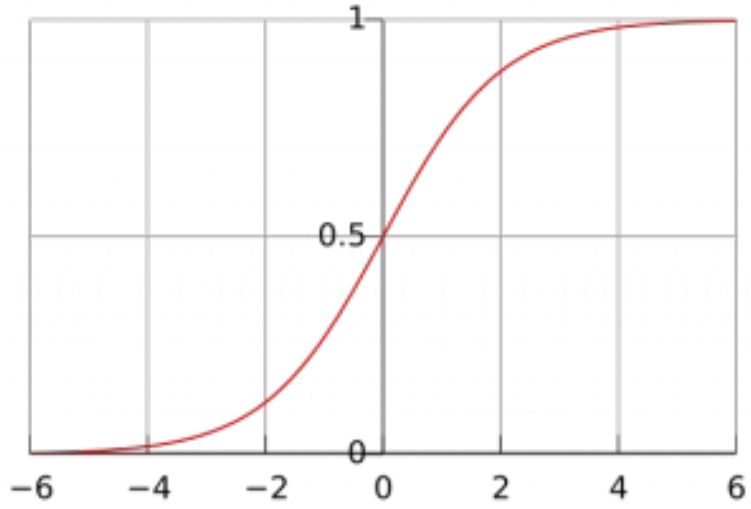
ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

Хотим предсказывать не классы, а вероятности классов.

- Логистическая регрессия: $\boldsymbol{a(x, w) = \sigma(w, x)}$,

где $\sigma(z) = \frac{1}{1+e^{-z}}$ - сигмоида (логистическая функция),

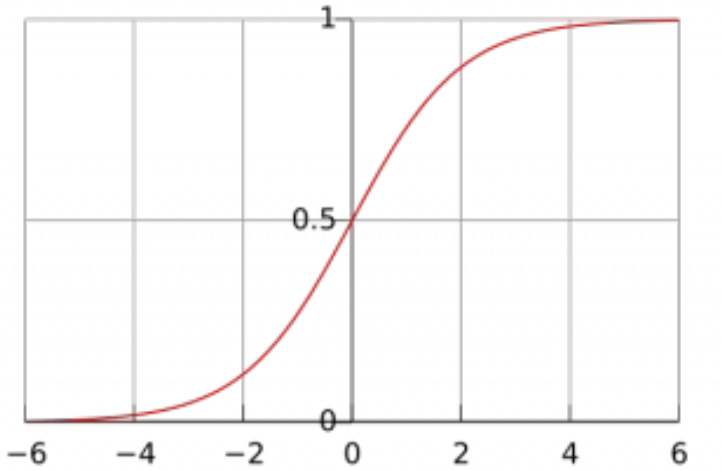
$\sigma(z) \in (0; 1)$.



Логистическая регрессия: $\boldsymbol{a(x, w) = \frac{1}{1+e^{-(w,x)}}}$

РАЗДЕЛЯЮЩАЯ ГРАНИЦА

Предсказываем $y = +1$, если $a\,(x, w) \geq 0.5$.



$a(x, w) = \sigma(w, x) \geq 0.5$, если $(w, x) \geq 0$.

Получаем, что

- $y = +1$ при $(w, x) \geq 0$
- $y = -1$ при $(w, x) < 0$,

т.е. $(w, x) = w_1 x_1 + w_2 x_2 + \cdots = 0$ – разделяющая гиперплоскость.

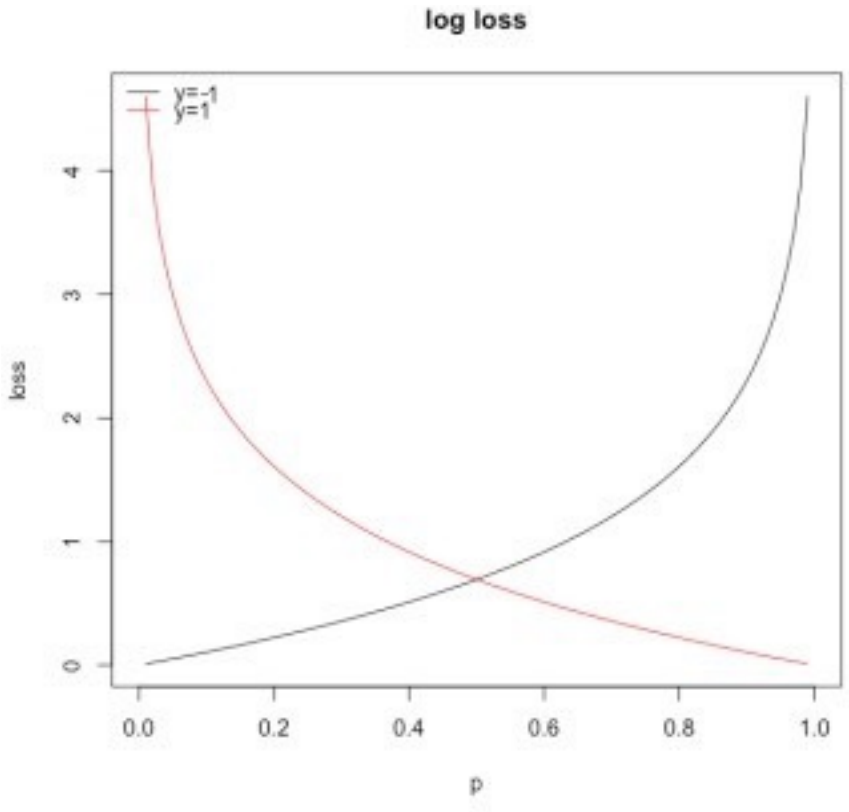
ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

Логистическая регрессия - это линейный классификатор!

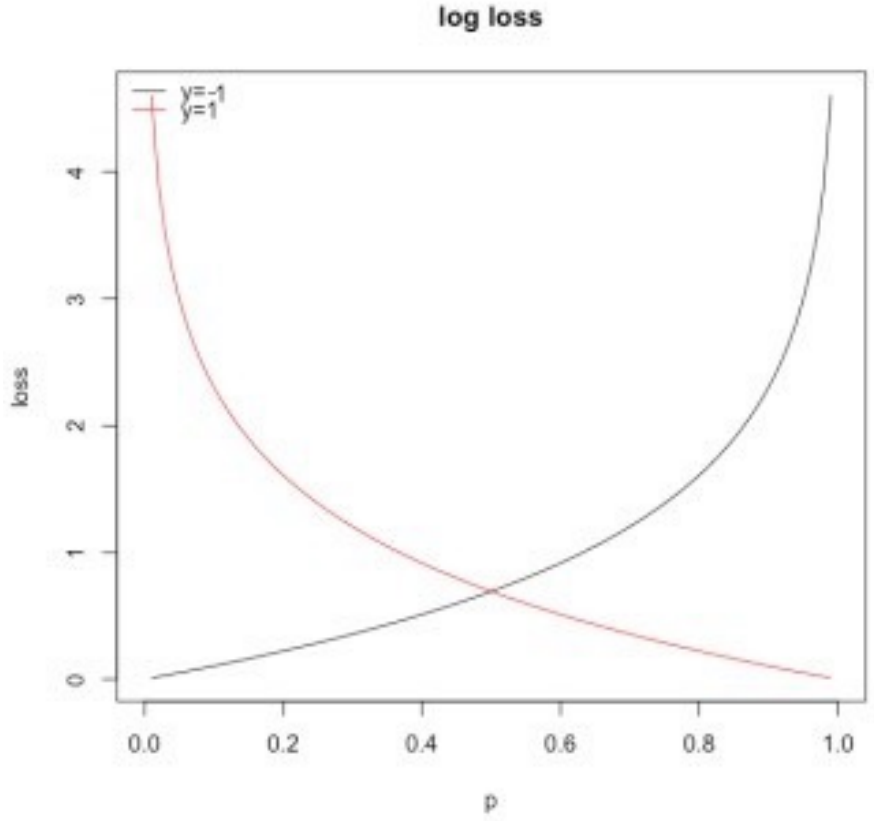
ФУНКЦИЯ ПОТЕРЬ ЛОГИСТИЧЕСКОЙ РЕГРЕССИИ

Возьмем логистическую функцию потерь (log-loss):

$$Q(w) = -\sum_{i=1}^{l}([y_i = +1] \cdot \log(a(x_i, w)) + [y_i = -1] \cdot \log(1 - a(x_i, w)))$$



ЛОГИСТИЧЕСКАЯ ФУНКЦИЯ ПОТЕРЬ



- если $a(x, w) = 1$ и y $= +1$, то штраф $L(a, y) = 0$
- если $a(x, w) \to 0$, а y $= +1$, то штраф $L(a, y) \to +\infty$

МЕТОД БОРЬБЫ С ПЕРЕОБУЧЕНИЕМ: РЕГУЛЯРИЗАЦИЯ

Большие значения параметров (весов) модели $\boldsymbol{w}$ *– признак переобучения.*

Решение проблемы – *регуляризация*.

Будем минимизировать регуляризованный функционал ошибки:

$$\boldsymbol{Q_{alpha}}(\boldsymbol{w}) = \boldsymbol{Q}(\boldsymbol{w}) + \boldsymbol{\alpha} \cdot \boldsymbol{R}(\boldsymbol{w}) \rightarrow \mathbf{min},$$

где $R(w)$ - регуляризатор.

РЕГУЛЯРИЗАЦИЯ

- **Регуляризация штрафует за слишком большие веса.**

Наиболее используемые регуляризаторы:

- L_2-регуляризатор: $R(w) = \left|\left|w\right|\right|_2 = \sum_{i=1}^{d} w_i^2$
- L_1-регуляризатор: $R(w) = \left|\left|w\right|\right|_1 = \sum_{i=1}^{d} |w_i|$

Метрики качества классификации

Пример: предсказание модели

id	Предсказанная вероятность	Правильный ответ	Предсказанный класс
1	0.6	-1	1
2	0.8	1	1
3	0.3	-1	-1
4	0.55	-1	1
5	0.1	-1	-1
6	0.96	1	1
7	0.33	1	-1
8	0.2	-1	-1
9	0.14	-1	-1
10	0.88	1	1

Accuracy

- **Accuracy** – это доля правильных ответов алгоритма

Accuracy = 0.7

id	Предсказанная вероятность	Правильный ответ	Предсказанный класс
1	0.6	-1	**1**
2	0.8	1	1
3	0.3	-1	-1
4	0.55	-1	**1**
5	0.1	-1	-1
6	0.96	1	1
7	0.33	1	**-1**
8	0.2	-1	-1
9	0.14	-1	-1
10	0.88	1	1

Accuracy

- 1000 объектов:

950 – не мошенники (класс 0)
50 – мошенники (класс +1)

- Модель: a(x) = 0

Accuracy?

Accuracy

- 1000 объектов:

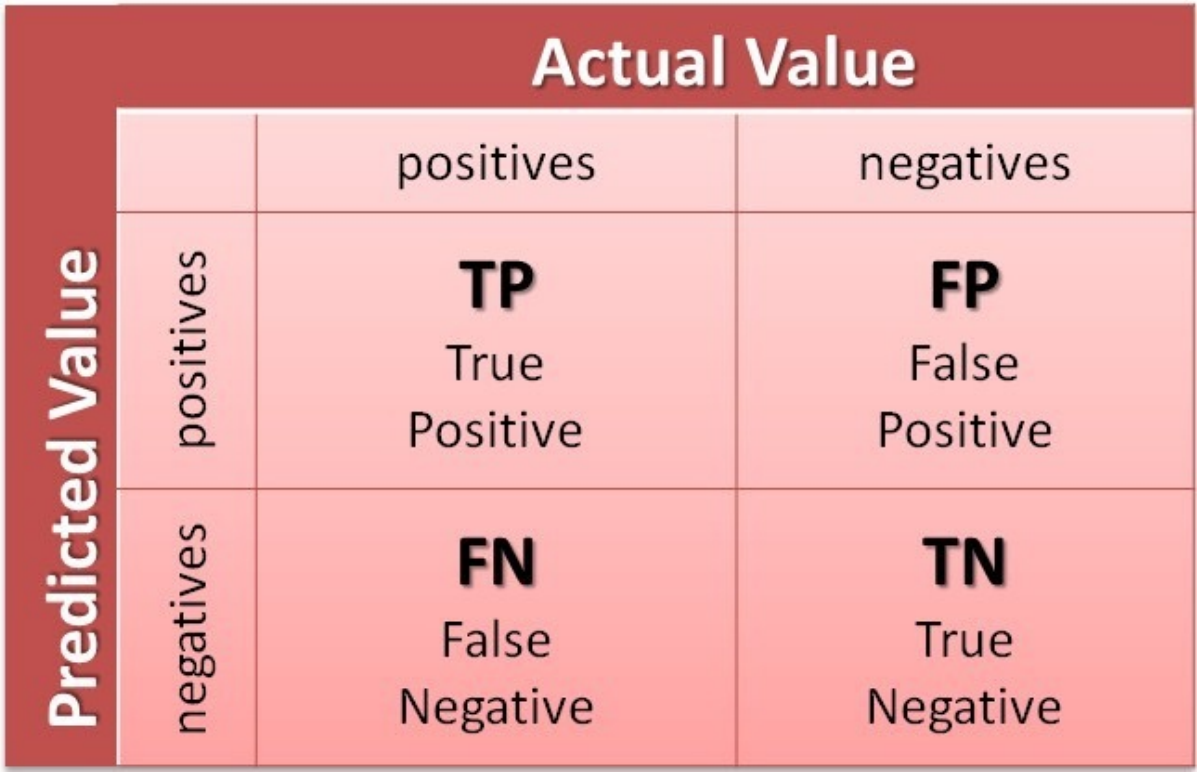
950 – не мошенники (класс 0)
50 – мошенники (класс +1)

- Модель: a(x) =0

Accuracy = 0.95

- *Если классы несбалансированы, то accuracy не надо использовать!*
- *Метрика не показывает какие классы между собой путаем*

Матрица ошибок



Пример: кредитный скоринг

Модель 1: одобряет 100 кредитов

- 80 кредитов вернули
- 20 кредитов не вернули

Модель 2: одобряет 50 кредитов

- 48 кредитов вернули
- 2кредита не вернули

На тестовой выборке, где 100 вернули, 100 не вернули

Какая модель лучше?

Точность (precision)

Точность показывает, насколько можно доверять классификатору в случае если он выдает положительный класс $a(x) = +1$

$$precision = \frac{TP}{TP + FP}$$

Predicted Value		Actual Value: positives	Actual Value: negatives
	positives	**TP** True Positive	**FP** False Positive
	negatives	**FN** False Negative	**TN** True Negative

Точность (precision)

Точность показывает, насколько можно доверять классификатору в случае если он выдает положительный класс $a(x) = +1$

$a_1(x)$: $precision = 0.8$ $a_2(x)$: $precision = 0.96$

	$y = 1$ Могут вернуть	$y = -1$ Не могут вернуть
$a(x) = 1$ Получили кредит	80	20
$a(x) = -1$ Не получили кредит	20	80

	$y = 1$ Могут вернуть	$y = -1$ Не могут вернуть
$a(x) = 1$ Получили кредит	48	2
$a(x) = -1$ Не получили кредит	52	98

Полнота (recall)

Полнота показывает как много объектов положительного класса нашел классификатор

$$precision = \frac{TP}{TP + FN}$$

		Actual Value	
		positives	negatives
Predicted Value	positives	**TP** True Positive	**FP** False Positive
	negatives	**FN** False Negative	**TN** True Negative

Полнота (recall)

Полнота показывает как много объектов положительного класса нашел классификатор

$a_1(x)$: $recall$ = 0.8

	$y = 1$ Могут вернуть	$y = -1$ Не могут вернуть
$a(x) = 1$ Получили кредит	80	20
$a(x) = -1$ Не получили кредит	20	80

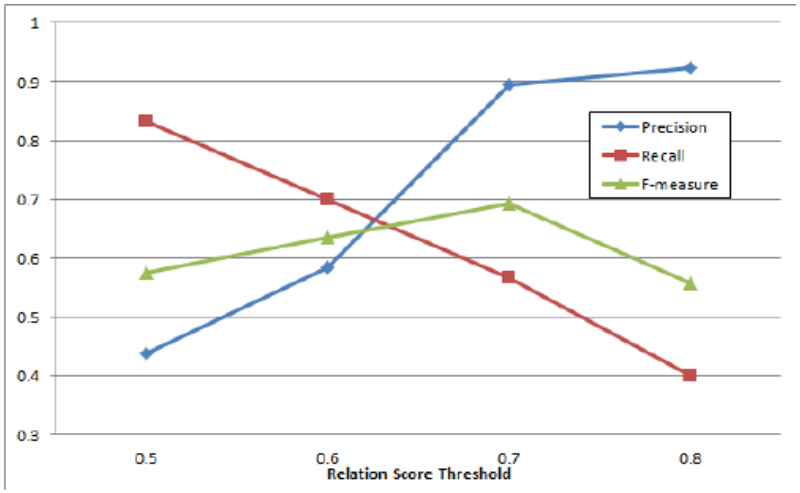
$a_2(x)$: $recall$ = 0.48

	$y = 1$ Могут вернуть	$y = -1$ Не могут вернуть
$a(x) = 1$ Получили кредит	48	2
$a(x) = -1$ Не получили кредит	52	98

F-мера

F-мера (F1-score) - среднее гармоническое точности и полноты

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

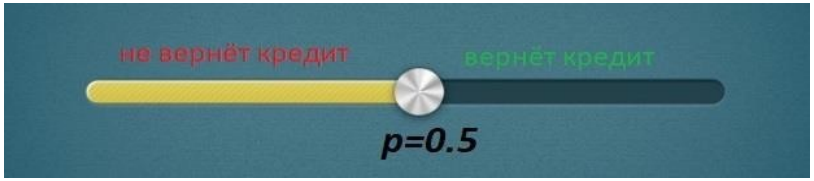


Регулируем точность и полноту

Пусть p(x) - уверенность классификатора в том, что объект x относится к классу +1, p(x) лежит на отрезке [0;1].

Обычно

- если p(x) > 0.5, то мы относим объект к положительному классу
- а иначе – к отрицательному



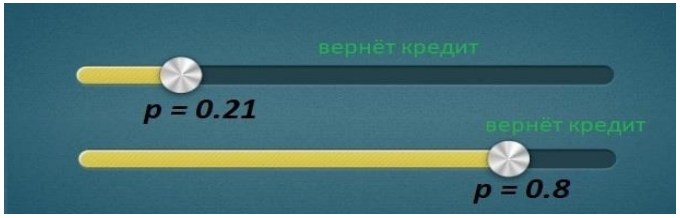
Регулируем точность и полноту

Пусть p(x) - уверенность классификатора в том, что объект x относится к классу +1, p(x) лежит на отрезке [0;1].

Обычно

- если p(x) > 0.5, то мы относим объект к положительному классу
- а иначе – к отрицательному

Можно изменять этот порог, то есть вместо 0.5 брать другое число из отрезка [0;1].

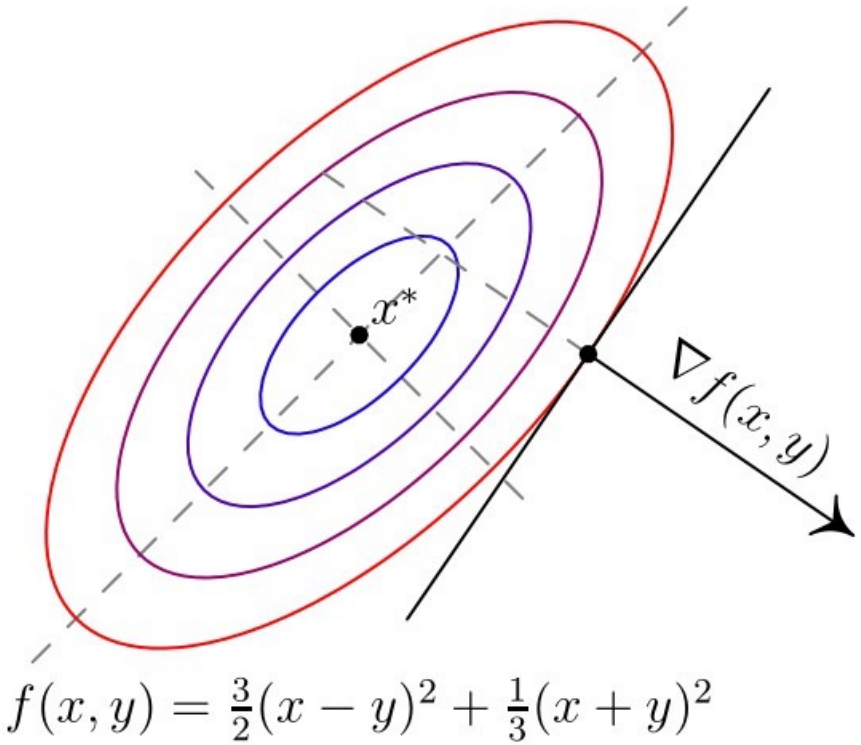


Градиентный спуск

ТЕОРЕМА О ГРАДИЕНТЕ

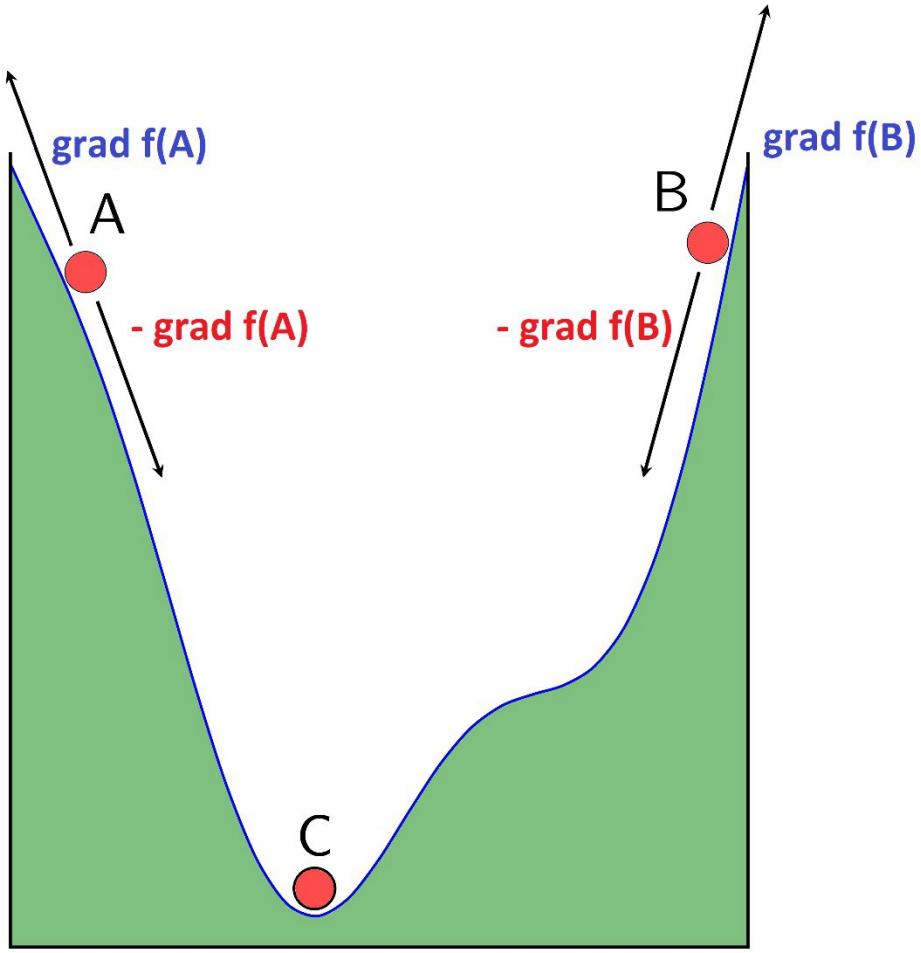
Теорема. Градиент – это вектор, в направлении которого функция быстрее всего растёт.

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



ТЕОРЕМА О ГРАДИЕНТЕ

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

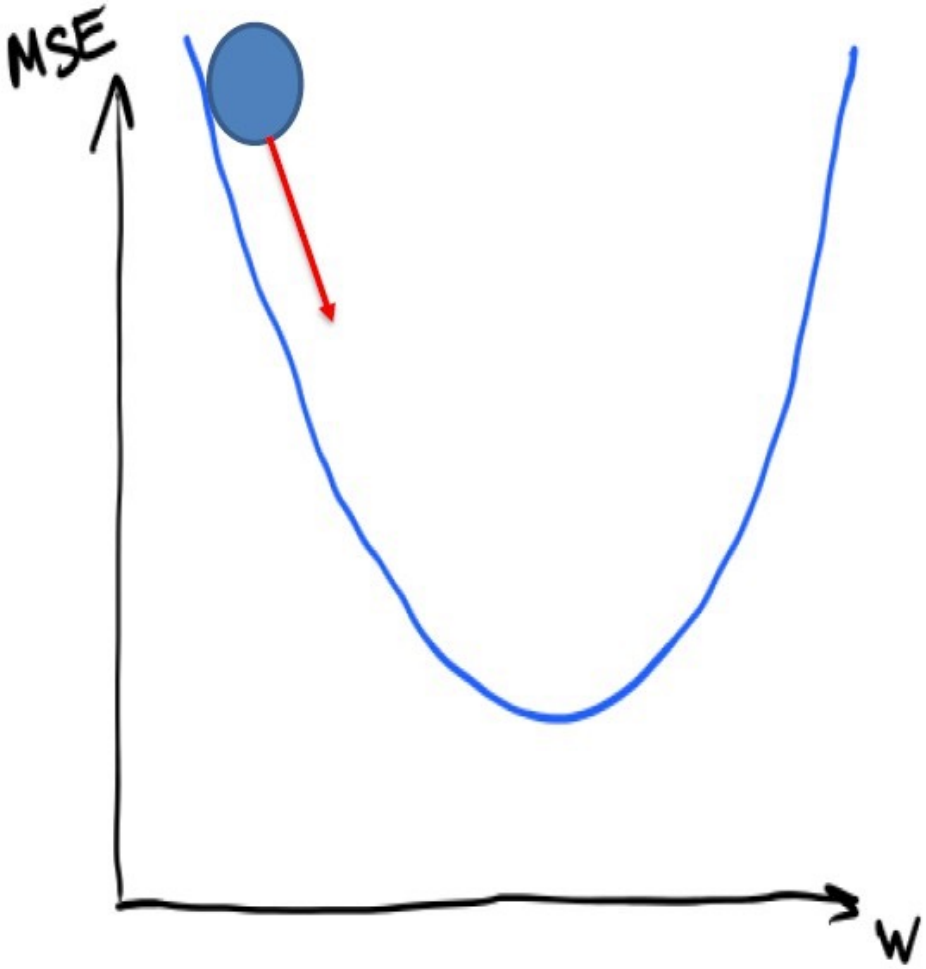


МЕТОД ГРАДИЕНТНОГО СПУСКА

- Наша задача при обучении модели – найти такие веса $\boldsymbol{w}$, на которых достигается минимум функции ошибки.
- В простейшем случае, если ошибка среднеквадратичная, то её график – это парабола.

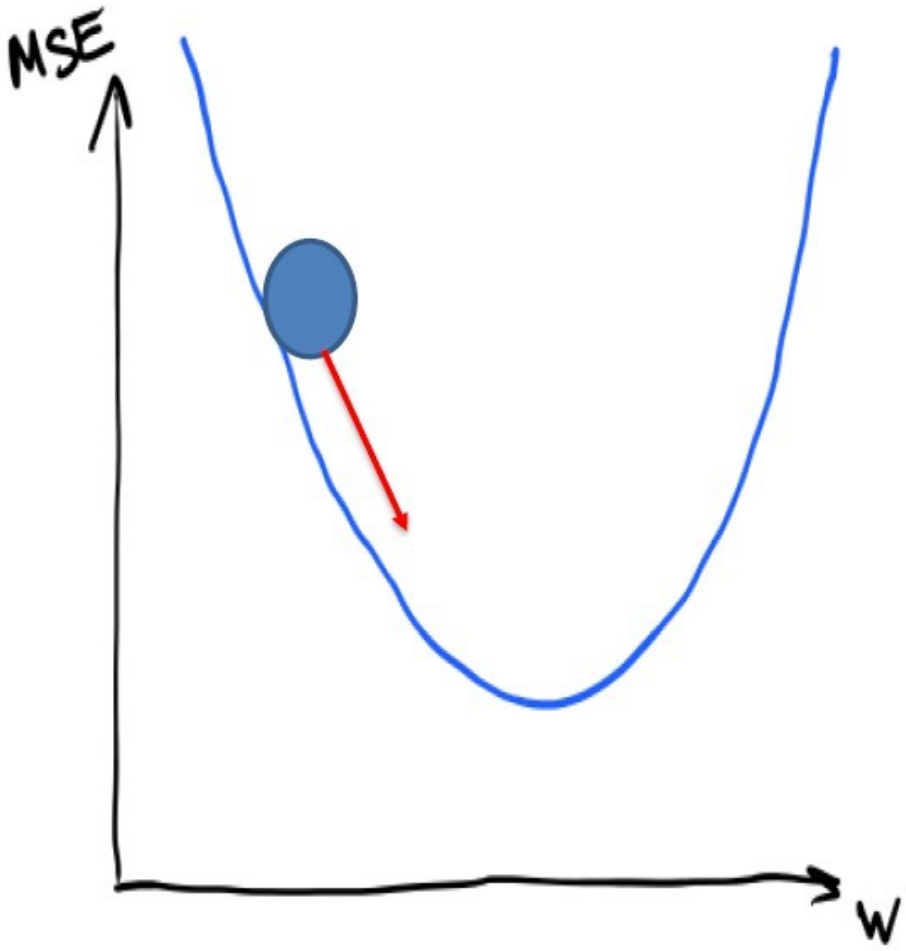
МЕТОД ГРАДИЕНТНОГО СПУСКА

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



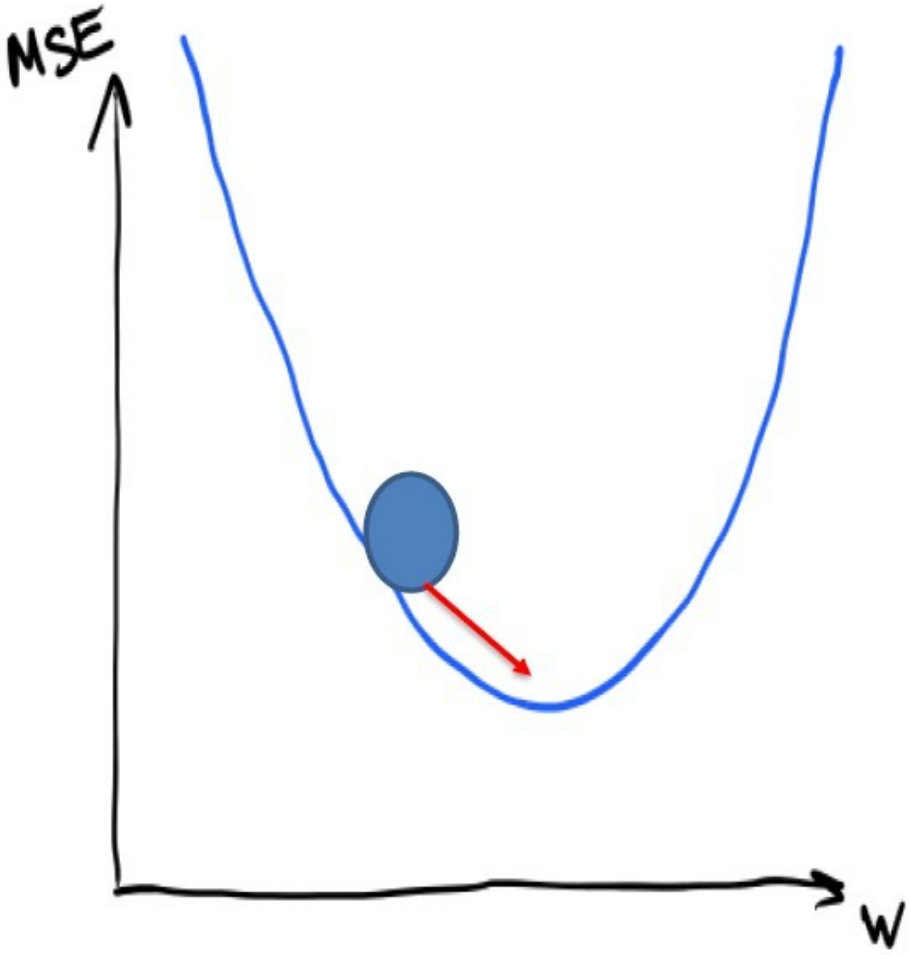
МЕТОД ГРАДИЕНТНОГО СПУСКА

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



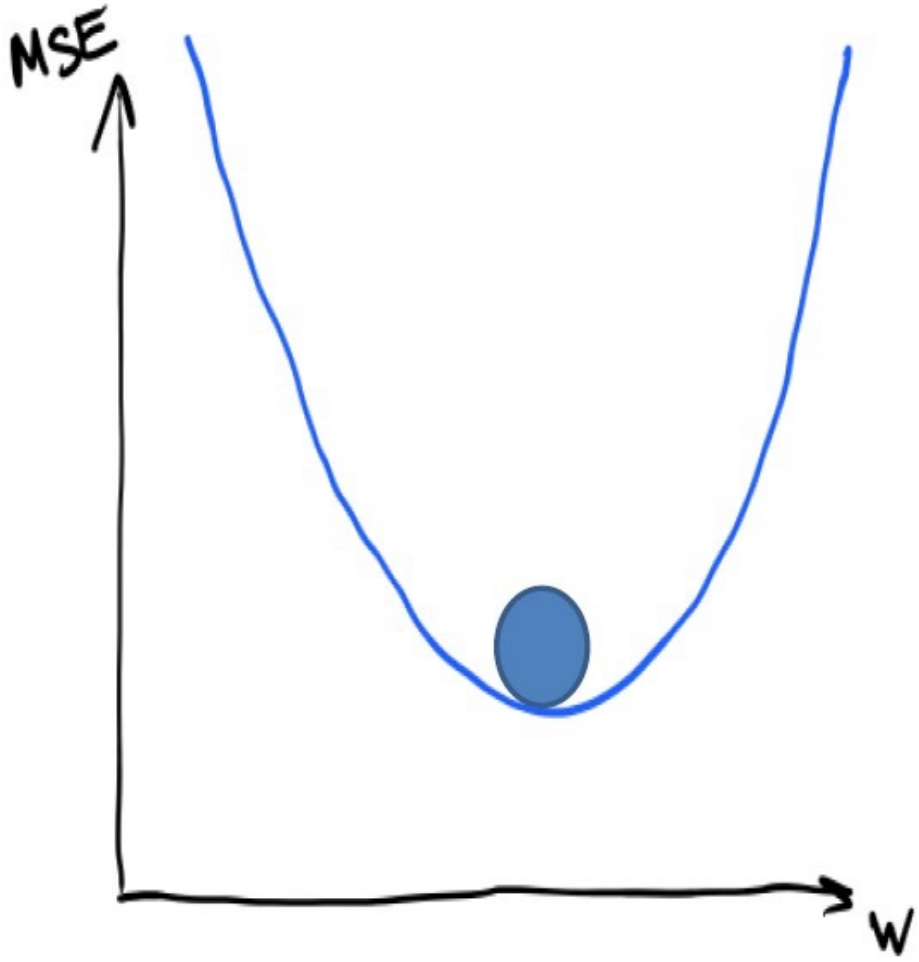
МЕТОД ГРАДИЕНТНОГО СПУСКА

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

- Наша задача при обучении модели – найти такие веса $\boldsymbol{w}$, 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 $\boldsymbol{grad\ Q}$ *или* $\boldsymbol{\nabla Q}$.

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):

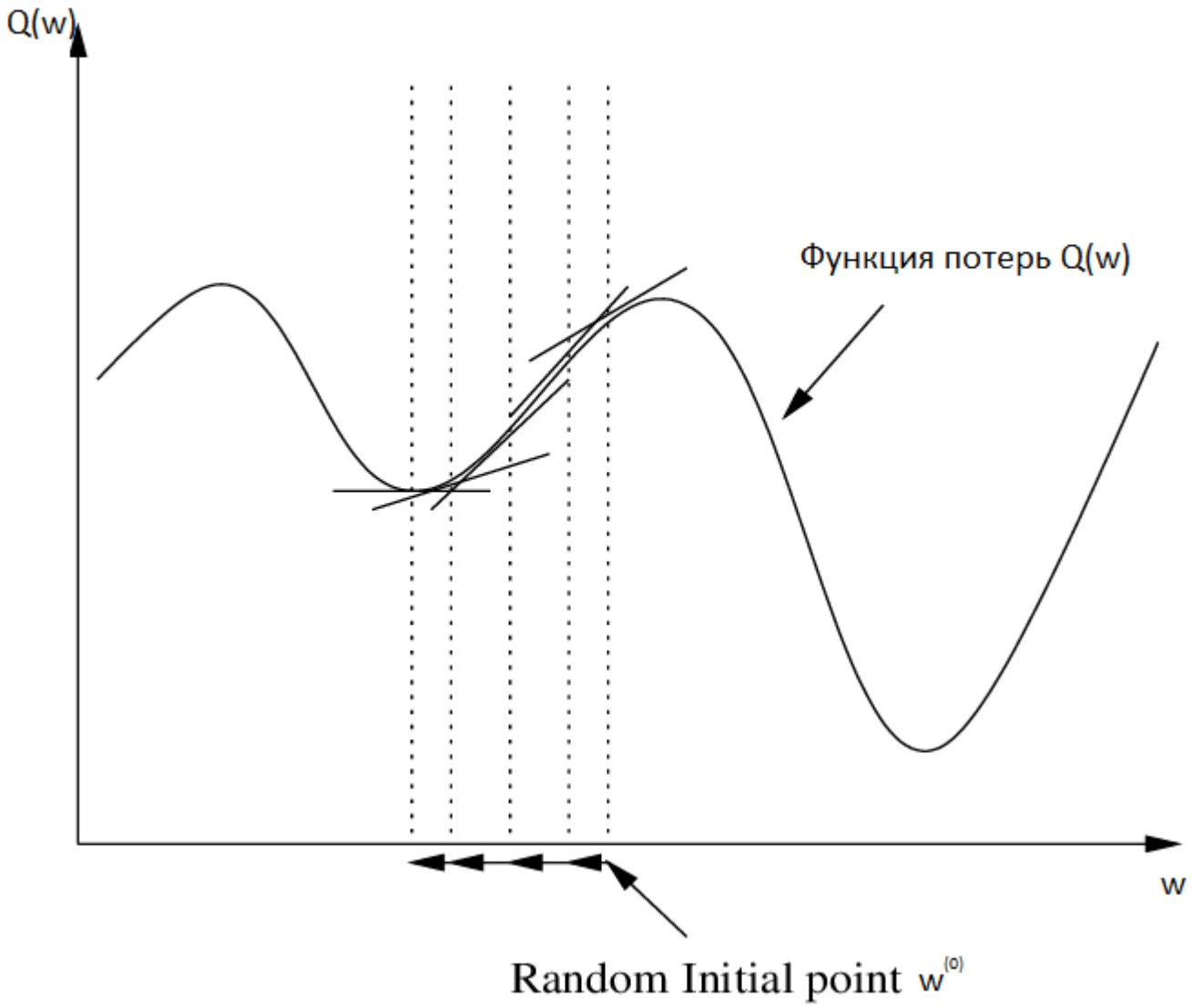
Пусть у нас только один вес - w.

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

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

МЕТОД ГРАДИЕНТНОГО СПУСКА



МЕТОД ГРАДИЕНТНОГО СПУСКА

Формулу для обновления весов можно записать в векторном виде:

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

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

МЕТОД ГРАДИЕНТНОГО СПУСКА

Формулу для обновления весов можно записать в векторном виде:

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

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

В формулу обычно добавляют параметр $\boldsymbol{\eta}$ – величина градиентного шага (learning rate). Он отвечает за скорость движения в сторону антиградиента:

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

ВАРИАНТЫ ИНИЦИАЛИЗАЦИИ ВЕСОВ

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

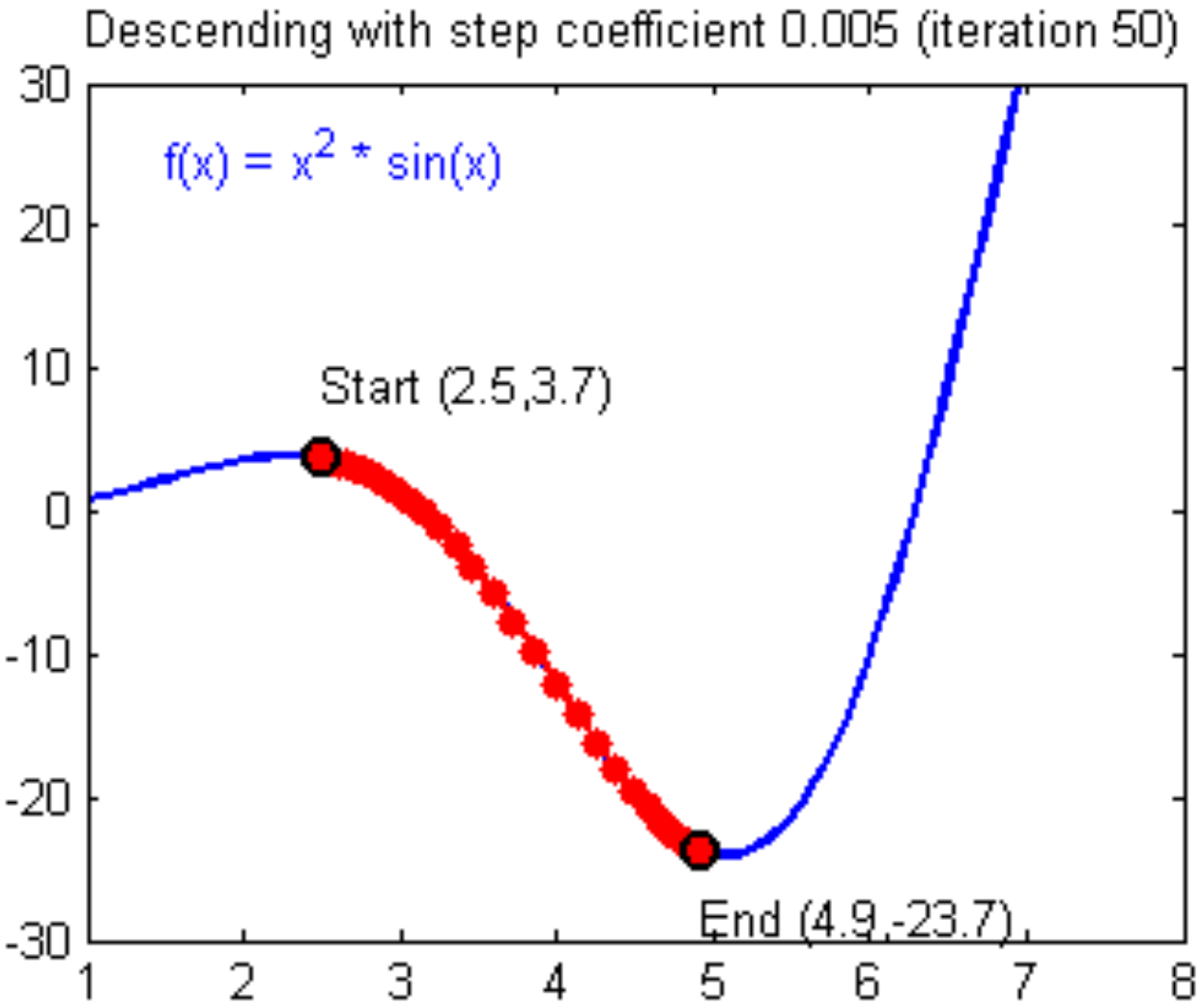
$$w_j := random(-\varepsilon, \varepsilon)$$

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

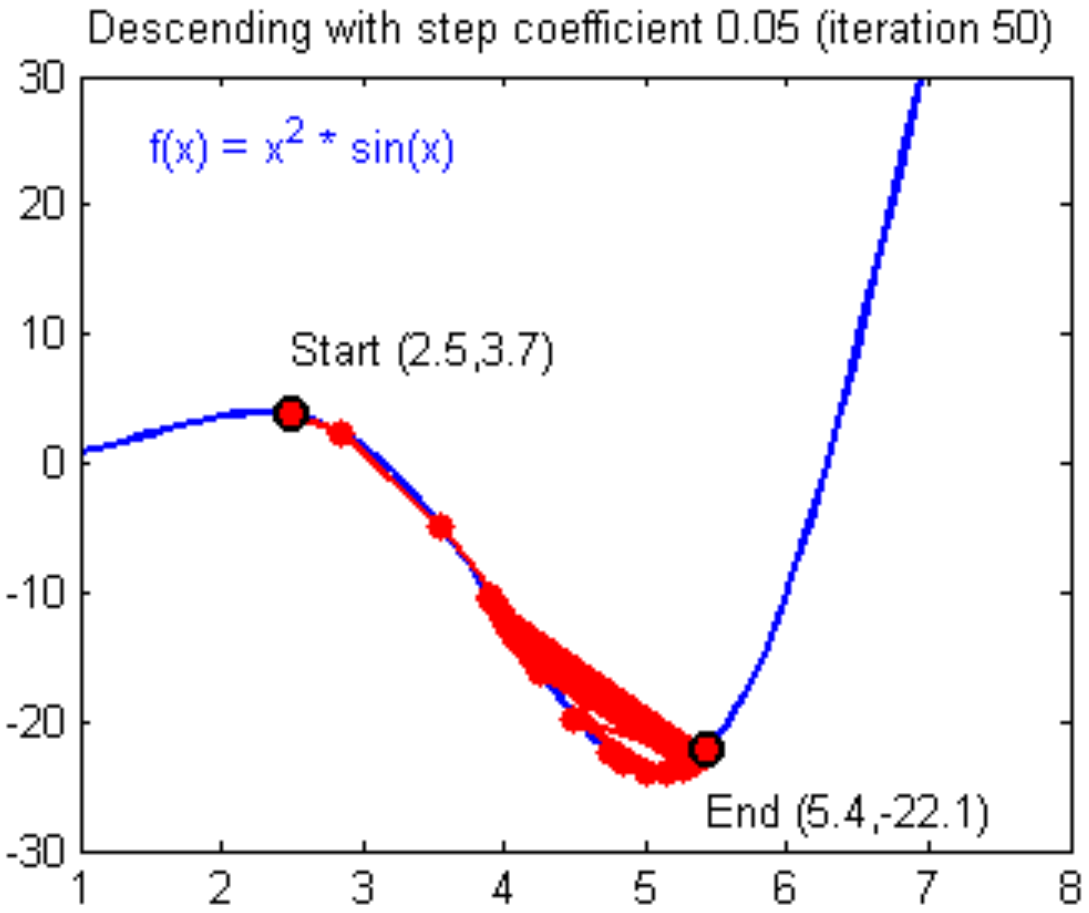
КРИТЕРИИ ОСТАНОВА

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

ГРАДИЕНТНЫЙ СПУСК



ПРОБЛЕМА ВЫБОРА ГРАДИЕНТНОГО ШАГА



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

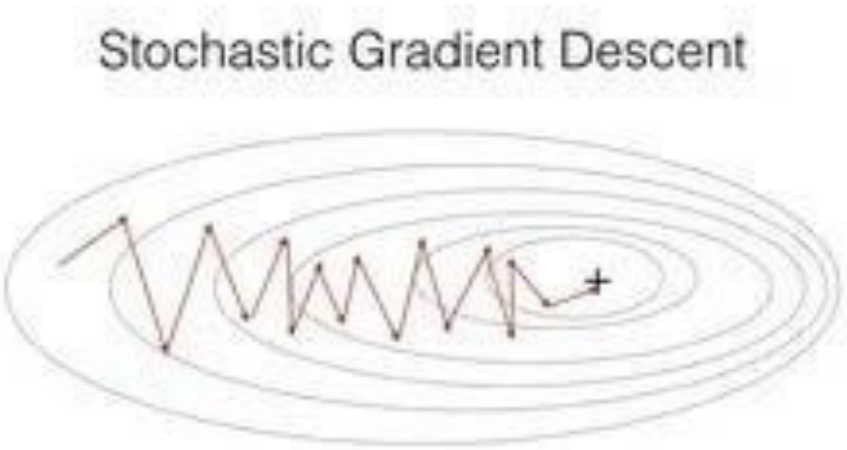
Stochastic gradient descent (SGD):

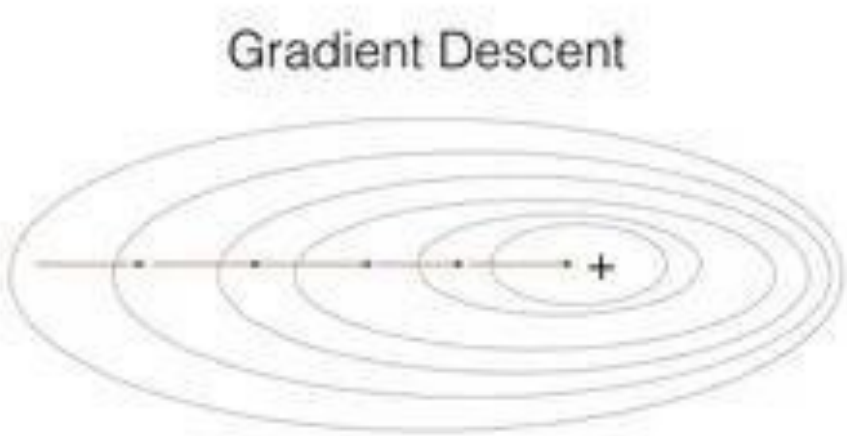
- На каждом шаге выбираем ***один случайный объект*** и сдвигаемся в сторону антиградиента по этому объекту:

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

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

СТОХАСТИЧЕСКИЙ ГРАДИЕНТНЫЙ СПУСК





MINI-BATCH GRADIENT DESCENT

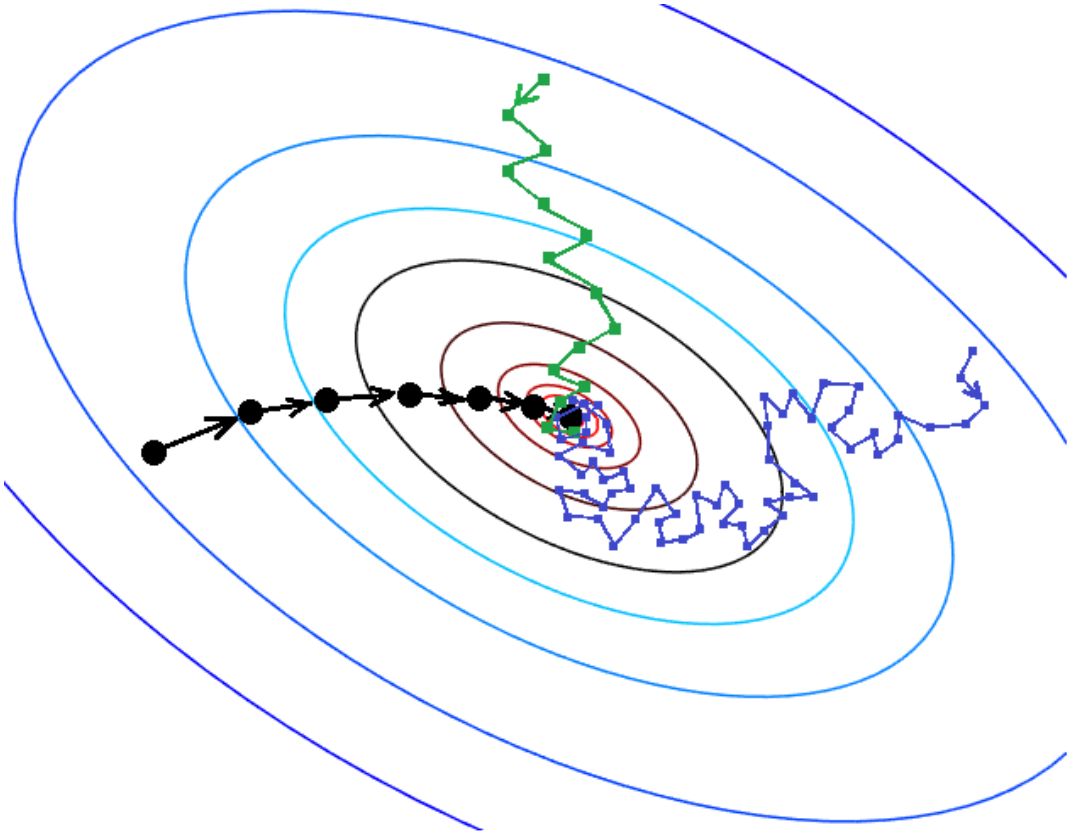
Промежуточное решение между классическим градиентным спуском и стохастическим вариантом.

- 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:

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

где $\nabla Q_i\left(w^{(k-1)}\right)$ - 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