Методы классификации и регрессии.

Кантонистова Елена

elena.kantonistova@yandex.ru

5 декабря 2017

План лекции

1. ROC-AUC
2. Решающие деревья
3. Решающий лес
4. Методы отбора признаков
5. Линейная регрессия

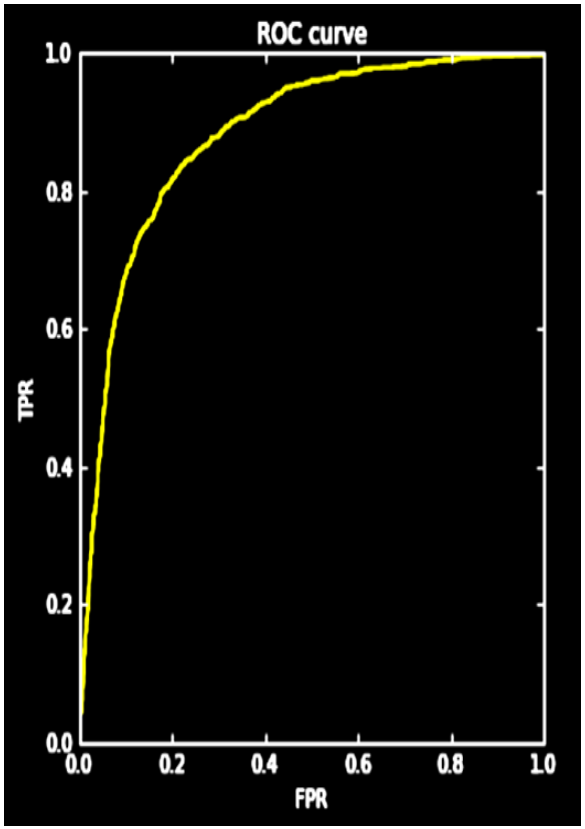
Будем решать задачу двухклассовой классификации. Каждый объект принадлежит классу 0 или 1. Алгоритм в качестве ответа выдает вероятность принадлежности объекта классу 1.

ROC-AUC - метрика качества бинарной классификации для таких алгоритмов.

Определение. Порог вероятности - это значение вероятности, начиная с которого мы относим все объекты к классу 1 (например, если порог вероятности 0.3, то все объекты с вероятностью ≥ 0.3 мы относим к классу 1).

Пример

- ROC-кривая - это кривая, отражающая качество классификации для различных значений порога вероятности.
- Каждая точка на этой кривой - это качество классификации для фиксированного порога вероятности.



ROC-AUC

id	оценка	класс
1	0.5	0
2	0.1	0
3	0.2	0
4	0.6	1
5	0.2	1
6	0.3	1
7	0.0	0

Табл. 1

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

Табл. 2

id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

- Упорядочим объекты по убыванию предсказанных вероятностей.
- В идеале столбец "класс"тоже станет упорядочен (сначала идут 1, а потом 0)

ROC-AUC

- Пара объектов, один из которых принадлежит классу 1, а второй - классу 0, *упорядочена правильно*, если предсказанная вероятность для объекта класса 1 больше, чем вероятность для класса 0.

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

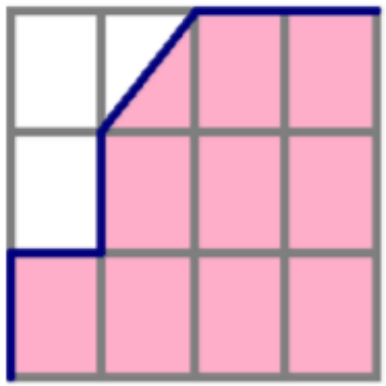
Зеленым обозначена правильно упорядоченная пара объектов, красным - неправильно упорядоченная пара объектов.

ROC-AUC

- Алгоритм тем лучше, чем более качественно он упорядочивает объекты.
- Можно измерять качество алгоритма как долю правильно упорядоченных пар объектов. AUC - площадь под ROC-кривой - и есть эта доля.
- AUC для данного примера равен $9.5/12 \approx 0.79$.

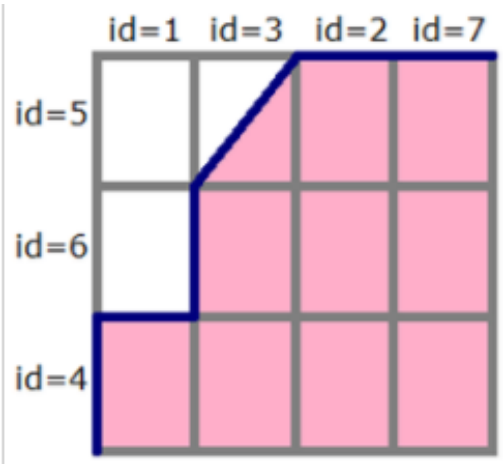
Построение ROC-кривой

- Пользуемся упорядоченными по убыванию вероятности объектами.
- Стартуем из точки $(0, 0)$. Если значение метки класса в просматриваемой строке 1, то делаем шаг вверх; если 0, то делаем шаг вправо.



Интерпретация ROC-кривой

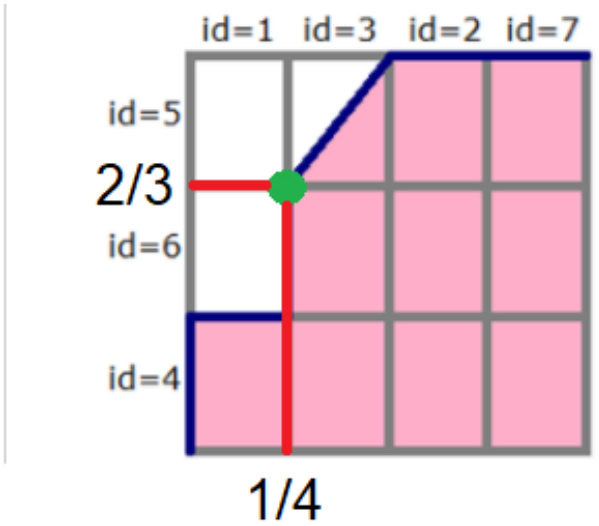
Каждый закрашенный блок на рисунке соответствует паре (объект класса 1, объект класса 0), для которой наш алгоритм правильно предсказал порядок (объект класса 1 имеет предсказанную вероятность выше, чем объект класса 0), незакрашенный блок – паре, на которой ошибся.



Отсюда видно, что площадь под ROC-кривой - это доля пар, для которых алгоритм правильно предсказал порядок.

Выбор порога по ROC-кривой

Выбор порога определяется не только кривой, но и требованиями задачи (ошибка в какую сторону для нас менее приемлема).



Например, если в качестве порога взять $p = 0.3$ (см.таблицу с упорядоченными данными), то:

- 1/4 - доля неверно классифицированных точек класса 0
- 2/3 - доля верно классифицированных точек класса 1

Выбор оптимального порога

- При выборе порога стараемся уменьшить первое число и увеличить второе, то есть, грубо говоря, выбрать точку на ROC-кривой, которая находится ближе всего к левому верхнему углу квадрата.
- Опираемся на требования задачи, т.е. какая из ошибок для нас менее страшна.

Дерево решений

Дерево решений - это дерево. На нём есть метки:

- В узлах, не являющиеся листьями: атрибуты (условия), по которым различаются случаи
- В листьях: значения целевой функции
- На рёбрах: значения атрибута, из которого исходит ребро

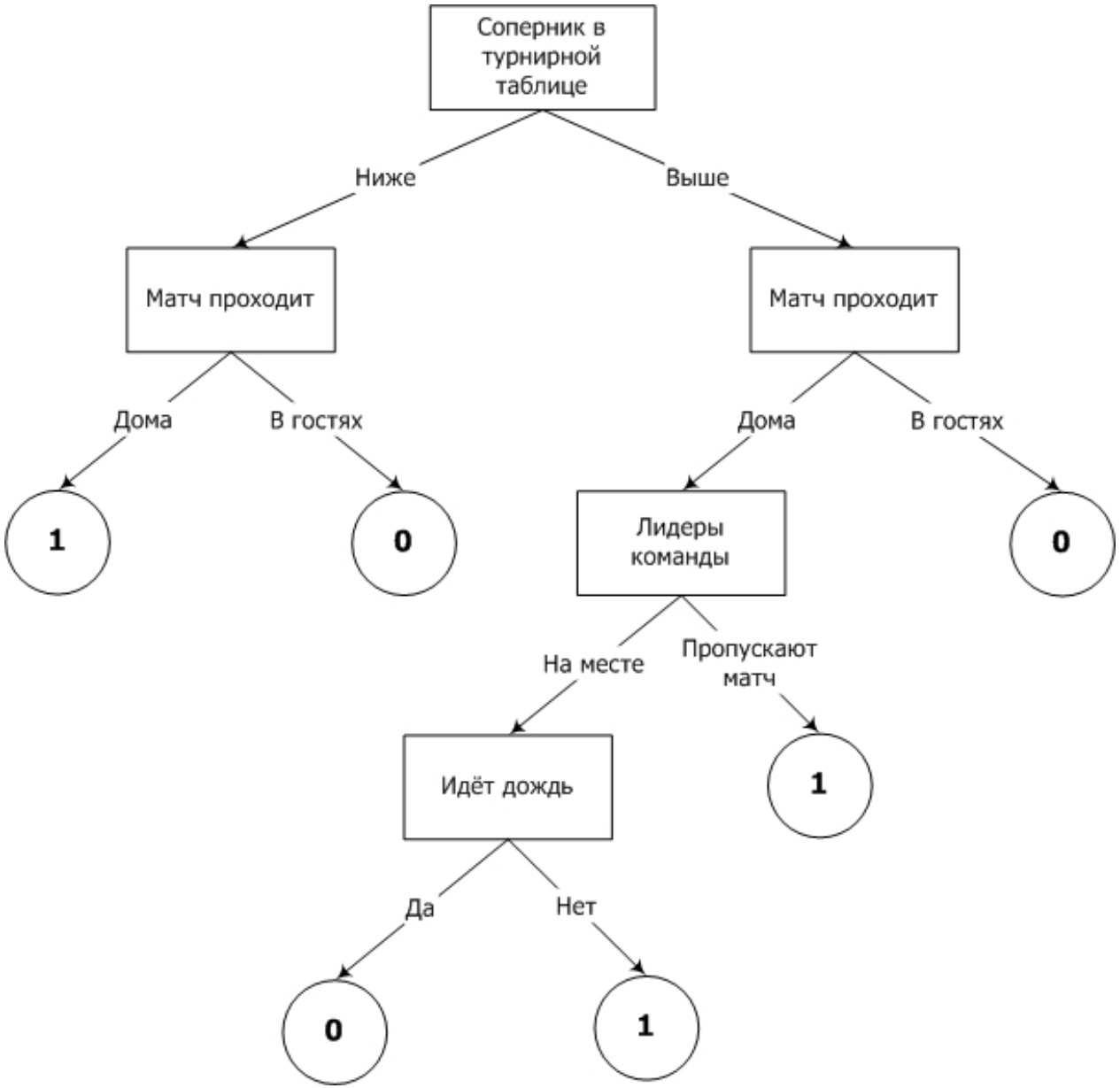
Чтобы классифицировать новый случай, нужно спуститься по дереву до листа и выдать соответствующее значение.

Пример

Таблица 1: Как играет «Зенит».

Соперник	Играем	Лидеры	Дождь	Победа
Выше	Дома	На месте	Да	Нет
Выше	Дома	На месте	Нет	Да
Выше	Дома	Пропускают	Нет	Да
Ниже	Дома	Пропускают	Нет	Да
Ниже	В гостях	Пропускают	Нет	Нет
Ниже	Дома	Пропускают	Да	Да
Выше	В гостях	На месте	Да	Нет
Ниже	В гостях	На месте	Нет	*???*

Само дерево


Соперник в турнирной таблице
Ниже
Выше
Матч проходит
Матч проходит
Дома
В гостях
Дома
В гостях
1
0
Лидеры команды
0
На месте
Пропускают матч
Идёт дождь
1
Да
Нет
0
1

Его использование

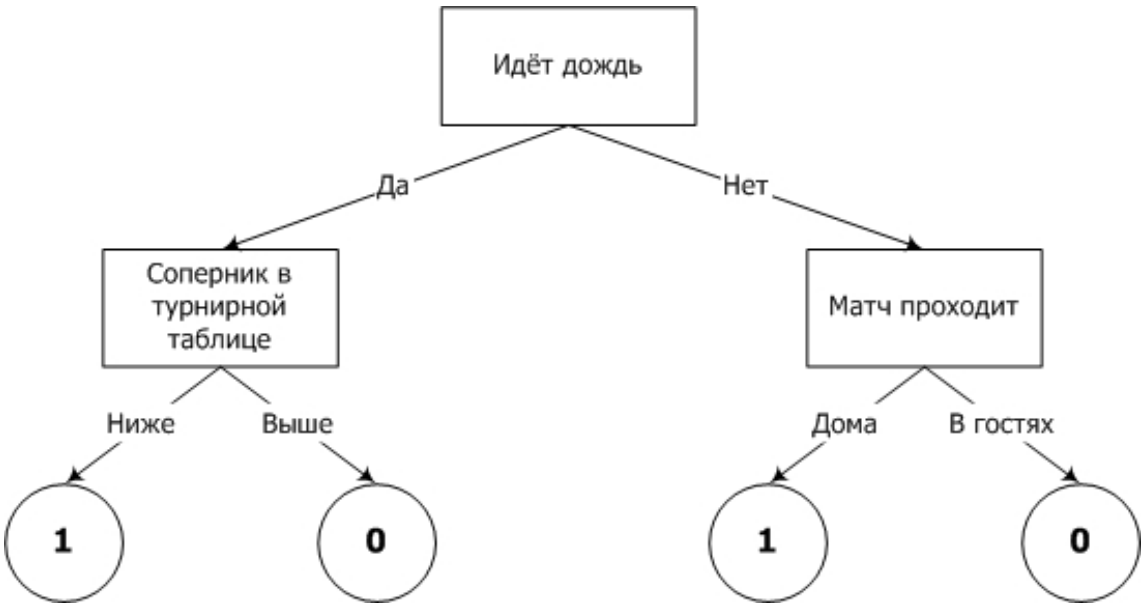
Как использовать:

Соперник = Ниже
Играем = В гостях
Лидеры = На месте
Дождь = Нет
Победа = ???

Спускаемся по дереву, выбирая нужные атрибуты, и получаем ответ: судя по нашему дереву, «Зенит» этот матч должен проиграть.

Оптимальное дерево

Это большое дерево. А вот дерево для тех же самых данных, но куда меньше:



Алгоритм построения

Как строить дерево:

- Выбираем очередной атрибут (условие) A, помещаем его в корень
- Для всех его значений i:
  - Разбиваем исходную выборку на две части: условие A выполняется / условие A не выполняется
  - Рекурсивно строим дерево для каждой из полученных частей
- Выдаём полученное дерево

Главная проблема:

- Как выбирать условие в вершине (атрибут A)?

Наша цель - добиться того, чтобы в каждом листе дерева находились объекты одного класса. Поэтому условие в вершине дерева тем лучше, чем более однородные объекты получаются в двух группах, на которые разбились объекты этим условием.

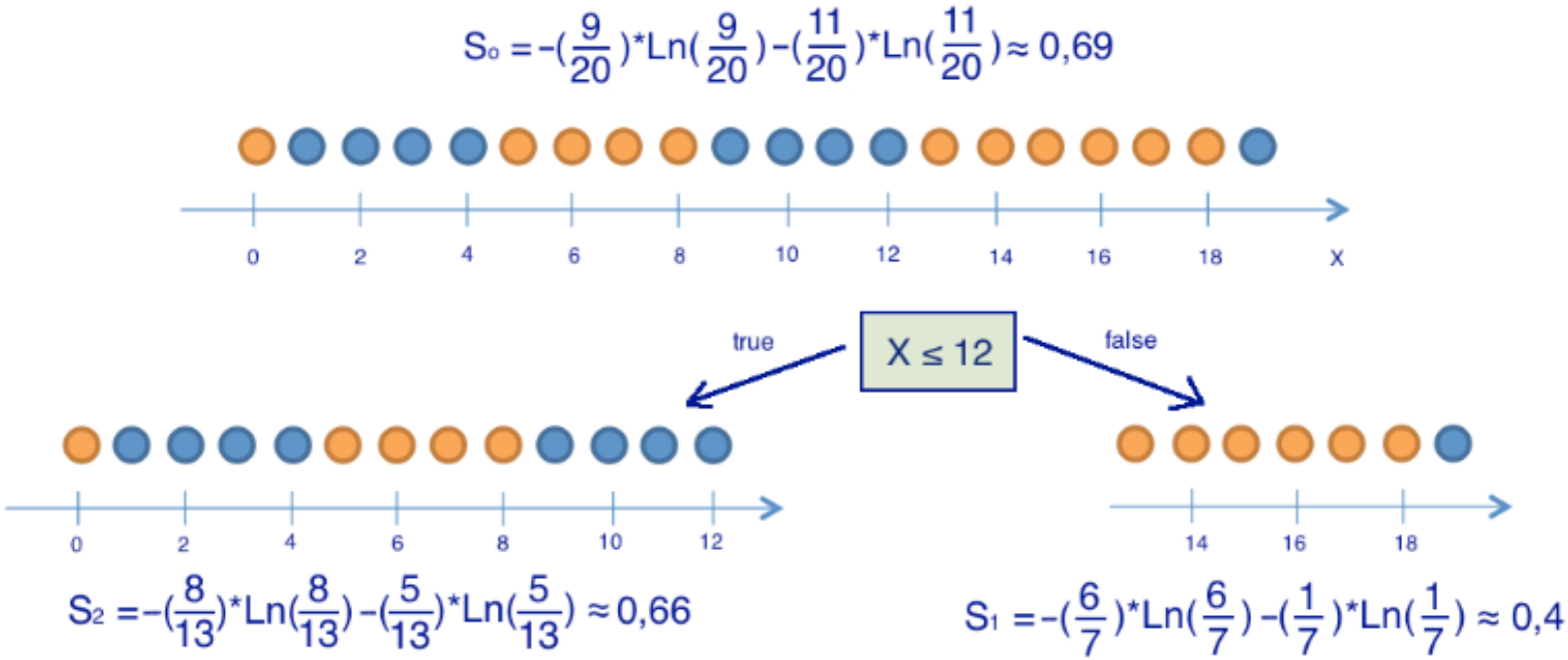
Энтропия

Пусть величина x принимает значения $x_1, \ldots, x_n$ с вероятностями $p_1, \ldots, p_n$. Тогда величина

$$H(x) = -\sum_{i=1}^{n} p_i \log p_i$$

называется энтропией случайной величины x.

Энтропийный критерий



В решающем дереве выбирается такое условие разбиения, чтобы средняя энтропия (с весами, равными мощностям каждой из групп после разбиения) была минимальной.

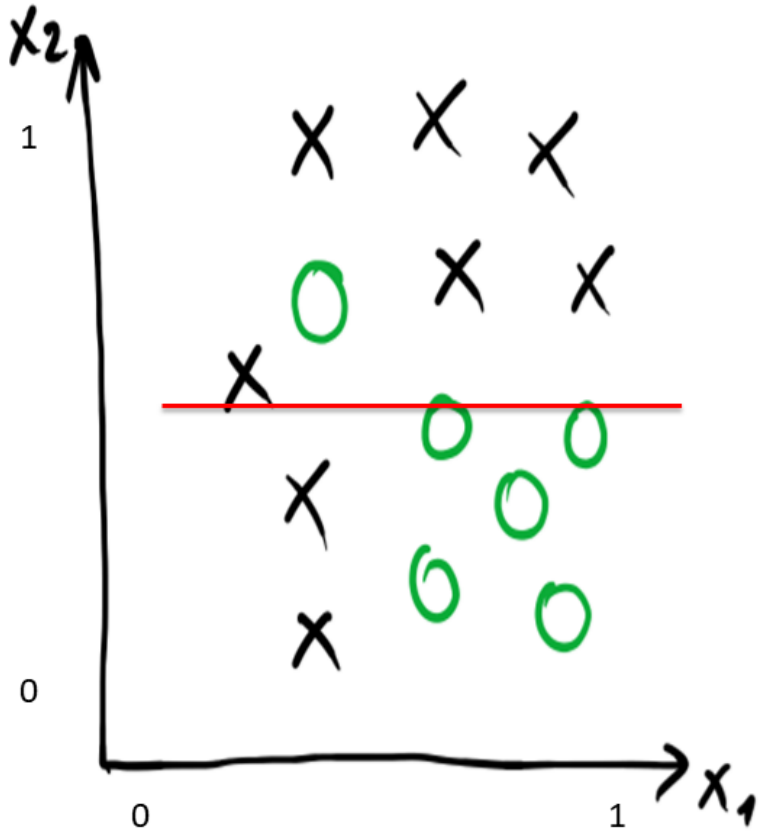
Критерий информативности Джини

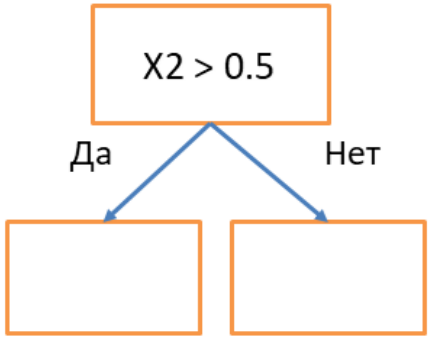
Другим способом подобрать условие разбиения в вершине дерева является оптимизация критерия Джини (не путать с ROC-AUC индексом Джини!):

$$G(x) = \sum_{i=1}^{n} p_i(1 - p_i)$$

Проблема переобучения

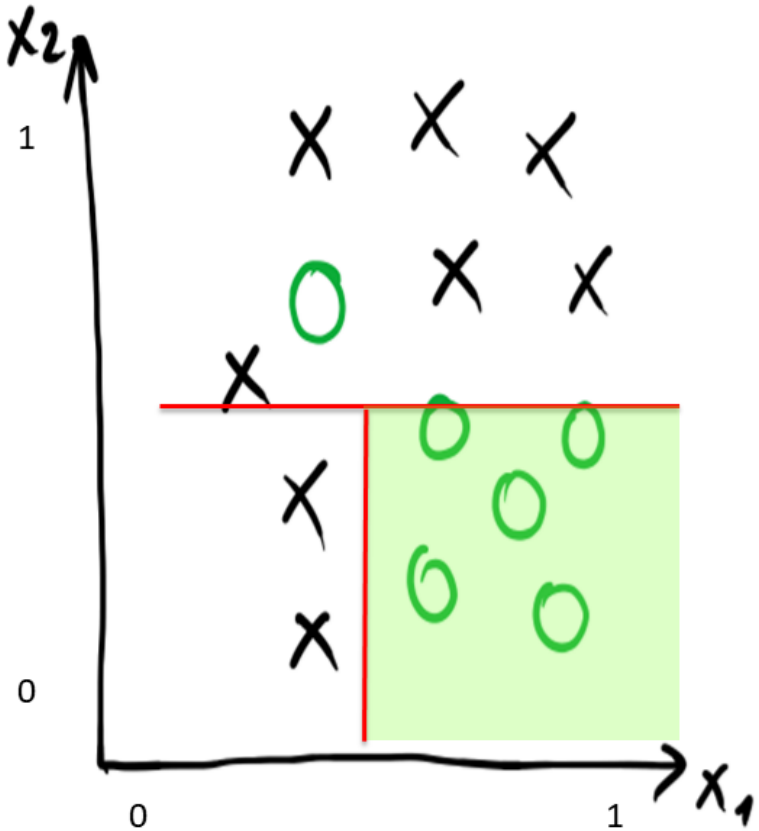
Нашли лучшее первое ветвление

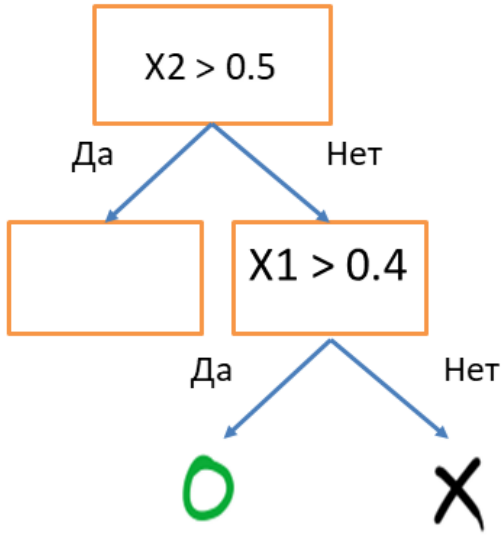




Проблема переобучения

Нашли лучшее второе ветвление

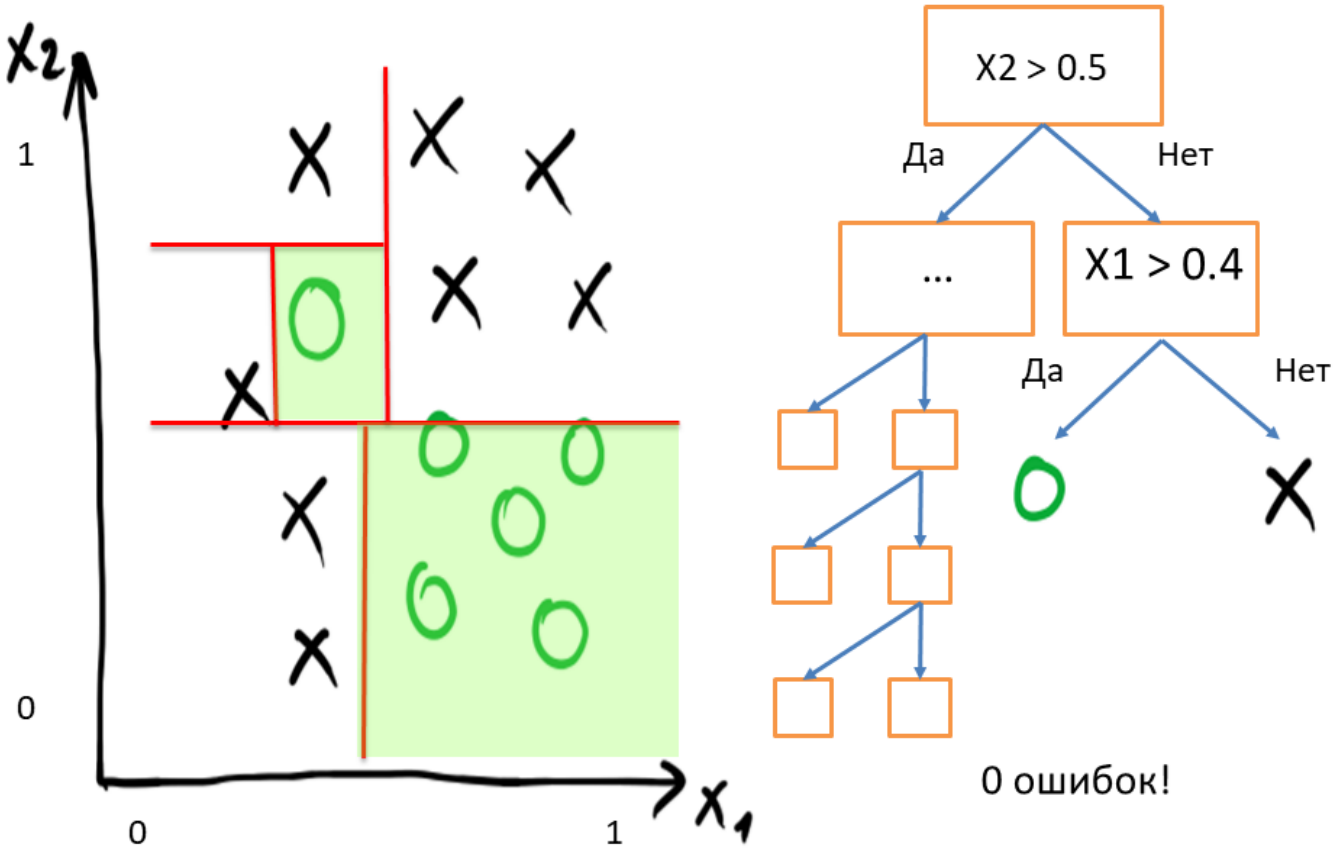




8

Проблема переобучения

Построили все дерево



8

Скорее всего переобучились!

Методы борьбы с переобучением

- Ограничение максимальной глубины дерева
- Ограничение минимального числа объектов в листе
- Стрижка дерева: сначала строится переобученное дерево, а затем отрезаем некоторые вершины таким образом, чтобы максимально увеличить предсказательную способность

Бэггинг

Чтобы снизить переобученность алгоритма (решающего дерева), можно обучать его на:

- подмножестве X_0 всех объектов X
- подмножестве f_0 всех признаков f

Бэггинг

Бэггинг:

- Обучим N алгоритмов $b_1(x), b_2(x), \ldots, b_N(x)$ из одного семейства (например, решающих деревьев), каждый из которых построен на различных подмножествах объектов и признаков.
- В качестве финального предсказания используем алгоритм, усредняющий ответы полученных алгоритмов

$$a(x) = \frac{1}{N} \sum_{i=1}^{N} b_i(x).$$

В случае, если алгоритмы $b_i(x)$ - это решающие деревья, то алгоритм $a(x)$ называется *решающим лесом*.

Настраиваемые параметры решающего леса

В реализации sklearn алгоритм RandomForest имеет следующие параметры:

- n estimators - число деревьев
- criterion - критерий поиска оптимального условия в вершине
- max depth - максимальная глубина деревьев
- min samples leaf - минимальное число деревьев в листе
- и другие (см. sklearn)

Решающий лес - резюме

- Решающий лес - это бэггинг над решающими деревьями
- Решающий лес снижает эффект переобученности каждого дерева за счет обучения по подвыборкам и усреднения объектов
- Предсказательная способность решающего леса сильно превосходит предсказательную способность решающего дерева

Отбор признаков

Иногда в задаче присутствует излишнее количество признаков, и их разумный отбор значительно улучшает качество предсказательной модели.

VarianceThreshold

Можем удалить признаки, которые имеют очень маленькую дисперсию, т.е. практически константы.

Отбор по корреляции с целевой переменной

Для каждого признака вычислим его корреляцию с целевой переменной. Будем выкидывать признаки, имеющие маленькую корреляцию.

Более сложные методы отбора признаков

- Filtration methods (фильтрационные методы)
- Wrapping methods (оберточные методы)
- Model selection (встроенный в модель отбор признаков)

Фильтрационные методы

Фильтрационные методы - это отбор признаков по различным статистическим тестам. Идея метода состоит в вычислении влияния каждого признака в отдельности на целевую переменную (с помощью вычисления некоторой статистики).

- Очевидный плюс метода: скорость, так как мы N раз вычисляем значение некоторой статистики, где N - количество признаков.

Фильтрационные методы

В sklearn есть сразу несколько методов, использующих отбор по статистическим критериям. Среди них выделим следующие:

- SelectKBest - оставляет k признаков с наибольшим значением выбранной статистики
- SelectPercentile - оставляет признаки со значениями выбранной статистики, попавшими в заданную пользователем квантиль
- и другие (см.sklearn)

Статистические тесты для отбора признаков

- mutual information: для векторов X и Y статистика вычисляется по формуле

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)\,p(y)} \right)$$

- хи-квадрат:

$$\chi^2(X,Y) = \sum_{i=1}^{n} \frac{(Y_i - X_i)^2}{X_i}$$

- и другие (см.sklearn)

Оберточные методы

Оберточные методы используют жадный отбор признаков, т.е. последовательно выкидывают наименее подходящие по мнению методов признаки.
В sklearn есть оберточный метод - Recursive Feature Elimination (RFE).

Параметры метода:
a) алгоритм, используемый для отбора признаков (например, RandomForest)
b) число признаков, которое мы хотим оставить.

Встроенные в модель методы

- В алгоритме sklearn random forest есть метод feature importances, который показывает важность каждого признака (по мнению random forest). С помощью этого метода можно выкидывать признаки, имеющие маленькую важность для алгоритма.
- При добавлении регуляризатора в модель мы уменьшаем влияние различных признаков на финальную модель. В случае, если мы добавляем l_1-регуляризатор, то в силу вида регуляризатора (сумма модулей весов), модель автоматически обнуляет веса некоторых признаков, то есть выкидывает их. Линейная модель с l_1-регуляризатором в sklearn: LASSO.

Линейная регрессия. Постановка задачи.

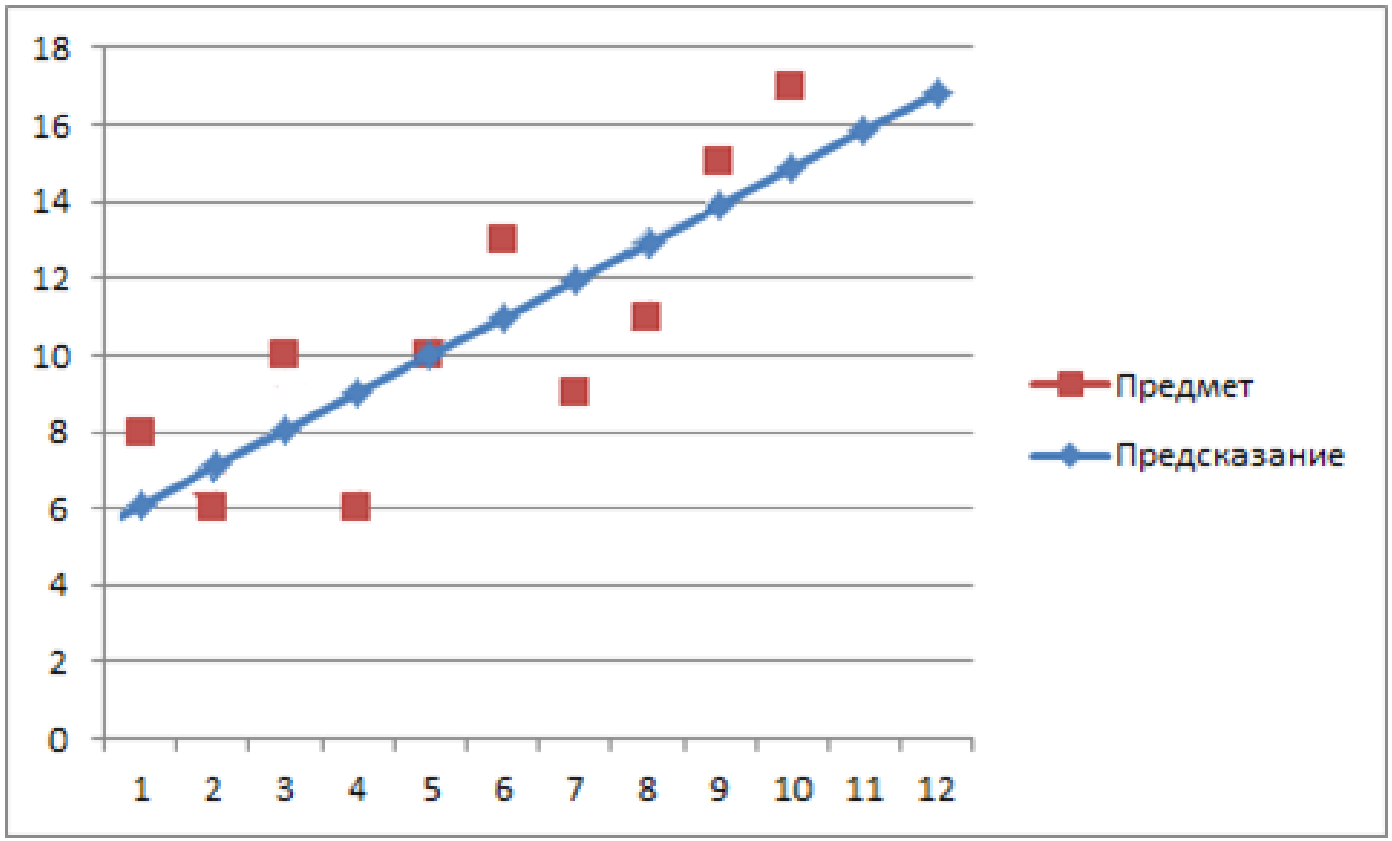
Пусть X - матрица признаков (каждый объект описывается n вещественными признаками), y - вектор ответов, $y \in \mathbb{R}$. *Задача регрессии* - задача восстановления зависимости $y = y(x)$.

Если x_1,x_2,... - признаки объекта, а y - некоторый вещественный ответ на этом объекте, то *линейная регрессия* - это зависимость

$$y(x) = w_1 \cdot x_1 + w_2 \cdot x_2 + \dots.$$

В процессе обучения метода подбираются оптимальные коэффициенты (веса) $w_1, w_2, \dots$ при признаках.

Пример

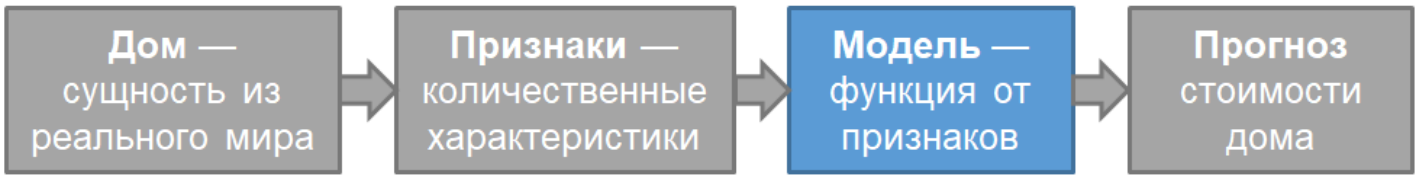


Примеры регрессионных задач

- Недвижимость: приблизительная стоимость объекта, исходя из данных по уже совершенным сделкам
- HeadHunter.ru: вероятная заработная плата для вакансий, в которых это напрямую не указано
- Нефтяные компании: потенциальный эффект от дополнительных мероприятий по повышению эффективности добычи
- Трейдеры: прогнозирование стоимости акций на фондовом рынке

Пример: предсказание стоимости дома

- Задача: предсказание стоимости дома по его характеристикам
- Машинное обучение — извлечение закономерностей из данных



Обучающая выборка

Площадь	Цена
50	250
60	340
10	20
90	800

Какие признаки можно использовать в данной задаче?

Инжиниринг фичей

Возможные признаки:

- площадь
- площадь^2
- площадь^3
- $\sin$ (площадь)
- $\sqrt{}$(площадь)
- и так далее

Возможные модели

- w_1*площадь
- w_1*площадь^2
- w_1*площадь+w_2*площадь^2
- и так далее

Метрики качества в задачах регрессии: MSE

Основной способ измерить ошибку - вычилить среднеквадратичное отклонение предсказания от ответа:

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

MSE плохо интерпретируема, так как, например, если предсказываем стоимость в рублях, то MSE выдает ответ в рублях в квадрате.

Метрики качества в задачах регрессии: RMSE

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

- сохраняет размерность
- подходит для сравнения различных моделей между собой

Но RMSE не позволяет оценить, насколько хорошо данная модель решает задачу.

Метрики качества в задачах регрессии: R2

Коэффициент детерминации:

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

где $\overline{y}$ - среднее значение вектора ответов.

Линейная модель по площади (кейс со стоимостью дома)

Модель $a(x)$=5*площадь

Площадь	Прогноз	Цена	$(a - y)^2$
50	250	250	0
60	300	340	1600
10	50	20	900
90	450	800	122500

MSE: 125000
RMSE: 353.55

Квадратичная модель по площади (кейс со стоимостью дома)

Модель $a(x)$=0.1*площадь^2

Площадь	Прогноз	Цена	$(a - y)^2$
50	250	250	0
60	360	340	400
10	10	20	100
90	810	800	100

MSE: 600
RMSE: 24.5

R2

Эта статистика показывает, насколько условная дисперсия модели отличается от дисперсии реальных значений Y. Если этот коэффициент близок к 1, то условная дисперсия модели достаточно мала и весьма вероятно, что модель неплохо описывает данные.
Чем больше наблюдений, тем больше наименьшее значение R-квадрат, которое нас устроит.

Обучение линейной регрессии

Пусть дана матрица объектов-признаков X. Обучение линейной регрессии $a(x)$ состоит в подборе весов $w_1, w_2, \ldots$ при признаках $x_1, x_2, \ldots$ таким образом, чтобы на этих весах достигался минимум функционала потерь $Q(a, X)$.
Обычно в задачах регрессии при обучении минимизируют MSE.

Обучение происходит методом *градиентного спуска*.

Метод градиентного спуска

Градиент функции - это вектор

$$\nabla f(x_1, \dots, x_n) = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots)$$

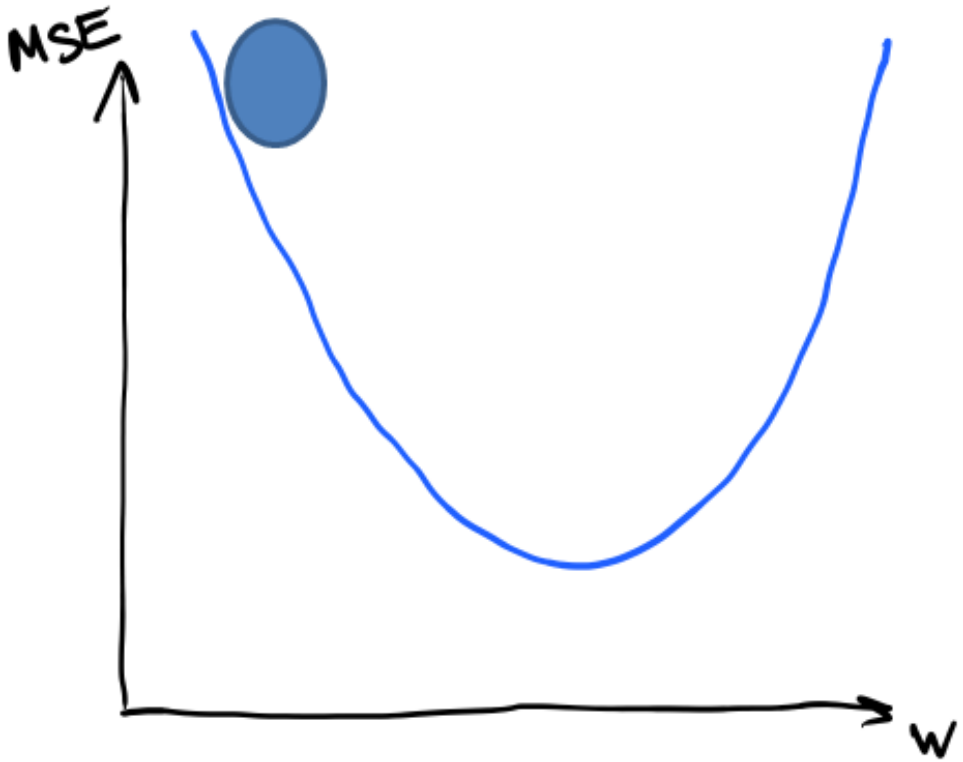
Теорема. *Градиент - это направление наискорейшего возрастания функции, а антиградиент - направление наискорейшего убывания функции.*

Шаг метода градиентного спуска.

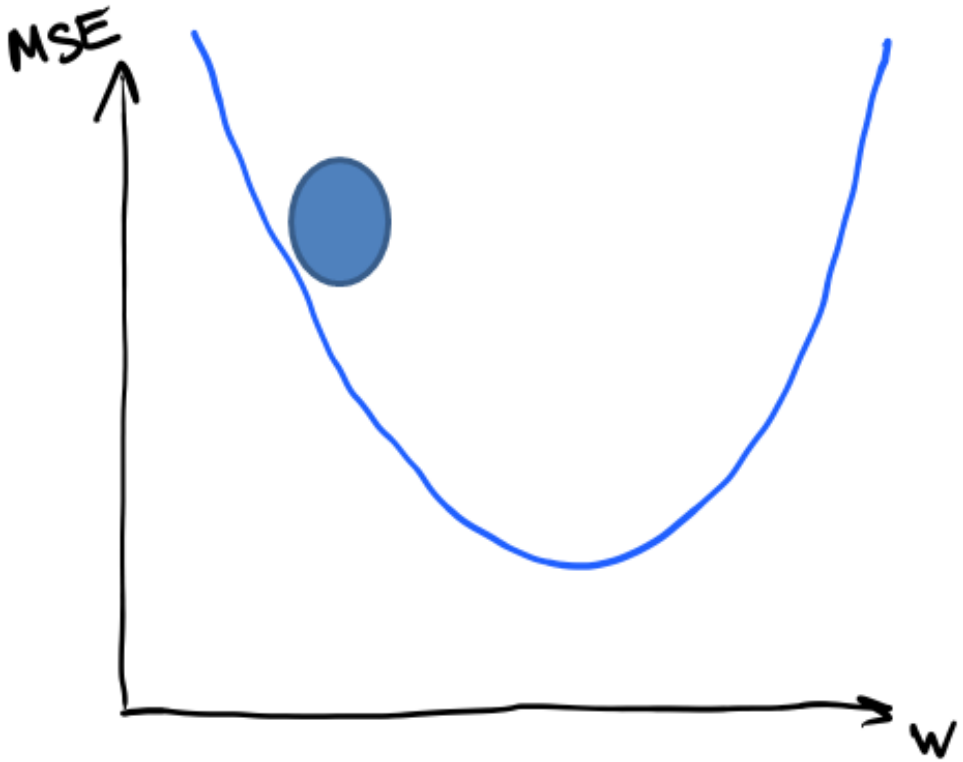
$$w^{(k)} = w^{(k-1)} - \nu \nabla Q(w^{(k)}),$$

где ν - размер градиентного шага.

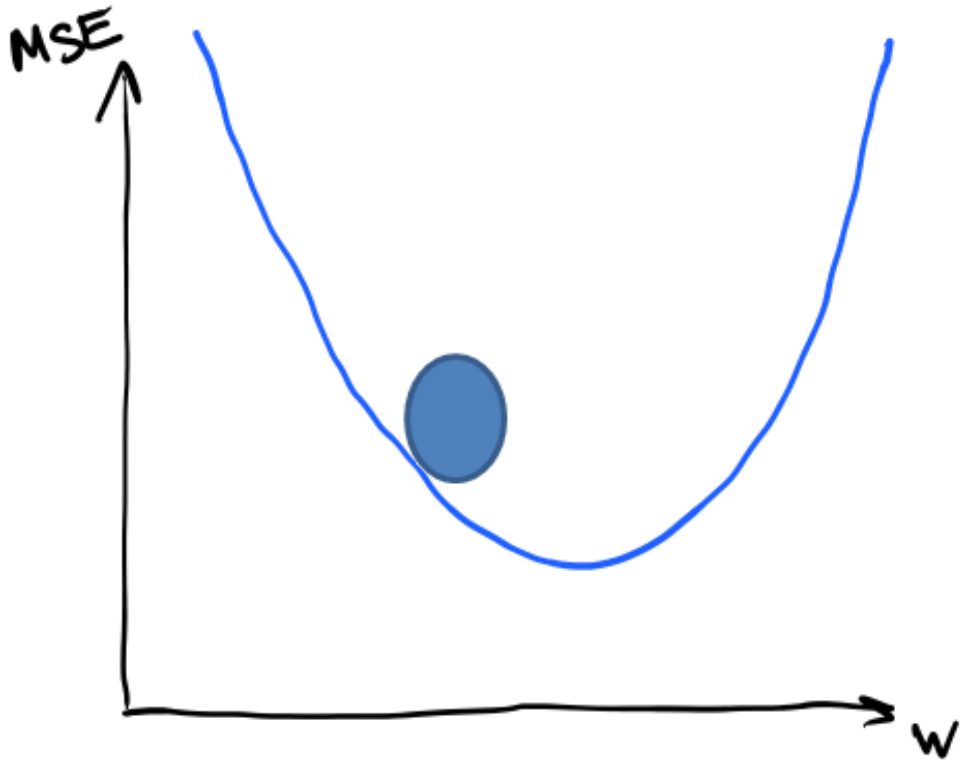
Градиентный спуск



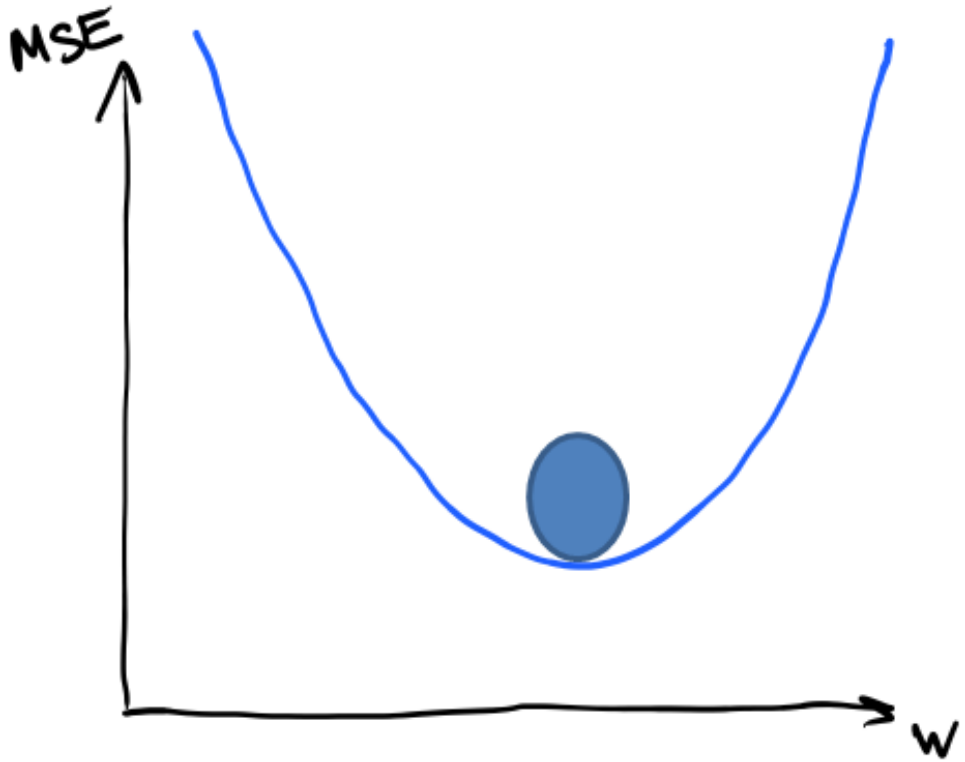
Градиентный спуск



Градиентный спуск



Градиентный спуск



Линейная регрессия - резюме

Линейная модель в общем виде суммирует значения всех признаков с некоторыми весами.
Веса при признаках —- параметры, которые необходимо настраивать в процессе обучения.

Плюсы:

- Линейные модели способны обучаться на сверхбольших выборках
- Могут работать на данных с большим количеством признаков (например, на текстах)
- Хорошо интерпретируются

Минусы:

- Линейные модели очень чувствительны к выбросам
- Могут восстанавливать только линейные зависимости