Занятие 3. Линейные методы регрессии. Часть 2

Колмагоров Евгений
ml.hse.dpo@yandex.ru

21 октября 2024

План лекции

1. Метрики качества
2. Переобучение
3. Кросс-валидация
4. Регуляризация модели
5. Гиперпараметры модели

Метрики качества. Мотивация

Введённый функционал ошибки ***Q(a, X)*** необходим для поиска оптимальный весов модели ***a(x, w)***:

$$w^* = argmin_w Q(a, X)$$

При этом конкретное значение Q, может хорошо подходить для поиска оптимальных весов ***w***, но быть плохо интерпретируемым

Вспомним MSE

$$Q(a, X) = \frac{1}{N} \sum_{i=0}^{N} (a(x_i) - y_i)^2$$

- Насколько данная метрика интерпретируемая?
- В каких единицах измеряется?
- Значение равное 10 это хорошо или плохо?

Метрика качества

Для того, чтобы иметь более понятное представление о том, насколько хорошо решается целевая задача вводят специальные метрики качества.

Метрика качества – функция, которую используют для оценки качества построенной (уже обученной) модели.

Ещё раз про функционал и метрики качества

- **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.

Замечание: бывает, что одна и та же функция может быть использована и как функционал ошибки для поиска весов и как метрика качества

Так ли плох MSE в качестве метрики качества?

$$Q(a, X) = \frac{1}{N} \sum_{i=0}^{N} (a(x_i) - y_i)^2$$

Плюсы:

- Несмотря на плохую интерпретацию тем не менее позволяет сравнивать модели
- Подходит для контроля качества во время обучения

Минусы:

- Плохо интерпретируем, нет сохранения единиц измерения
- Тяжело понять, насколько хорошо данная модель решает задачу, так как нет ограничения на данную метрику сверху

Попробуем исправить первый минус MSE

Извлечём корень из среднеквадратичной ошибки:

$$Q(a, X) = \sqrt{\frac{1}{N} \sum_{i=0}^{N} (a(x_i) - y_i)^2}$$

Плюсы:

- Все плюсы MSE
- Сохраняет единицы измерения

Минусы:

- Тяжело понять, насколько хорошо данная модель решает задачу, так как нет ограничения на данную метрику сверху

RMSE

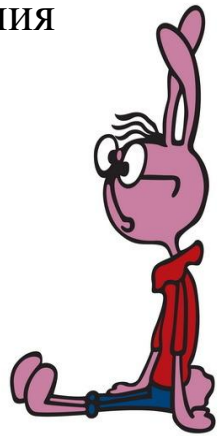
MSE

Попробуем ограничить

Одинаковые значения ошибки MSE в одних задачах могут быть очень хорошим результатом обучения, а в других очень плохим.

Так например:

- Значение ошибки *10* с целевой переменной y с диапазоном изменения [10000, 50000] очень **хороший** результат
- Но то же самое значение *10* очень **плохой** результат для целевой переменной с диапазоном [0, 1]



Нормализация MSE

Попробуем нормализовать MSE так, чтобы в независимости от того, какая задача решается было понятно насколько хорошо.

Коэффициент детерминации R^2:

$$R^2(a, X) = 1 - \frac{MSE(a,Y)}{D[Y]},$$ где D[Y] - дисперсия целевой переменной.

Или в развёрнутом виде:

$$R^2(a, X) = 1 - \frac{\sum_{i=0}^{N}(a(x_i)-y_i)^2}{\sum_{i=0}^{N}(y_i-\bar{y})^2}$$, где $\bar{y} = \frac{1}{N}\sum_{i=0}^{N} y_i$

Почему делим на дисперсию?

Теорема: Предсказания равные среднему значению целевой переменной y, являются оптимальным решением для функционала ошибки MSE в классе константных алгоритмов.

Или другими словами: если в качестве ответа алгоритма на любом объекте используется некоторая константа C, то самым лучшим таким ответом с точки зрения среднеквадратичной ошибки является среднее значение целевых переменных

Доказательство

Пусть a - константный алгоритм, тогда функционал ошибки для него:

$$Q(a, X) = \sum_{i=1}^{N} (a - y_i)^2 \rightarrow min_a$$

Продифференцируем Q по a и приравняем производную к нулю (необходимое условие экстремума):

$$\frac{\partial Q}{\partial a} = \frac{2}{N} \sum_{i=0}^{N} (a - y_i) = 0$$

Разобъём сумму на две и перенесём часть с y вправо:

$$\sum_{i=0}^{N} a = \sum_{i=0}^{N} y_i$$

$$a = \frac{1}{N} \sum_{i=0}^{N} y_i = \bar{y}$$

Коэффициент детерминации

Если подставить среднее значение y в формулу MSE получим дисперсию.

Поэтому отношение

$$\frac{\sum_{i=0}^{N}(a(x_i)-y_i)^2}{\sum_{i=0}^{N}(\bar{y}-y_i)^2}$$

Показывает во сколько раз среднеквадратичная ошибка алгоритма **меньше** ошибки константного решения.

В каких диапазонах лежат значения данного отношения?

Коэффциент детерминации

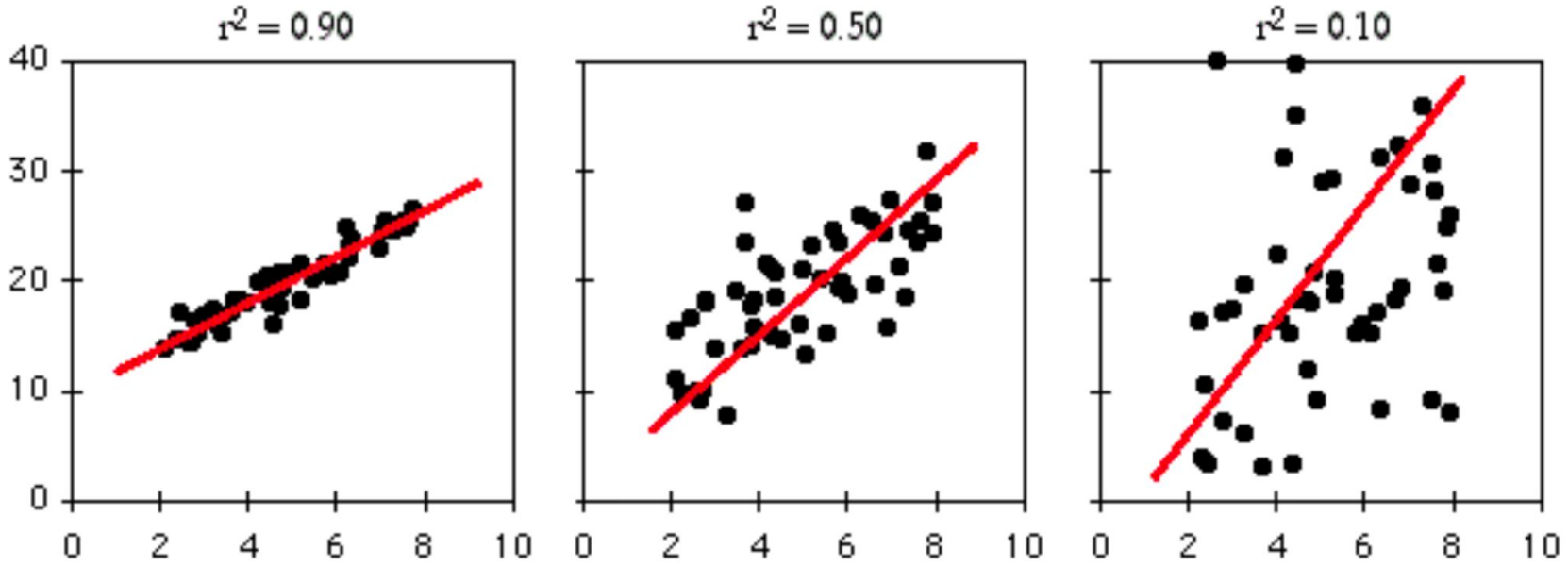
Нормализуем формулу так, чтобы максимальное принимало значение 1

$$R^2(a, X) = 1 - \frac{\sum_{i=0}^{N}(a(x_i) - y_i)^2}{\sum_{i=0}^{N}(y_i - \bar{y})^2}$$

Таким образом в такой форме коэффициент детерминации выражает долю *дисперсии целевой переменной, объясняемую моделью.*

- Чем ближе R^2 к 1, тем лучше модель объясняет данные
- Чем ближе R^2 к 0, тем ближе модель к константному решению
- Отрицательный R^2 говорит о том, что модель работает хуже чем константное решение и плохо объясняет данные

Различные значения коэффициента детерминации



А минусы будут?

Несмотря на универсальность данного коэффициента для целевых задач достаточно сложно объяснить заказчикам, как улучшение коэффициента детерминации повлияет на бизнес-метрики.

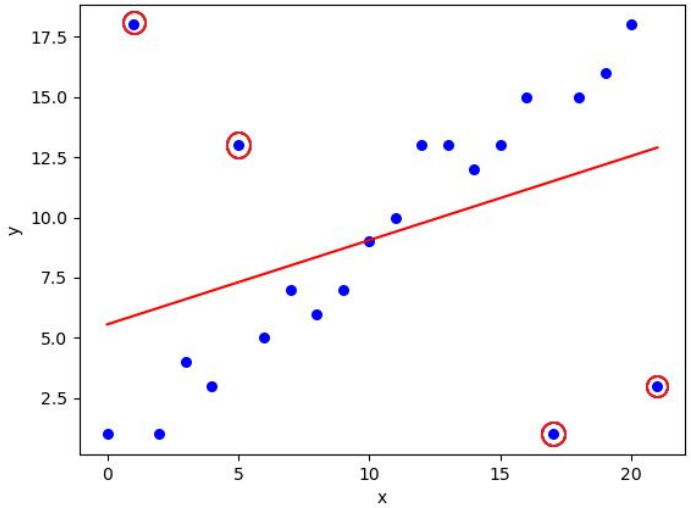
Например, сколько принесёт добавочной прибыли улучшение алгоритма оценки стоимости домов при улучшении коэффициента детерминации на 0.1?

MAE - mean absolute error

Средняя абсолютная ошибка:

$$MAE(a, X) = \sum_{i=0}^{N} |a(x_i) - y_i|$$

Так как теперь все ошибки не возводятся в квадрат, то штраф на отдалённых объектах не будет вносить такого большого веса, как в случае с MSE



Влияние выбросов на функционал

y	$a_1(x)$	$(a_1(x) - y)^2$	$\lvert a_1(x) - y \rvert$
1	2	1	1
2	1	1	1
3	2	1	1
4	5	1	1
5	6	1	1
100	7	8649	93
7	6	1	1
		$\mathrm{MSE} = 1236$	$\mathrm{MAE} = 14.14$

MAE - mean absolute error

Средняя абсолютная ошибка:

$$MAE(a, X) = \sum_{i=0}^{N} |a(x_i) - y_i|$$

Плюсы:

- Меньшая чувствительность к выбросам

Минусы:

- Функция модуля не дифференцируема, поэтому сложно производить оптимизацию напрямую

Наилучшее константное решение для MAE

Теорема: Предсказания равные медианному значению целевой переменной y, являются оптимальным решением для функционала ошибки MAE в классе константных алгоритмов.

$$a^* = median\{y_1, y_2, \ldots, y_n\}$$

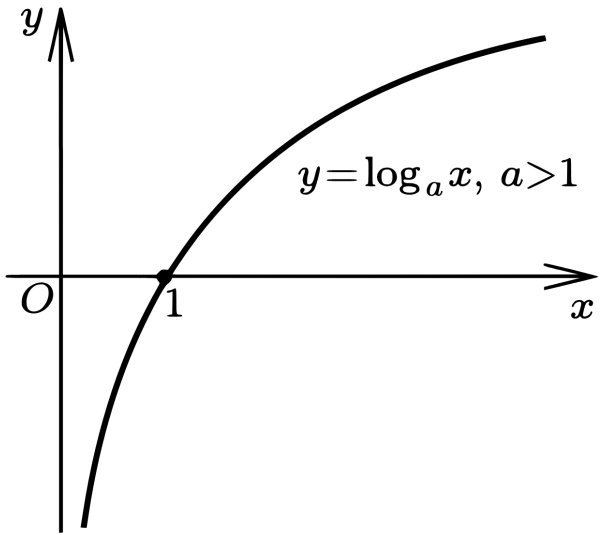
Доказательство данного факта аналогично теореме об MSE с той лишь разницей, что теперь производная MAE по a, будет представлять собой, сумму ряда $sign(a-y_i)$, минимум которого достигается если половина членов будет +1, а другая половина -1

MSLE - Mean Squared Logarithmic Error

Среднеквадратичная логарифмическая ошибка:

$$MSLE(a, X) = \frac{1}{N} \sum_{i=0}^{N} (log(a(x_i) + 1) - log(y_i + 1))^2$$

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



MAPE - Mean Absolute Percentage Error

Отнормированная на целевую переменную у MAE ошибка:

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

Так как целевая переменная находится в знаменателе, то MAPE измеряет относительную ошибку

MAPE - Mean Absolute Percentage Error

Отнормированная на целевую переменную у MAE ошибка:

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

Плюсы:

- Ограничена интервалом $0 \leq \text{MAPE} \leq 1$
- Хорошо интерпретируема: например, значение 0.16 означает, что ошибка модели в среднем составляет 16% от фактических значений

MAPE - Mean Absolute Percentage Error

Отнормированная на целевую переменную у MAE ошибка:

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

Минусы:

- Оценка смещена относительно перепрогноза и недопрогноза. Например, если правильный ответ *y = 10*, а прогноз *a(x) = 20*, то

  $$MAPE(a, X) = \frac{|10-20|}{|10|} = 1$$

  Но если *y=30*, то

  $$MAPE(a, X) = \frac{|30-20|}{|30|} = 0.33$$

SMAPE - Symetric Mean Absolute Percentage Error

SMAPE - попытка сделать ошибки на при недо- и пере- прогнозах одинаковыми:

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

SMAPE - Symetric Mean Absolute Percentage Error

SMAPE - попытка сделать ошибки на при недо- и пере- прогнозах одинаковыми:

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

Но на самом деле это не так….

Пусть $y = 10$, а прогноз a(x) = 20, то ошибка

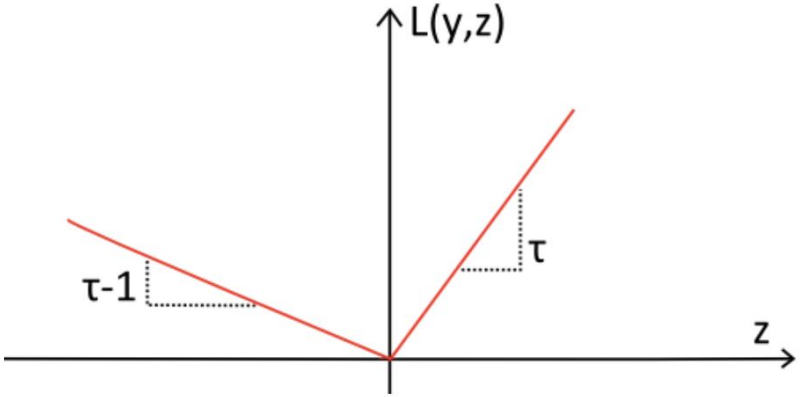
$SMAPE(a, X) = \frac{|10-20|}{(|10|+|20|)/2} = \frac{2}{3} \approx 0.67$, а если истинный ответ y = 30, то ошибка

$SMAPE(a, X) = \frac{|30-20|}{(|30|+|20|)/2} = \frac{2}{5} = 0.4$

Квантильная регрессия

Квантильная функция потерь:

$$Q_\tau(a, X) = \sum_{i=1}^{N} \rho_\tau(y_i - a(x_i))$$



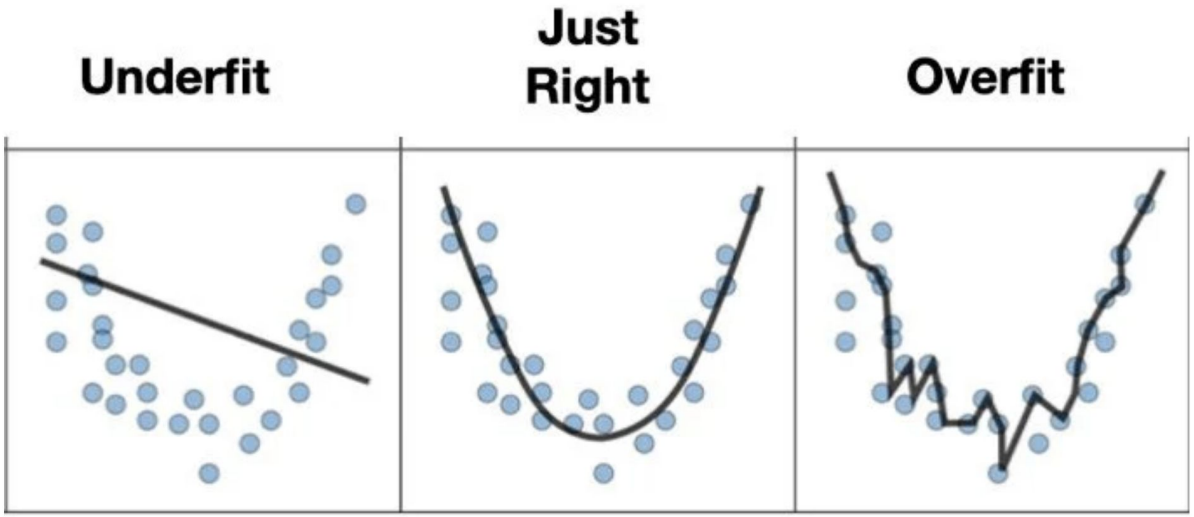
Где:

$$\rho(z) = \begin{cases} (\tau - 1) \cdot z, & z < 0 \\ \tau \cdot z, & z \geq 0 \end{cases}$$

Параметр $\tau \subset [0,1]$, чем больше τ тем больше штрафует за занижение прогноза

Эффект переобучения

Переобучение (overfitting) - явление, при котором качество модели на новых данных сильно хуже, чем на обучающей выборке.

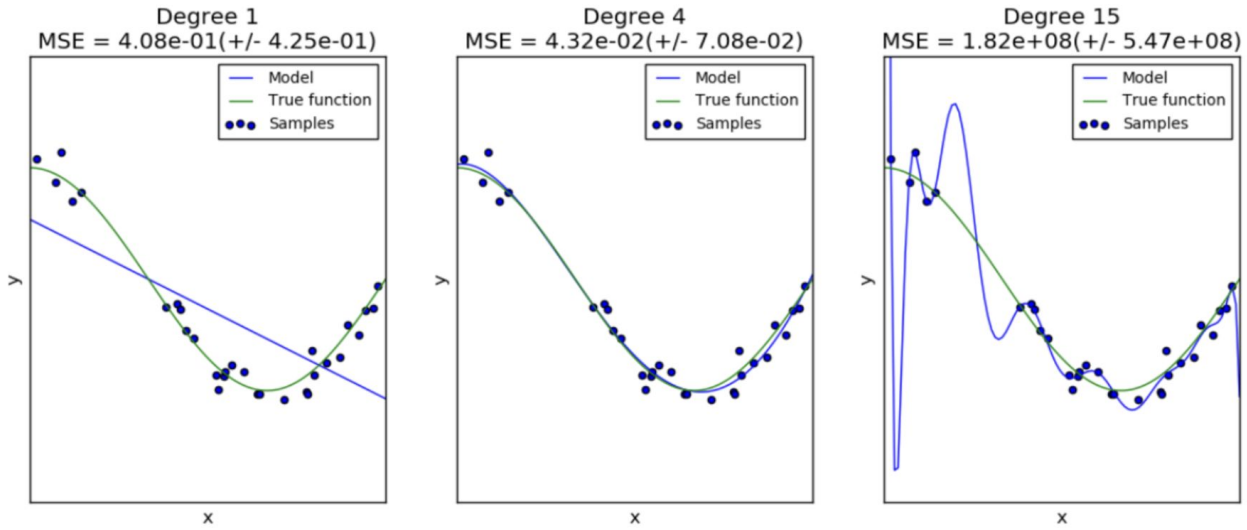


Пример. Полиномиальная регрессия

Пример: Пусть истинная зависимость представляет собой $y = cos(1.5\pi x) + N(0, 0.01)$.
Попробуем обучить линейную полиномиальную регрессию

$$a(x.w) = w_0 + w_1 \cdot x + w_2 \cdot x^2 + \ldots + w_d \cdot x^d$$

с разным числом параметров $\{w_0, w_1\}$, $\{w_0, w_1, w_2, w_3, w_4\}$ и $\{w_0, w_1, \ldots, w_{15}\}$.



https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

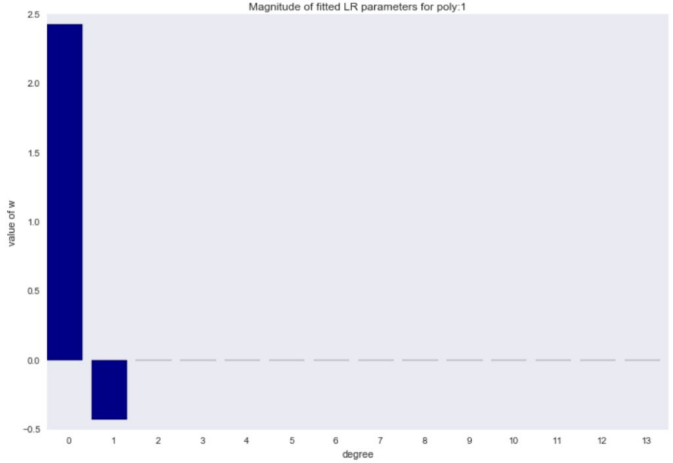
Причины переобучения

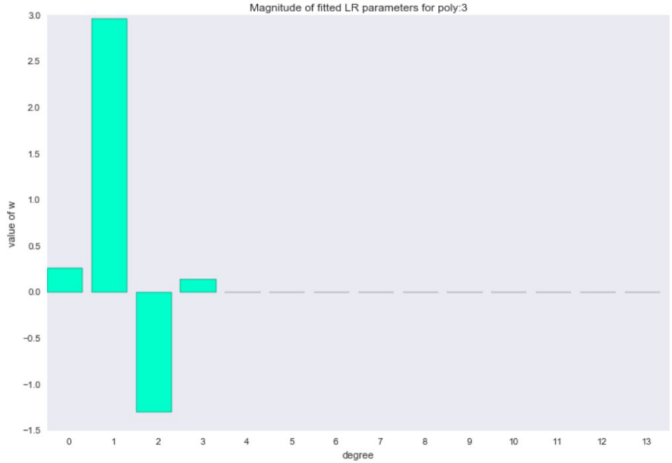
- **Сложность модели**. Если модель слишком сложная для доступного набора данных, она может чрезмерно подгоняться к обучающим данным и улавливать шумы и флуктуации, а не истинные базовые закономерности.
- **Неадекватные данные**. Когда обучающий набор данных невелик, моделям может быть трудно усвоить реально существующие шаблоны. При меньшем количестве примеров выше вероятность того, что модель запомнит обучающие данные, а не будет обобщать их на новые, ранее невиданные экземпляры.
- **Зашумлённые данные**. Если обучающие данные содержат шум, выбросы и нерелевантную информацию, модель может включать их в свой процесс обучения.

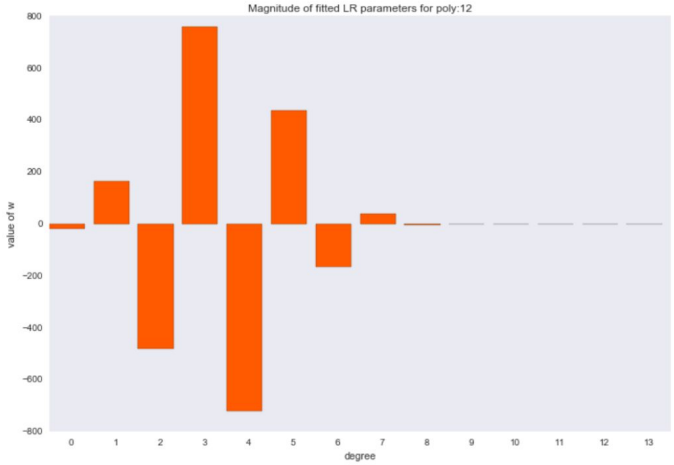
Сигналы переобученной модели

- Большая разница между ошибками на обучающей и тренировочных выборках
- Большие значения у параметров модели (w_j)
- Неустойчивость разделяющей поверхности

Значения весов из предыдущего примера







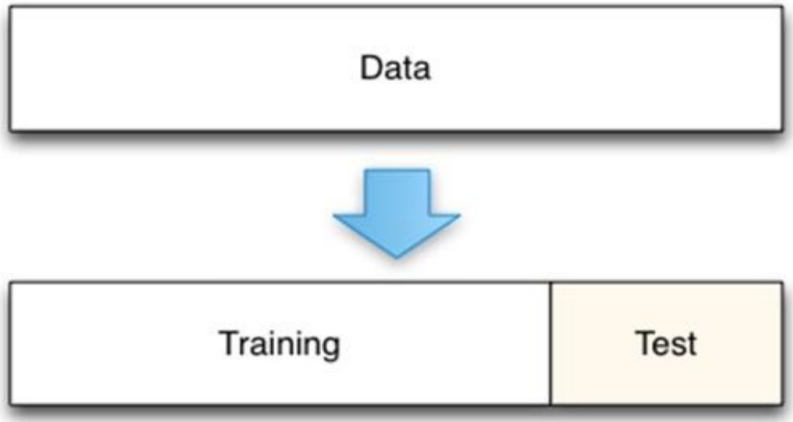
Оценивание модели

- Отложенная выборка
- Кросс-валидация

Отложенная выборка

Делим имеющуюся тренировочную выборку на две части:

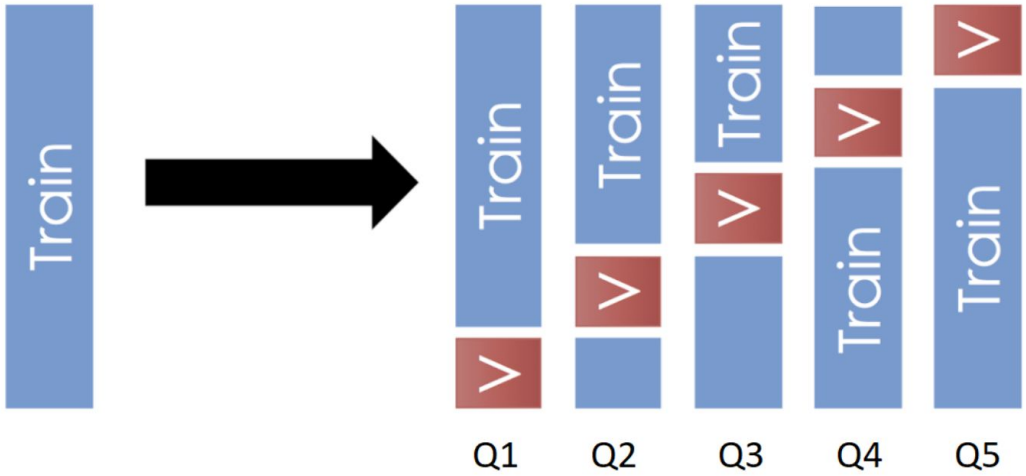
- Данные на которых будет происходить обучение модели (train)
- Данные на которых будем оценивать качество модели (test), в некоторых случаях их ещё называют валидацией



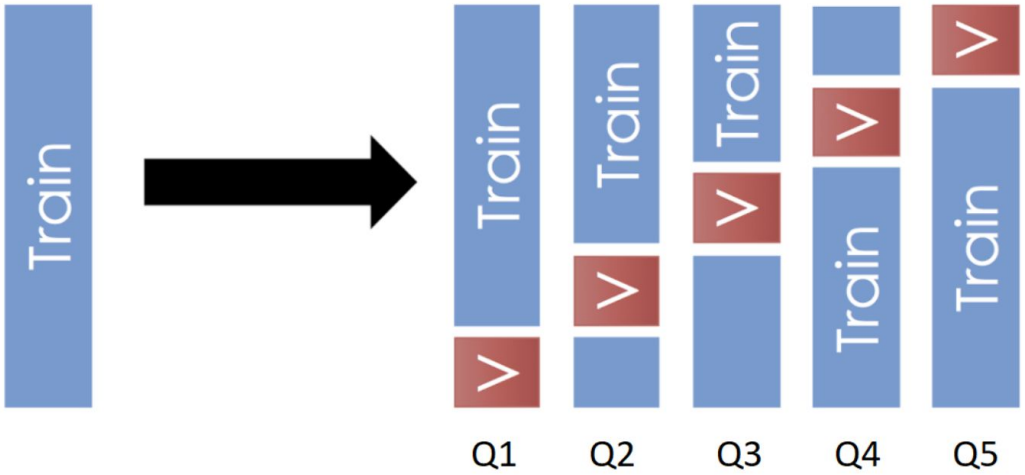
Недостаток: Результат сильно зависит от того, каким именно было разбиение. Может оказаться так, что в тест могли преимущественно попасть “лёгкие” для модели объекты

Кросс-валидация

- Разбиваем объекты на тренровку (train) и валидацию (validation) несколько раз
- Для каждого разбиения вычисляем качество на валидационной части
- Усредняем полученные результаты



Кросс-валидация



$$CV = \frac{1}{k} \sum_{i=1}^{k} Q(a_i, X_i)$$

Виды кросс-валидации

В зависимости от того, как будут строиться разбиения на валидацию и каково будет их количество, выделяют разные виды валидации:

- **k-fold** cross-validation - делим данные на **k блоков**, каждый из которых по очереди становится контрольным
- **Complete** cross-validation - перебираем **все** разбиения
- **leave-one-out** cross-validation - размер одного блока **равен одному объекту** => число блоков = числу объектов в выборке

Выбор количества блоков

- *Чем приходится платить при большом k?*
- *Чем приходиться платить при маленьком k?*

Выбор количества блоков

- При большом оценка может иметь большую дисперсию из-за маленького размера валидационной части + требуется больше времени на оценку
- При маленьком *k* количество данных, на которых происходит обучение невелико, поэтому оценка может быть пессимистично занижена

Борьба с переобучением. Регуляризация

Утверждение: Если в выборке есть линейно зависимые признаки, то задача оптимизации $Q(a, X) \to \min$ имеет бесконечное число решений.

Линейно зависимые признаки X могут встречаться если:

- матрица X^TX плохо обусловлена (сильно коррелирующие признаки)
- Мало обучающих примеров ($N < d$) матрица X^TX вырождена

Регуляризация

Одной из причин переобучения модели являются слишком большие значения весов ***w***.

Поэтому добавим в задачу оптимизации функционала ошибки дополнительный член ***R(w)***, который будет штрафовать за большие значения весов:

$$Q^{reg}(a, X) = Q(a, X) + \alpha \cdot R(w)$$

Виды регуляризации

Регуляризация должна штрафовать за слишком большие значения весов.

Можно ввести различные метрики, того что считать большим весом:

- L_2 - регуляризатор, считаем сумму квадратов: $R(w) = \sum_{i=1}^{d} w_i^2$
- L_1 - регуляризатор, считаем сумму абсолютных значений: $R(w) = \sum_{i=1}^{d} |w_i|$

Вопрос: почему не добавляем член w_0 в регуляризацию?

Пример регуляризационного функционала

Регуляризация должна штрафовать за слишком большие значения весов.

L_2-регуляризация для среднеквадратичной ошибки:

$$Q^{reg}(a, X) = \frac{1}{N} \sum_{i=0}^{N} ((w, x_i) - y_i)^2 + \alpha \sum_{i=1}^{d} w_i^2$$

Параметр α – коэффициент регуляризации, который балансирует между **точностью** решения на обучающей выборке и **величиной весов** модели

Аналитическое решение с регуляризацией

Вспомним задачу оптимизации среднеквадартичной ошибки в матричном виде:

$$Q(a, X) = (Xw - Y)^T (Xw - Y) + \alpha w^T w \to min_w$$

При вычислении градиента этого функционала получим:

$$\nabla_w Q = \frac{1}{N}(2X^T Xw - 2X^T Y + \alpha N w) =$$
$$= \{\alpha^* = 2\alpha N\} =$$
$$= \frac{1}{N}(2(X^T X + \alpha^* I)w - 2X^T Y) = 0$$

Аналитическое решение с регуляризацией

Вспомним задачу оптимизации среднеквадартичной ошибки в матричном виде:

$$Q(a, X) = (Xw - Y)^T (Xw - Y) + \alpha w^T w \to min_w$$

Таким образом теперь новое аналитическое решение:

$$w^* = (X^T X + \alpha I)^{-1} X^T Y$$

Вопрос: Почему теперь матрица всегда обратима?

Об одном полезном свойстве L_1- регуляризации

Все ли признаки нужны при решении задачи?

- Некоторые признаки могут быть лишними и никак не относиться решению задачи;
- Избыточные признаки могут привносить дополнительные вычислительные расходы, поэтому чем их меньше тем быстрее предсказание;
- Если признаков больше чем объектов, то существуют различные решения;

Во всех указанных случаях уменьшение числа признаков будет иметь скорее положительный эффект

Об одном полезном свойстве L_1- регуляризации

Утверждение: В результате обучения модели с L_1- регуляризацией происходит зануление некоторых весов, что равносильно отбору признаков

Доказательство данного факта зиждется на том, что задача:

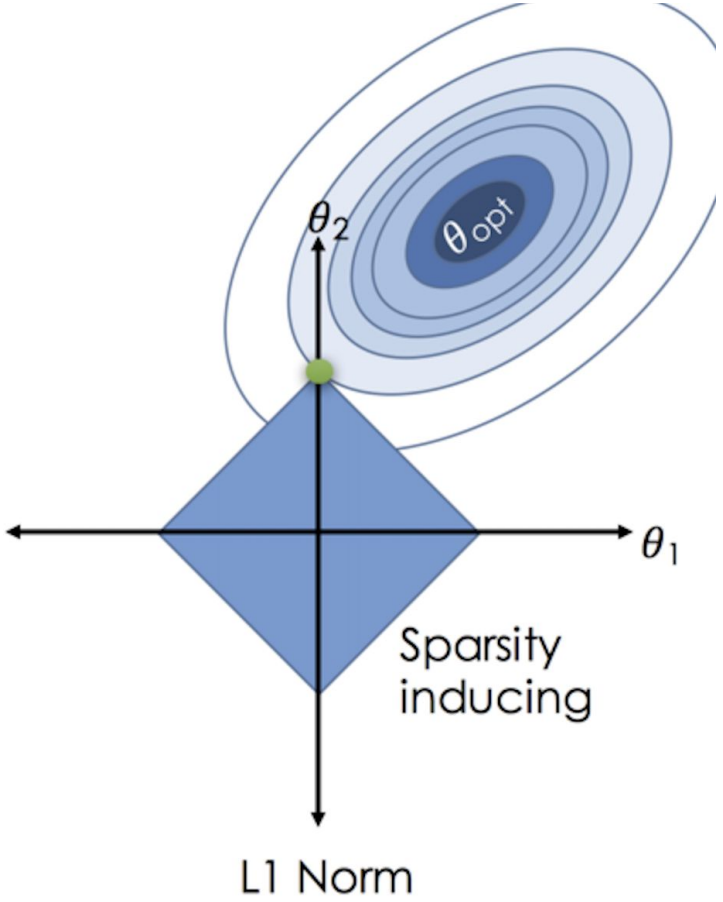
$$Q(a, X) + \alpha||w||_1 \rightarrow min_w$$

Эквивалентна

$$\begin{cases} Q(a, X) \rightarrow min_w \\ ||w||_1 \leq C \end{cases}$$

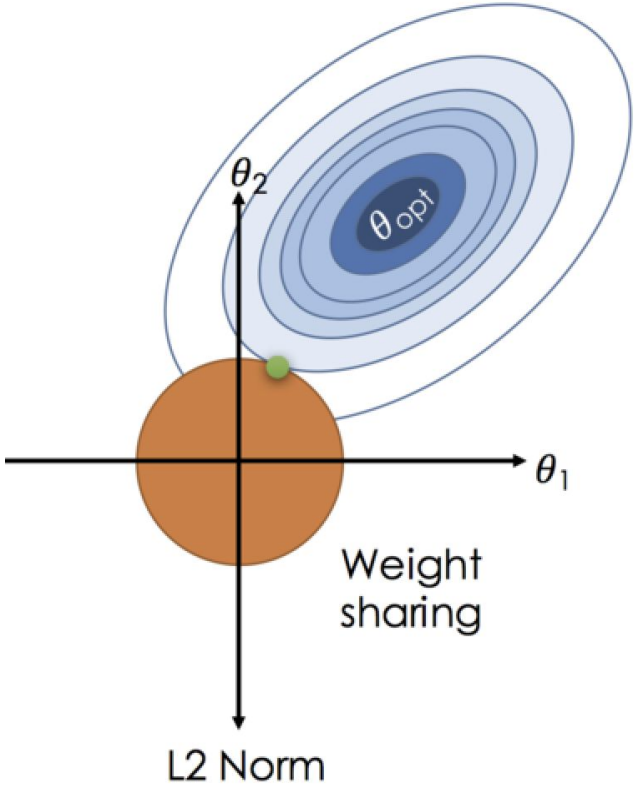
Нарисуем линии уровня и ограничения из системы

В большинстве случаев точка пересечения линии уровня и геометрического места точек ограничения системы $|w|_1 \leq C$ будет происходить по одной из координатных осей, что соответствует занулению некоторых весов.



Линии уровня для L_2- регуляризации

В случае с L_2 - регуляризацией точка пересечения линии уровня и геометрического места точек $||w||_2 \leq C$ происходит равномерно по сфере $||w||_2$, поэтому L_2 - регуляризация, скорее, уменьшит абсолютные значения как w_1 так и w_2 нежели занулит один из них



Разреженные модели

Модели в которых часть весов равна 0 или близко к нему, называются разреженными моделями

- L_1 - регуляризация зануляет часть весов, что делает модели разреженными

Гиперпараметры модели

При рассмотрении регуляризации был введён дополнительный параметр α

$$Q^{reg}(a, X) = Q(a, X) + \alpha \cdot R(w)$$

Данный параметр настраивается вручную до этапа обучения и носит название гиперпараметра модели.

Важно понимать отличие между тем, что является параметром, а что является гиперпараметром.

Гиперпараметры и параметры модели

- Параметры модели - величины, настраивающиеся по обучающей выборке (например веса w в линейной регресси)

- Гиперпараметры модели - величины, контролирующие процесс обучения. Поэтому они не могут быть настроены по обучающей выборке. Например, коэффициент α в регуляризации или η шаг обучения в градиентном спуске

Вопрос: как подбирать гиперпараметры?

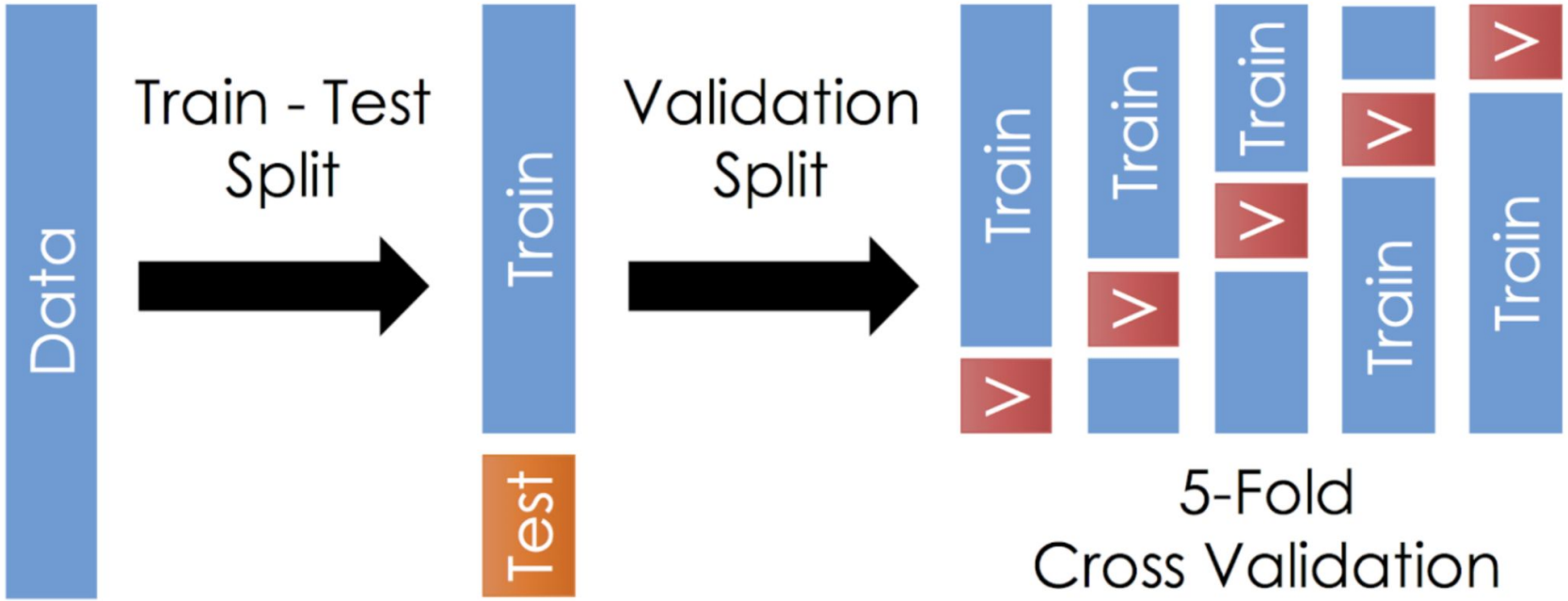
Подбор гиперпараметров

Как правило подбор гиперпараметров происходит по валидационной выборке. Но при таком подходе получается, что валидация начинает играть роль обучающего множества только для множества гиперпараметров.

Чтоб такого не происходило исходное обучающее множество данных, делят на три подмножества:

- Train - там где происходит поиск параметров модели w
- Validation - там где происходит замер качества алгоритмов с различными значениями гиперпараметров
- Test - финальный замер модели

Подбор гиперпараметров



В каких случаях линейная регрессия работает хорошо