МАШИННОЕ ОБУЧЕНИЕ

ВВЕДЕНИЕ В МАШИННОЕ ОБУЧЕНИЕ

МИХАИЛ ЛИПКОВИЧ

ЧТО ТАКОЕ МАШИННОЕ ОБУЧЕНИЕ?

ОПРЕДЕЛЕНИЕ БЕЗ ПАФОСА

- Набор алгоритмов, которые на основе имеющихся данных находят (или пытаются найти) такие параметры математической модели, при которых достигается экстремум заданного функционала

ОПРЕДЕЛЕНИЕ БЕЗ ПАФОСА

- Набор алгоритмов, которые на основе имеющихся данных находят (или пытаются найти) такие параметры математической модели, при которых достигается экстремум некоторого заданного функционала

- Имеющиеся данные - dataset. Ту часть на которой обучаемся обычно называют тренировочные данные / training set
- Поиск параметров обычно итеративен. Называется обучением / learning / fitting
- Функционал - функция потерь / функционал качества / loss function

ЗАМЕЧАНИЯ

- Функционал выбирается исходя из того чтобы задачу можно было решить, а не исходя из того что нам надо для бизнеса (см. метрики качества)
- Алгоритмы оптимизируют функционал на traning set, но вообще говоря мы хотим чтобы они работали на тех данных что у нас нет (см. переобучение/overfitting)
- У нашей модели могут быть "метапараметры", которые не обучаются (см. validation set)

ДАНО:

- Набор данных ("семплы")

$\{X_i, y_i\}, i = 1, \ldots, n$ где

X_i - вектора размерности m (фичи / features)

y_i могут быть вещественными числами (задача регрессии) или принимать значения из множества {1..k} - задача классификации

- Функция потерь

$\frac{1}{n} \sum_{i=1}^{n} J(y_i, y'_i)$ где y'_i наше предсказание по X_i

ТРЕБУЕТСЯ:

- Найти такую функцию $f(X_i, y_i, w)$

  что будет достигаться минимум J по всем семплам

  w - настраиваемые параметры модели

ПРИМЕР: ЛИНЕЙНАЯ РЕГРЕССИЯ

- X_i - вектор с параметрами "рост человека", "размер талии", "размер груди"
  y_i - вес человека

$$X_1 = (180, 70, 100) \quad y_1 = 82$$
$$X_2 = (163, 57, 71) \quad y_2 = 61$$
$$X_3 = (175, 99, 102) \quad y_2 = 96$$

ПРИМЕР: ЛИНЕЙНАЯ РЕГРЕССИЯ

- X_i - вектор с параметрами "рост человека", "размер талии", "размер груди"
  y_i - вес человека

$$X_1 = (180, 70, 100) \quad y_1 = 82$$
$$X_2 = (163, 57, 71) \quad y_2 = 61$$
$$X_3 = (175, 99, 102) \quad y_2 = 96$$

- Функция потерь $J(y_i, y_i') = (y - y_i')^2$

ПРИМЕР: ЛИНЕЙНАЯ РЕГРЕССИЯ

- X_i - вектор с параметрами "рост человека", "размер талии", "размер груди"
  y_i - вес человека

$$X_1 = (180, 70, 100) \quad y_1 = 82$$
$$X_2 = (163, 57, 71) \quad y_2 = 61$$
$$X_3 = (175, 99, 102) \quad y_2 = 96$$

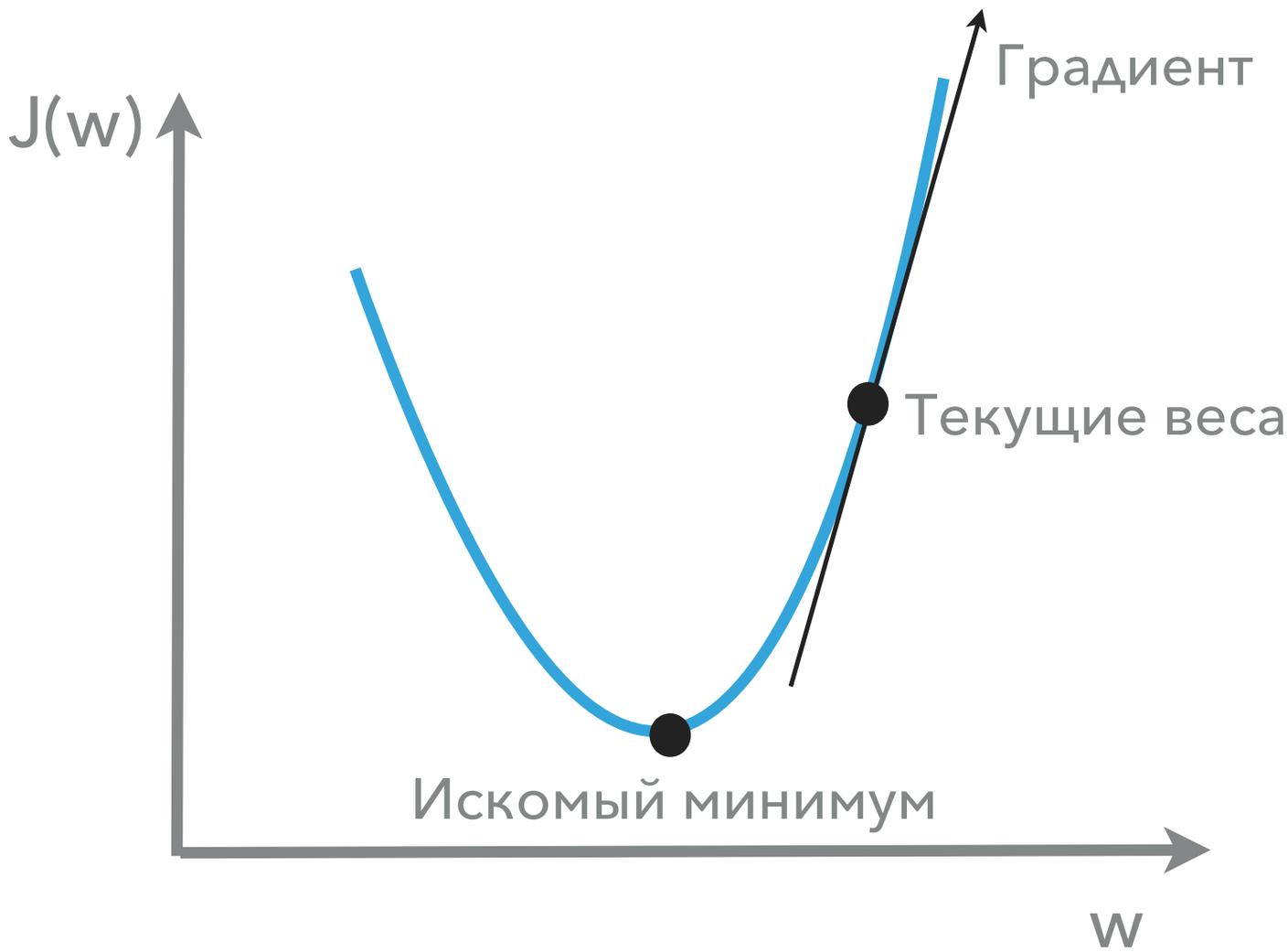
- Функция потерь $J(y_i, y_i') = (y - y_i')^2$

- Модель $f(X_i, w) = w_0 + \sum_{j=1}^{m} w_j X_j$

BATCH GRADIENT DESCENT

w - наши обучаемые параметры

J(w) - функция потерь, рассчитанная на *всем* датасете



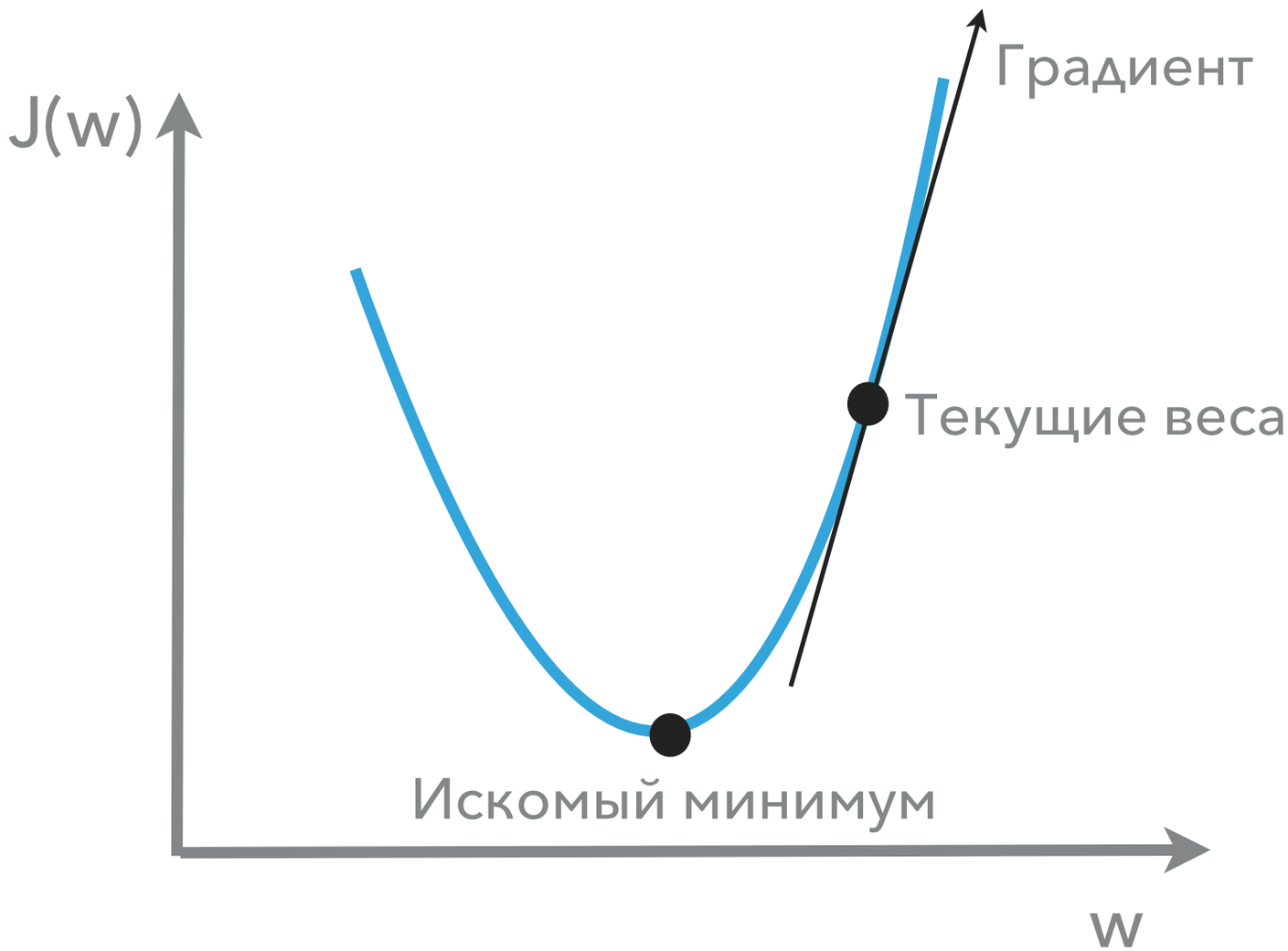
ЗАМЕЧАНИЯ

- Каждая итерация требует обхода всего тренировочного множества
- Теоретически будет давать точное направление к минимуму и сойдется в случае выпуклой функции ошибки
- Может застрять в локальном минимуме

STOCHASTIC GRADIENT DESCENT

w - наши обучаемые параметры

J(w) - функция потерь, рассчитанная на *одном случайном* семпле



ЗАМЕЧАНИЯ

- Каждая итерация требует расчета лишь одного градиента
- Есть промежуточный вариант - mini-batch gradient descent
- Теоретически сходится "почти гарантированно" в случае выпуклой функции
- Может помочь выбраться из локального минимума
- В комбинации с адаптивным learning rate используется в Deep Learning (RMSProp, Adam и т.д.)

ОБЩИЕ ЗАМЕЧАНИЯ

- Обычно решают задачу бинарной классификации (два класса)

ОБЩИЕ ЗАМЕЧАНИЯ

- Обычно решают задачу бинарной классификации (два класса). Все остальное сводится к ней (1 против всех или попарно)

ОБЩИЕ ЗАМЕЧАНИЯ

- Обычно решают задачу бинарной классификации (два класса). Все остальное сводится к ней (1 против всех или попарно)
- Чтобы применять градиентный спуск нам нужна "гладкая" функция потерь что дает не самые интуитивные варианты. Примеры:

$$\log(1 + \exp(-y\mathbf{w}^T\mathbf{x})), \quad y \in \{-1, +1\}$$ Logistic Regression

$$\max\{0, 1 - y\mathbf{w}^T\mathbf{x}\}, \quad y \in \{-1, +1\}$$ Linear SVM

А КАКИЕ ФУНКЦИИ ПОТЕРЬ БЫЛИ БЫ ИНТУИТИВНЫМИ?

- Количества неправильных предсказаний каждого из классов
- Эти количества относительно размеров классов
- Всякие показатели какой класс за какой чаще всего принимаем
- .....

А КАКИЕ ФУНКЦИИ ПОТЕРЬ БЫЛИ БЫ ИНТУИТИВНЫМИ?

Но на таких функциях не обучиться :(

Поэтому обучают на одну функцию потерь, а реально анализируют качество по другим

Эти функции которые нас реально интересуют называют метриками качества

ОБОЗНАЧЕНИЯ

Считаем что решаем задачу бинарной классификации (классы 0 и 1)

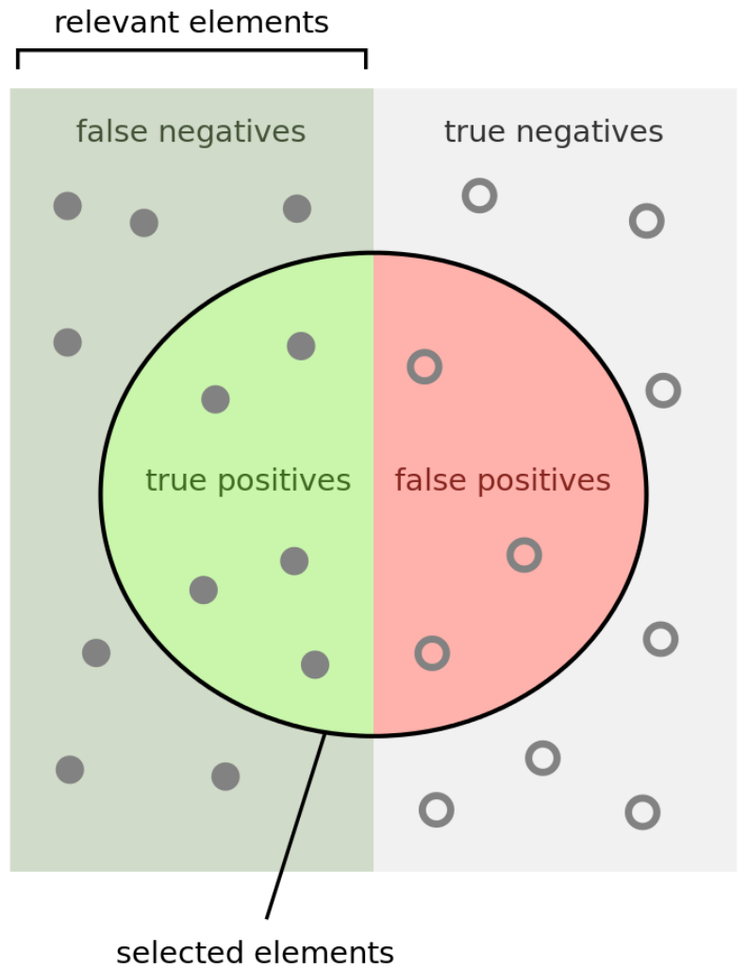
- Класс 0 принято называть негативным, а класс 1 - позитивным
- Positive (P) / Negative (N) - количество позитивных / негативных примеров
- True Positive (TP) / True Negative (TN) - количество правильно распознанных позитивных / негативных примеров
- False Positive (FP) / False Negative (FN) - количество неправильно распознанных позитивных / негативных примеров

МЕТРИКИ КЛАССИФИКАЦИИ

- Recall: TP / P
- Precision: TP / (TP + FP)
- Accuracy: (TP + TN) / (P + N)
- F1: 2 * (precision * recall) / (precision + recall)
- ROC/AUC

- Любые метрики которые могут идти от бизнеса!

ХОРОШАЯ КАРТИНКА ИЗ ВИКИПЕДИИ



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

https://en.wikipedia.org/wiki/Precision_and_recall

А В ЗАДАЧЕ РЕГРЕССИИ?

Метрика что мы рассматривали для регрессии - MSE (Mean Squared Error) вполне себе интуитивна, но есть еще метрики:

А В ЗАДАЧЕ РЕГРЕССИИ?

Метрика что мы рассматривали для регрессии - MSE (Mean Squared Error) вполне себе интуитивна, но есть еще метрики:

- RMSE - корень от MSE
- MAE (Mean Absolute Error) - MSE с модулем вместо квадрата
- R2 (коэффициент детерминации) - посмотреть самостоятельно
- И тоже метрики от бизнеса

ЧТО ИМЕЕМ:

- У нас есть наш датасет
- На нем с помощью градиентного спуска и потенциально не интересующей нас функции потерь обучили какую-то модель
- Посчитали интересующую нас метрику на датасете
-

ЧТО ИМЕЕМ:

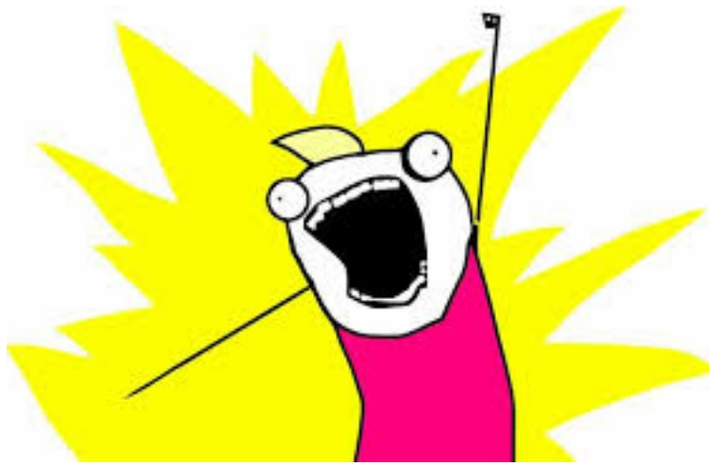
- У нас есть наш датасет
- На нем с помощью градиентного спуска и потенциально не интересующей нас функции потерь обучили какую-то модель
- Посчитали интересующую нас метрику на датасете
- Все ли хорошо в этой схеме?

ЧЕГО МЫ ХОТИМ НА САМОМ ДЕЛЕ:

- Хорошие показатели метрик на данных которые наша модель не видела!

ЧЕГО МЫ ХОТИМ НА САМОМ ДЕЛЕ:

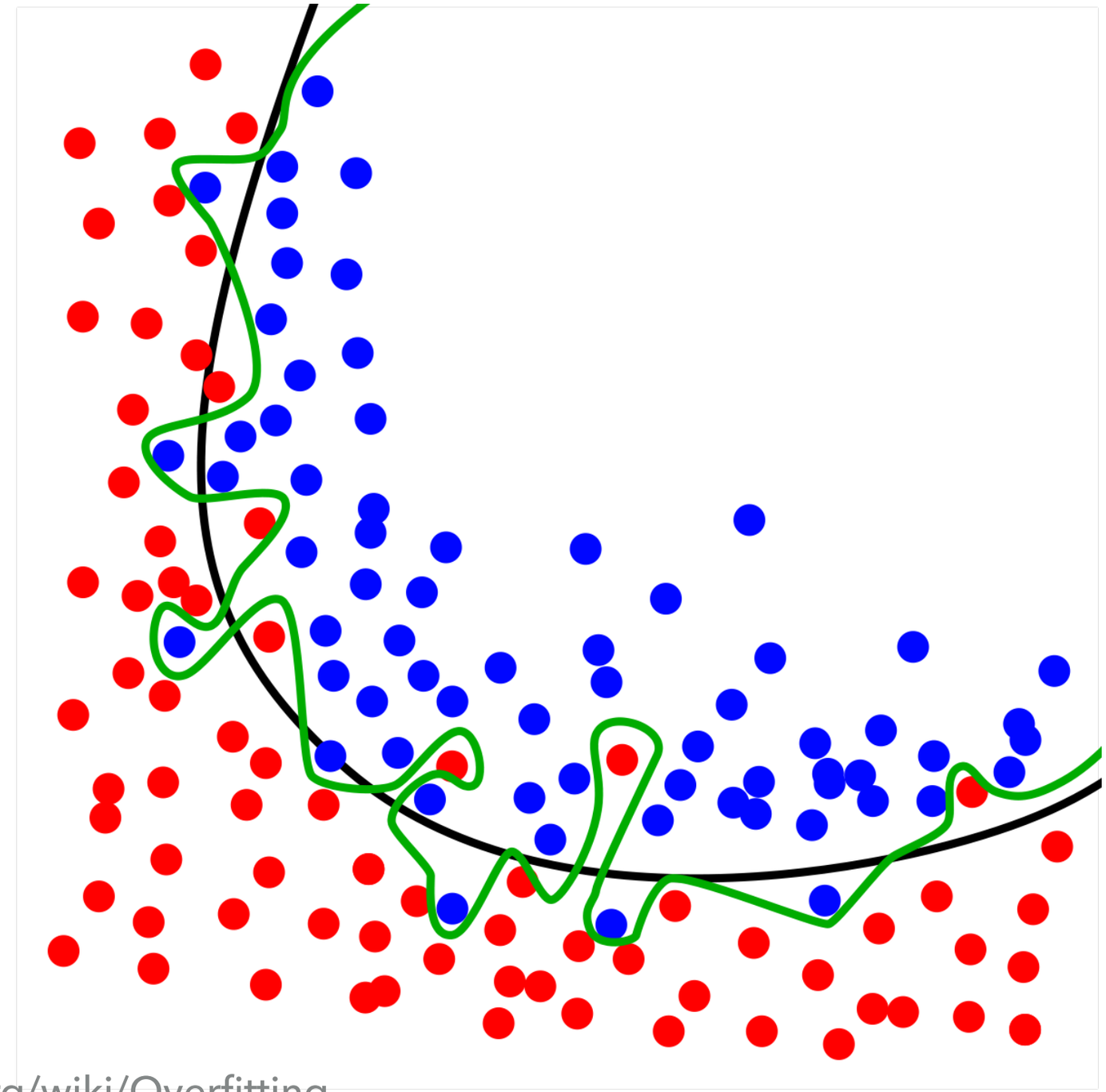
- Хорошие показатели метрик на данных которые наша модель не видела!



ПЕРЕОБУЧЕНИЕ

- Как нам гарантировать что наша модель действительно научится обобщать?
- Потенциально она может идеально приблизить train set и давать на нем хорошее качество, но быть совершенно неработающей на test set
- Эта проблема называется переобучением (overfitting)

И СНОВА КАРТИНКА ИЗ ВИКИ



https://en.wikipedia.org/wiki/Overfitting

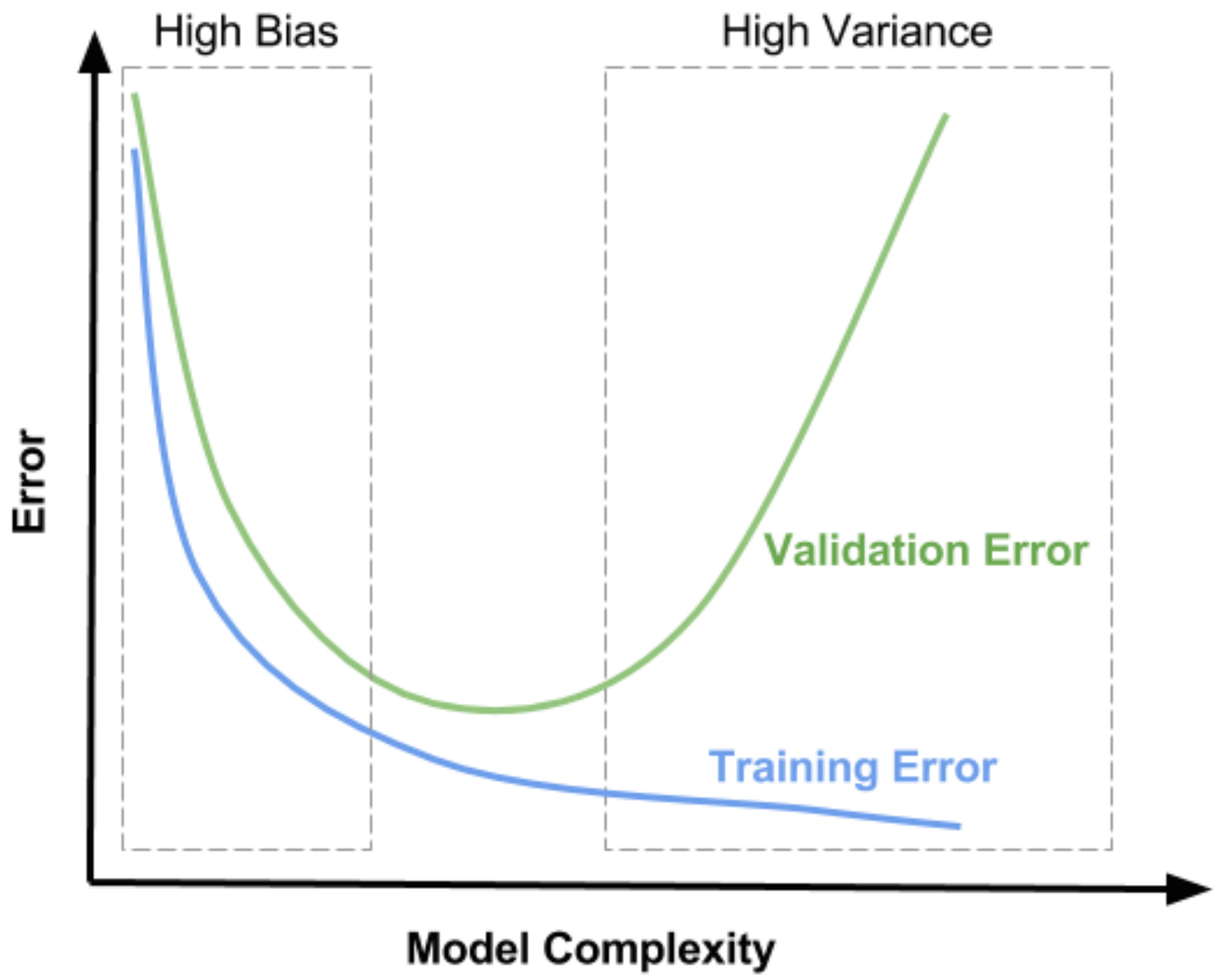
BIAS–VARIANCE TRADEOFF

У нас есть два источника проблем при попытке обобщения:

- High bias: проблема при которой модель не уловила связи что есть в данных (underfitting) - модель слишком слабая
- High variance: модель перестаралась и шум в данных восприняла как некоторые связи (overfitting) - модель слишком сложная

Решение этих проблем находится в противоречии друг с другом

УКРАДЕНО С САЙТА ИТМО



http://neerc.ifmo.ru/wiki

НАДО КОНТРОЛИРОВАТЬ СЛОЖНОСТЬ МОДЕЛИ

- Со слишком простой моделью все относительно просто: если качество нас не устраивает, то модель простовата (а может мы хотим многого)
- Надо ограничивать сложность модели сверху
- Есть разные приемы ограничения сложности модели в зависимости от природы модели, но наиболее универсальный подход - регуляризация

РЕГУЛЯРИЗАЦИЯ

- У нас была функция потерь. Например,

$$J(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - w^t X_i)^2$$

- Фактически сложность модели кроется в весах w (это можно даже доказать)

- Давайте тогда добавим на них ограничение:

$$J(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - w^t X_i)^2 + \lambda \|w\|^2$$

РЕГУЛЯРИЗАЦИЯ

$$J(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - w^t X_i)^2 + \lambda \|w\|^2$$

Параметр лямбда варьирует насколько сильно мы хотим штрафовать модель за сложность

Там может быть любая норма. Наиболее популярные - L1 и L2. Их так и называют: L1-регуляризация и L2-регуляризация

ElasticNet - линейная комбинация норм L1 и L2

А КАК ВЫБРАТЬ ПАРАМЕТР РЕГУЛЯРИЗАЦИИ?

А КАК ВЫБРАТЬ ПАРАМЕТР РЕГУЛЯРИЗАЦИИ?

- Это первый метапараметр с которым столкнулись
- Он не относится к параметрам которые являются настраиваемыми. Но выбрать хочется
- Обычно для таких параметров используют подход Grid Search: рассматривают некоторое множество параметров и просто перебирают их пока не получат наилучшее качество

А КАК ВЫБРАТЬ ПАРАМЕТР РЕГУЛЯРИЗАЦИИ?

- Это первый метапараметр с которым столкнулись
- Он не относится к параметрам которые являются настраиваемыми. Но выбрать хочется
- Обычно для таких параметров используют подход Grid Search: рассматривают некоторое множество параметров и просто перебирают их пока не получат наилучшее качество. Обычно это {0.01, 0.1, 1, 5}

ПОПРАВИМ СТРАТЕГИЮ

- Разобьем весь наш датасет на два: тренировочный (на котором будет обучать модель) - training set и тестовый (на котором будет оценивать наши метрики) - test set
- Обычно разбивают в соотношении 80 на 20, но вообще говоря это зависит от того сколько данных есть
- Разбивают полностью случайно или стратифицированно: сначала бьем на группы, а потом из каждой группы семплируем случайно
- Выбираем параметр регуляризации, который даст наилучшее качество на test set

ЧТО ИМЕЕМ [2]

- У нас есть наш датасет
- Разбили его на train set и test set
- На train set с помощью градиентного спуска и потенциально не интересующей нас функции потерь обучили какую-то модель
- Посчитали интересующую нас метрику на test set
- Повторили предыдущие два шага для разных параметров регуляризации

ЧТО ИМЕЕМ [2]

- У нас есть наш датасет
- Разбили его на train set и test set
- На train set с помощью градиентного спуска и потенциально не интересующей нас функции потерь обучили какую-то модель
- Посчитали интересующую нас метрику на test set
- Повторили предыдущие два шага для разных параметров регуляризации
- Теперь все хорошо?

НЕТ, НЕ ХОРОШО

- Идеологически это все-таки неверный подход: получилось что мы подобрали параметр под наш конкретный test set
- Лучше все-таки совсем не трогать test set и использовать только для финальной оценки качества
- Тогда от train set придется снова отколоть кусок (еще процентов 20), который называют validation set - по нему будем подбирать параметр регуляризации

ЧТО ИМЕЕМ [3]

- У нас есть наш датасет
- Разбили его на train set, validation set и test set
- Задали множество параметров регуляризации которые хотим проверить
- Для каждого параметра регуляризации:
  - Обучаем на train set с помощью градиентного спуска и функции потерь с регуляризацией нашу модель
  - Смотрим на интересующую нас метрику на validation set
- Выбрали параметр регуляризации при котором качество было лучшим на validation set
- С этим параметром обучились на всем train set
- Посчитали качество полученной модели на test set - это итоговое качество нашей модели

ТЕПЕРЬ МОЖЕМ ИСПОЛЬЗОВАТЬ МОДЕЛЬ СПОКОЙНО?

ТЕПЕРЬ МОЖЕМ ИСПОЛЬЗОВАТЬ МОДЕЛЬ СПОКОЙНО?

А что если мы очень удачно выбрали test set, и на самом деле модель слабее чем мы думали?

А может мы сравниваем несколько моделей, и для одной из них этот test set был удачным, а для второй нет?

Та же проблема и с validation set

КРОСС-ВАЛИДАЦИЯ

- Для полного спокойствия нужна кросс-валидация
- Есть разные подходы в зависимости от размера датасета и того что нас больше волнует, но общая суть примерно такая:
  - Бьем весь датасет на несколько частей (например, 5)
  - Повторяем следующую процедуру: одну часть откладываем как test set, а на оставшихся четырех обучаемся как на train set. Считаем качество
  - Итоговое качество - усреднение метрик на кусках

ЧТО ИМЕЕМ [4]

ЧТО ИМЕЕМ [4]

- Ну вы поняли