ТЕХНОСФЕРА

Лекция 2
Метрические алгоритмы

Всеволод Викулин

Содержание

1. Знакомимся с метрическими алгоритмами
2. Учимся оценивать качество и подбирать параметры
3. Улучшаем алгоритм

Часть 1
Ваш первый алгоритм

Напоминание

Объект описывается вектором его наблюдаемых характеристик (признаков) $x \in X$ и скрытых характеристик $y \in Y$ (целевая переменная).

Существует некоторая функция $f : X \to Y$

Задача: имея ограниченный набор объектов, построить функцию $a : X \to Y$, приближающую f на всем множестве объектов

$\{x_1, \ldots, x_N\} = X_{train}, \{y_1, \ldots, y_N\} = Y_{train}$

Обучение с учителем — известны X_{train}, Y_{train}

1) Классификация — $Y = \{1, \ldots, M\}$

2) Регрессия — $Y = \mathbb{R}, Y = \mathbb{R}^{\mathbb{M}}$

Напоминание

Функция потерь $L(a, x, y)$ — неотрицательная функция, показывающая величину ошибки алгоритма a на объекте x с ответом y.

Функционал качества

$Q(a, X, Y) = \frac{1}{N}\sum_{i=1}^{N} L(a, x_i, y_i), x_i \in X, y_i \in Y$

Принцип минимизации эмпирического риска:

$a^* = \underset{A}{\operatorname{argmin}}\, Q(a, X_{train}, Y_{train}), A$ — семейство алгоритмов.

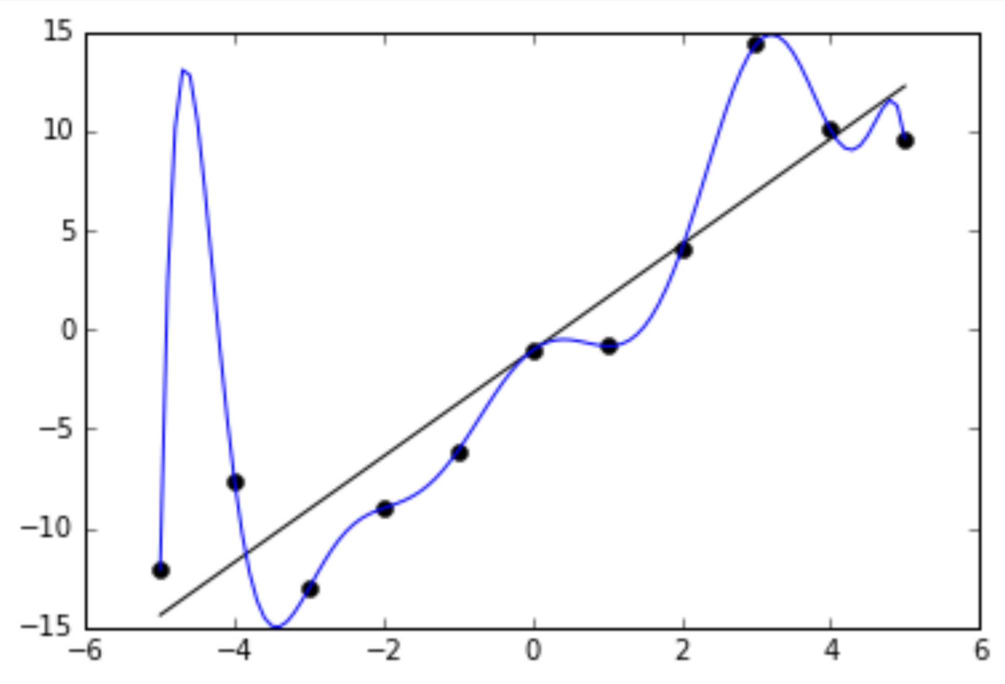
Переобучение — $Q(a, X_{train}, Y_{train}) \ll Q(a, X_{test}, Y_{test})$

Формула обучения:

Learning = Representation + Evaluation + Optimization

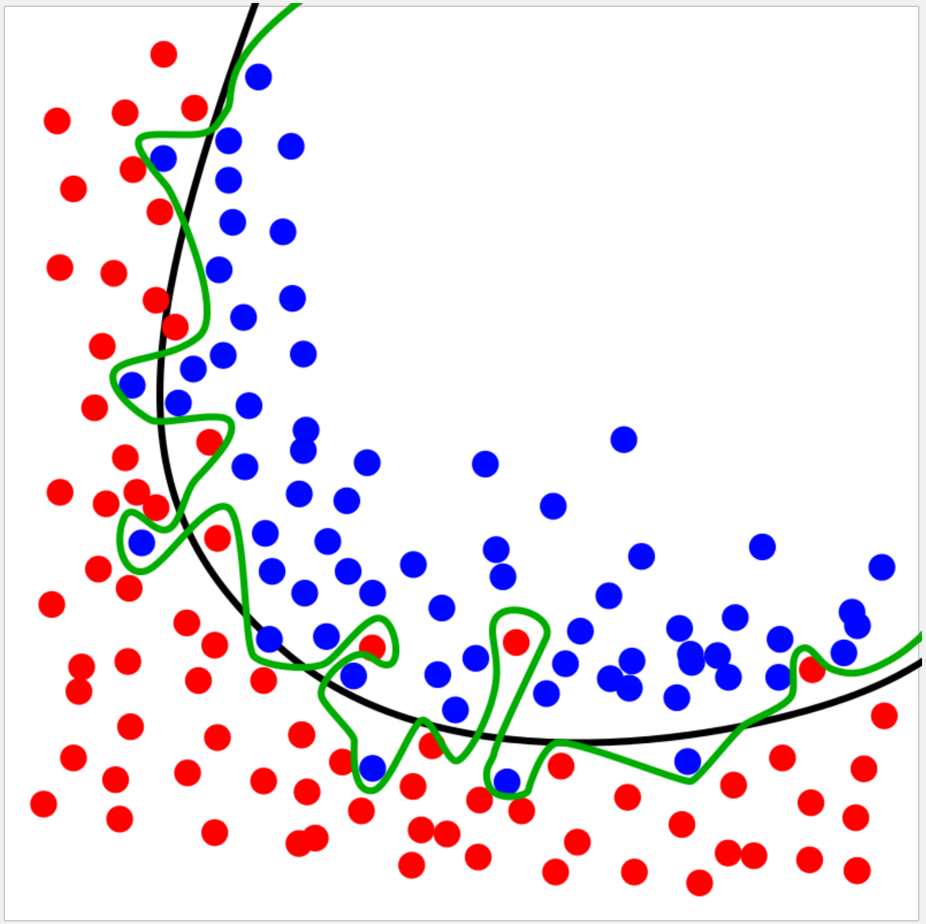
Источник: homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

Пример переобучения



Источник: en.wikipedia.org/wiki/Overfitting

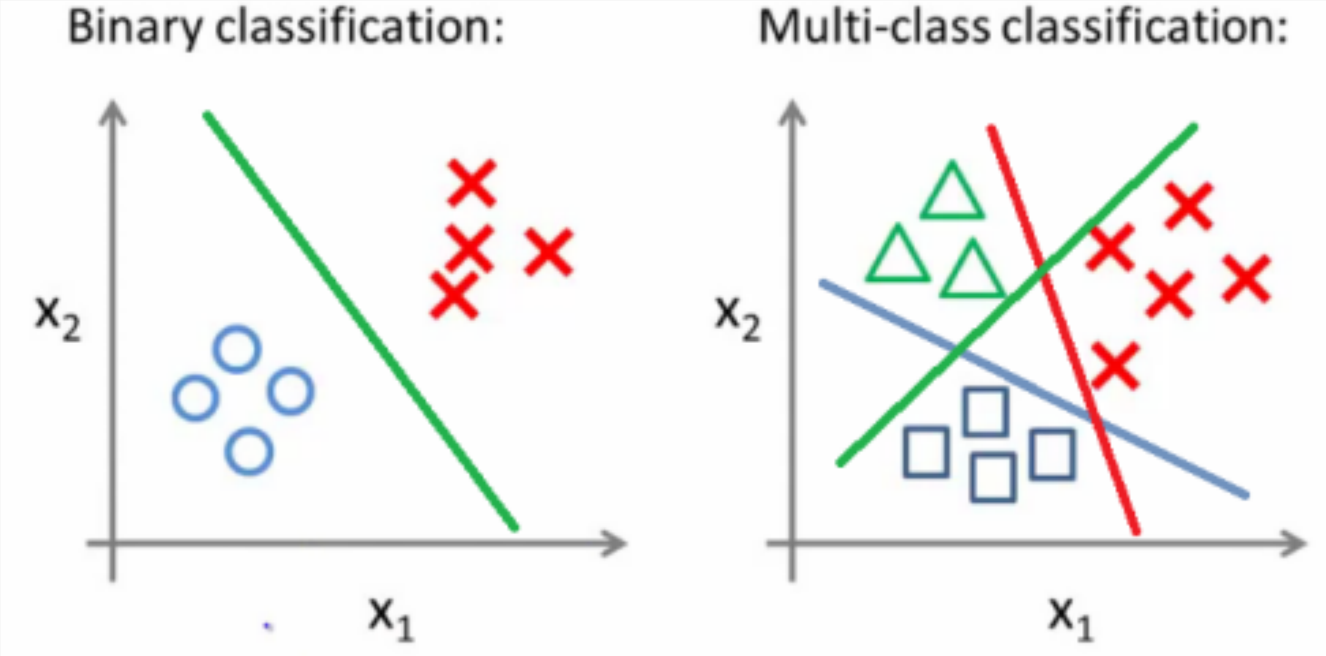
Пример переобучения



Источник: en.wikipedia.org/wiki/Overfitting

Гипотеза компактности

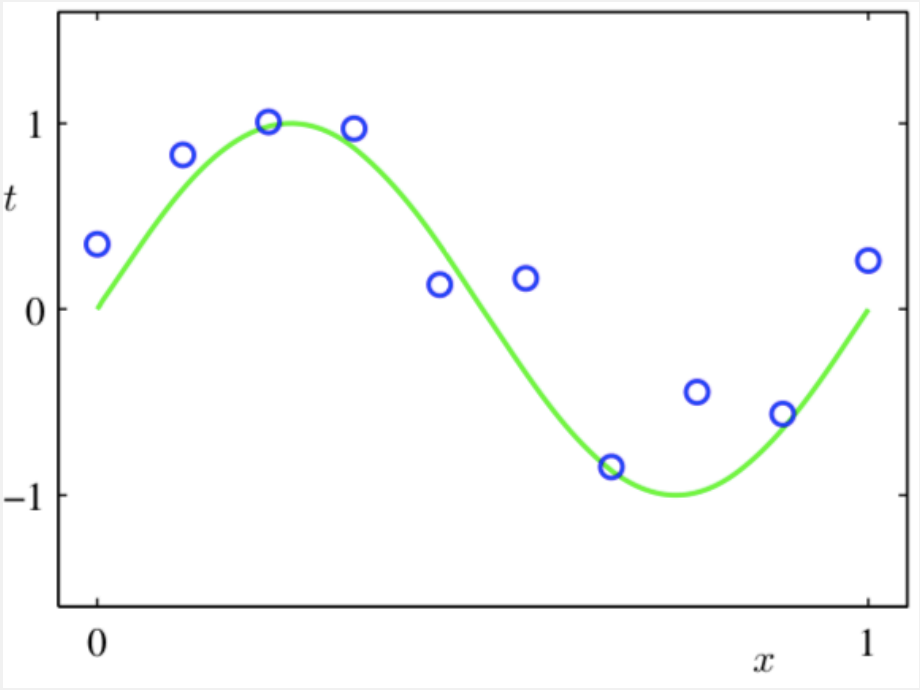
Гипотеза:
Похожие объекты лежат в одном классе



Источник: medium.com/@b.terryjack

Гипотеза непрерывности

Гипотеза:
Похожие объекты имеют похожий ответ



Источник: Bishop

Метрические алгоритмы

Пусть обучающая выборка размера N.

Общая формула для классификации:

$$a(x, X_{train}) = \underset{c}{\operatorname{argmax}} \sum_{i=1}^{N} w(x, x_i) I[y_i = c], x_i \in X_{train}$$

Общая формула для регрессии:

$$a(x, X_{train}) = \frac{\sum_{i=1}^{N} w(x, x_i) y_i}{\sum_{i=1}^{N} w(x, x_i)}, x_i \in X_{train}$$

Непараметрический, ленивый алгоритм

Задаем веса

- Если ненулевой вес только у ближайшего объекта, то алгоритм называют алгоритмом ближайшего соседа
- Если ненулевые веса для k ближайших объектов, то алгоритм называют алгоритмом k ближайших соседей (k-nearest neighbors, knn).

Пусть x_i — i-тый ближайший сосед объекта x

- $w(x, x_i) = 1/k$
- $w(x, x_i) = \frac{k + 1 - i}{k}$
- $w(x, x_i) = \alpha^i, \alpha \in (0,1)$

Задаем веса

Проблема: в прошлом варианте никак не учитываем величину расстояния.

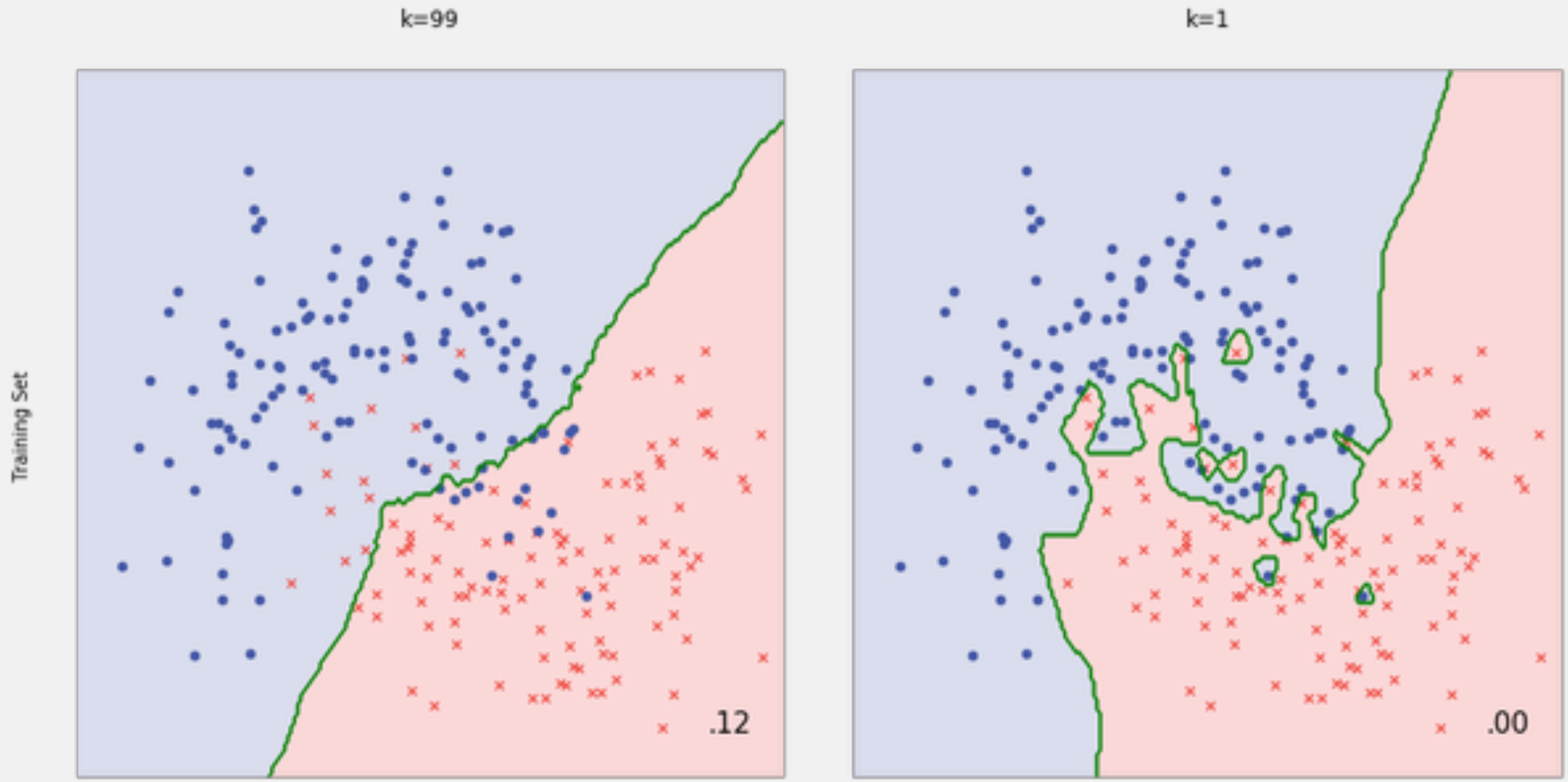
$$w(x, x_i) = K(\rho(x, x_i)),$$

где $K(x)$ — любая монотонно убывающая функция.

Примеры:

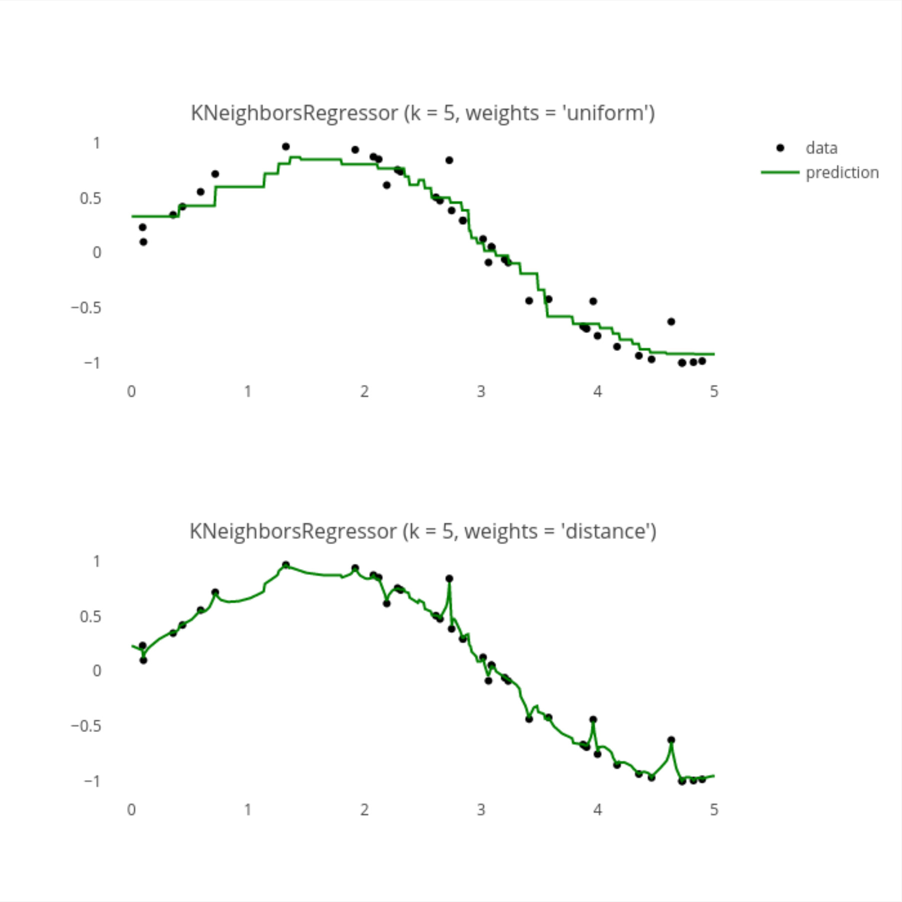
- $K(x) = \dfrac{1}{x + \beta}$
- $K(x) = \exp(-x)$
- $K(x) = \alpha^x, \alpha \in (0,1)$

Пример классификации



Источник: cambridgecoding.wordpress.com/2016/03/24/

Пример регрессии



Источник: plot.ly/python

Часть 2
Оцениваем качество, выбираем параметры

Структурные параметры

? *Параметры метрических алгоритмов настраивать на обучающей выборке? На тестовой? Почему?*

$$Q(a, X, Y) = \frac{1}{N}\sum_{i=1}^{N} L(a, x_i, y_i), x_i \in X, y_i \in Y$$

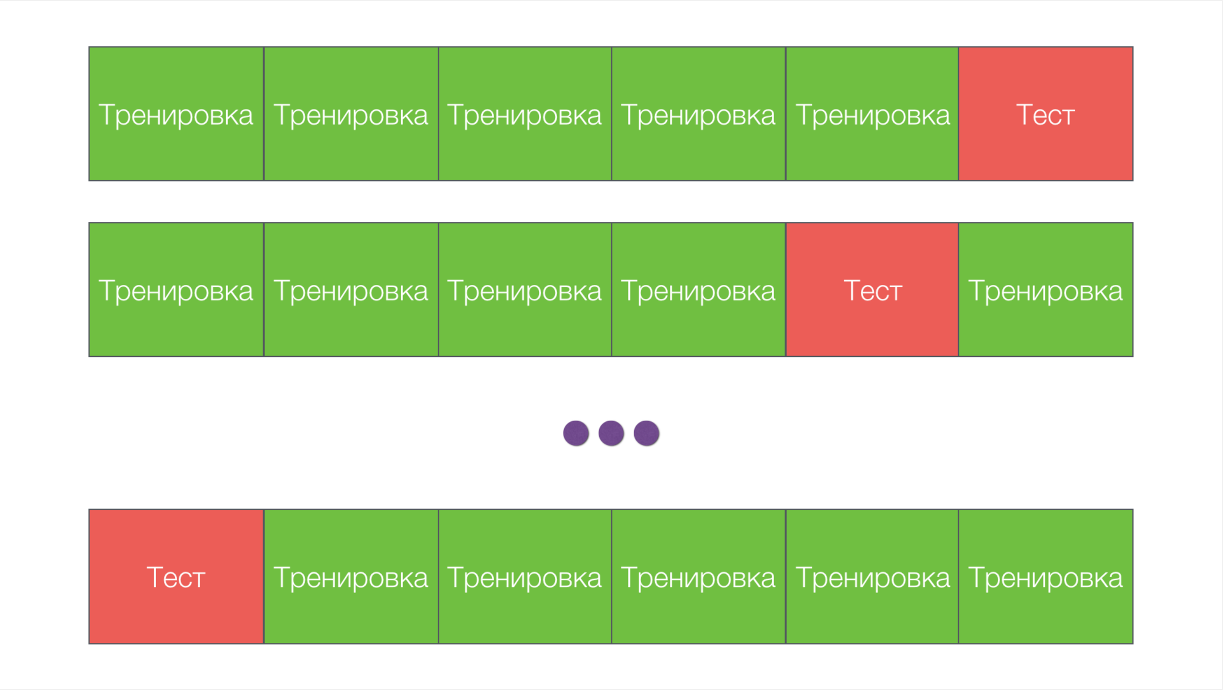
! *Переобучение возникает из-за излишней сложности модели*

Параметры, которые нельзя настраивать на обучающей выборке, будем называть **структурными**.

Обучающую выборку нужно разделить на обучающую и **валидационную**. На ней настраиваем структурные параметры!

Скользящий контроль

Качество зависит от объектов в валидации!
Решение — скользящий контроль (cross-validation).
В пределе, когда только 1 объект в тесте — LOO (leave one out).



Оцениваем качество регрессии

$$Q(a, X, Y) = \frac{1}{N}\sum_{i=1}^{N} L(a, x_i, y_i), x_i \in X, y_i \in Y$$

Функции потерь:

1. **Квадратичная**(Q — MSE)

$$L(a, x, y) = (a(x) - y)^2$$

2. **Абсолютная**(Q — MAE)

$$L(a, x, y) = |a(x) - y|$$

3. **Логарифмическая** (Q — MSLE)

$$L(a, x, y) = (\log(a(x) + 1) - \log(y + 1))^2$$

4. **Абсолютная-процентная** (Q —MAPE)

$$L(a, x, y) = \frac{|a(x) - y|}{y}$$

Оцениваем качество классификации

- **Accuracy** (точность) — процент правильно классифицированных объектов $L(a, x, y) = [a(x) = y]$
- **Precision** (аккуратность) —процент правильно классифицированных объектов класса 1 среди всех объектов, которым алгоритм присвоил метку 1.
- **Recall** (полнота) — процент правильно классифицированных объектов класса 1 среди всех объектов класса 1
- **F1-score** — среднее гармоническое Precision и Recall

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

?

Тестовая выборка содержит 10 объектов класса 1 и 990 объектов класса 0. Какая точность у константного алгоритма?

?

Почему именно среднее гармоническое?

Не всегда нужно разбивать случайно

Нельзя никогда забывать, какую задачу мы решаем!

Если выборка маленькая, то нужно сохранять баланс классов — stratified валидация.

?

Как сделать валидацию в случае:

- *Спам-фильтра*
- *Предсказания объема продаж на следующую неделю*
- *Предсказания стоимости квартир для всего дома целиком*

Часть 3
Улучшаем алгоритм

Метрики

Аксиомы:

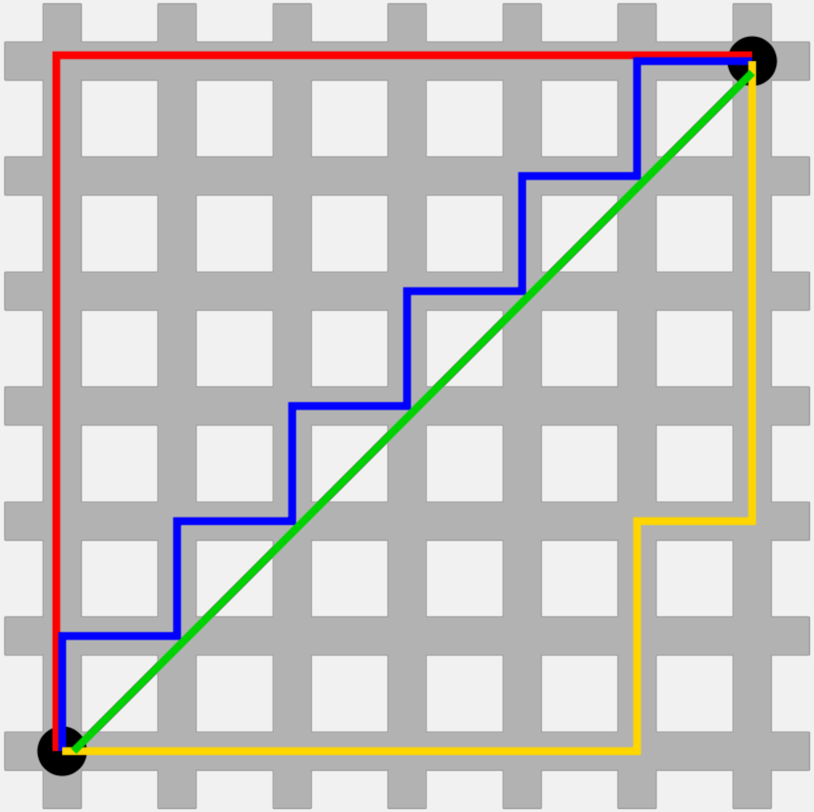
1. $\rho(x, y) = 0$, т.и.т.д $x = y$
2. $\rho(x, y) = \rho(y, x)$
3. $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$

Пусть D — число признаков. Метрика Минковского (при $p \in (0,1)$ не метрика):

$$\rho(x, y) = (\sum_{j=1}^{D} |x_j - y_j|^p)^{\frac{1}{p}}$$

- $p = 2$ — Евклидово расстояние
- $p = 1$ — Манхэттенское расстояние
- $p = \infty$ — Растояние Чебышева (максимальное расстояние между двумя признакми)

Манхэттенское расстояние



Источник: en.wikipedia.org/wiki/Taxicab_geometry

Нормировка признаков

$$\rho(x, y) = (\sum_{j=1}^{D} |x_j - y_j|^p)^{\frac{1}{p}}$$

Нельзя так считать расстояния с признаками разных масштабов!

Два виды нормировки:

1. Стандартизация — $x^j = \dfrac{x^j - mean(x^j)}{std(x^j)}$

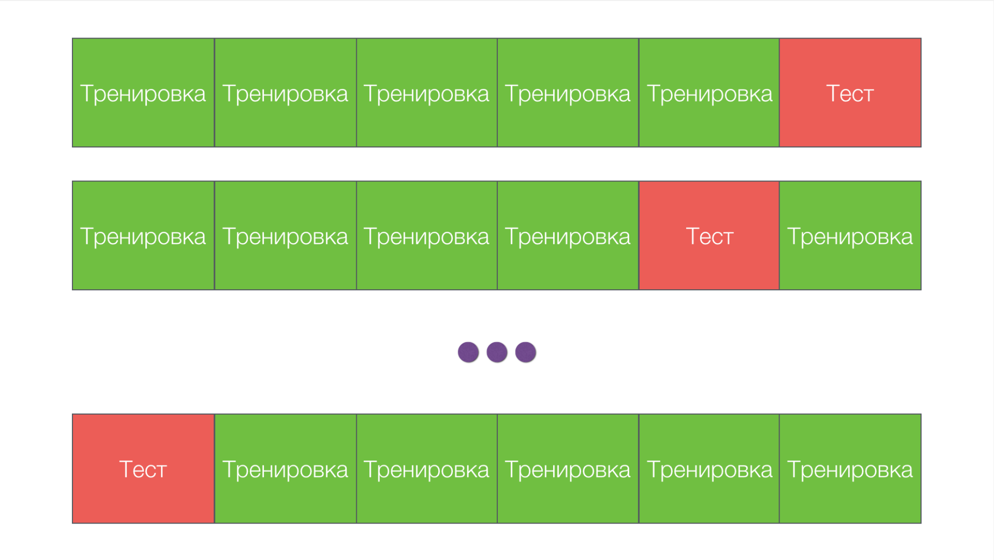
2. Нормализация — $x^j = \dfrac{x^j - min(x^j)}{max(x^j) - min(x^j)}$

? *В каком диапазоне будет лежать признак теперь?*

Расстояния на категориальных признаках

1. Расстояние Хэмминга — число категориальных признаков, которые имеют разные значения.
2. Счетчики — среднее значение признака/целевой переменной с такой категорией.

При кодировании признака с помощью целевой переменной нельзя использовать целевую переменную данного объекта!



Косинусное расстояние

По определению скалярного произведения считаем угол между векторами:

$$\rho(x, y) = \alpha = arccos\frac{x \cdot y}{|x||y|}$$

На практике обычно считают так:

$$sim(x, y) = \frac{x \cdot y}{|x||y|}$$

$$rho(x, y) = 1 - sim(x, y)$$

Косинусное расстояние часто используют для текстов. Почему?

Расстояние Джаккарда

Как померить расстояние между множествами?
Например, предложение — мешок (множество) слов.

$$\rho(X, Y) = 1 - \frac{X \cap Y}{X \cup Y}$$

?

Приведите пример задачи, где удобно использовать расстояние над множествами

Расстояние Левенштейна

Минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую.

$$\rho(kitten, sitting) = 3$$

В каких задачах часто применяется расстояние Левенштейна?

Metric learning

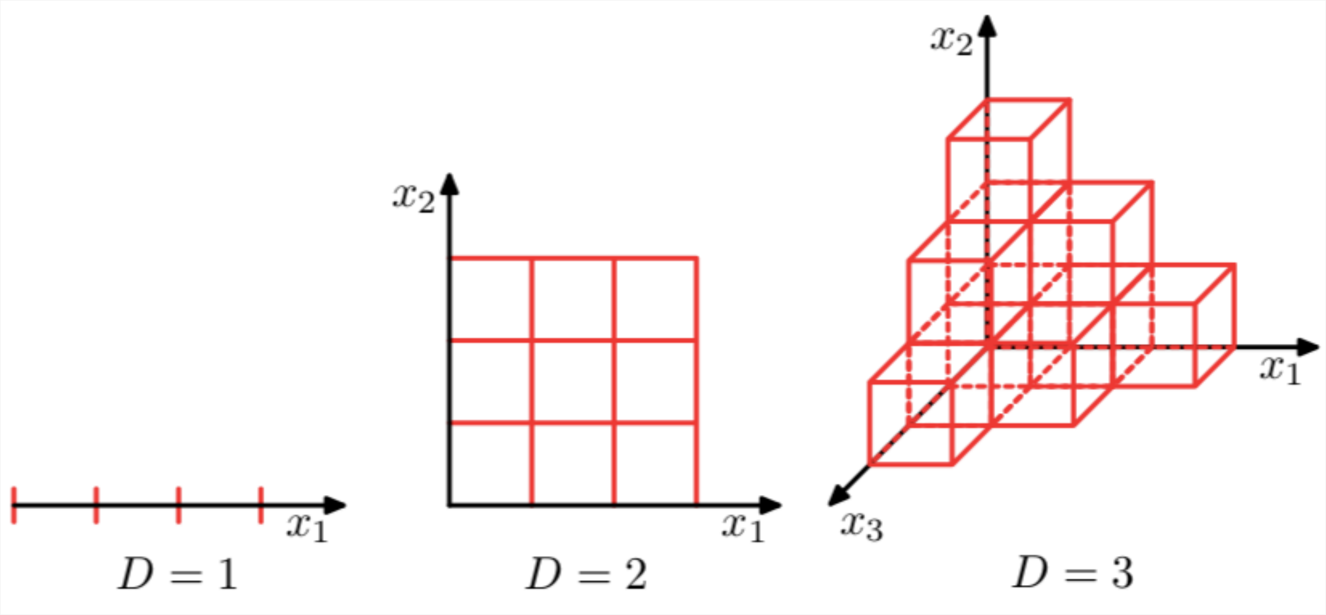
Посчитали расстояние по вещественным признакам, по категориальным, строковое, для множеств... Как теперь все объединить?

$$\rho(x, y) = c_1\rho_1(x, y) + c_2\rho_2(x, y) + \ldots$$

? *Как найти коэффициенты?*

Проклятие размерности

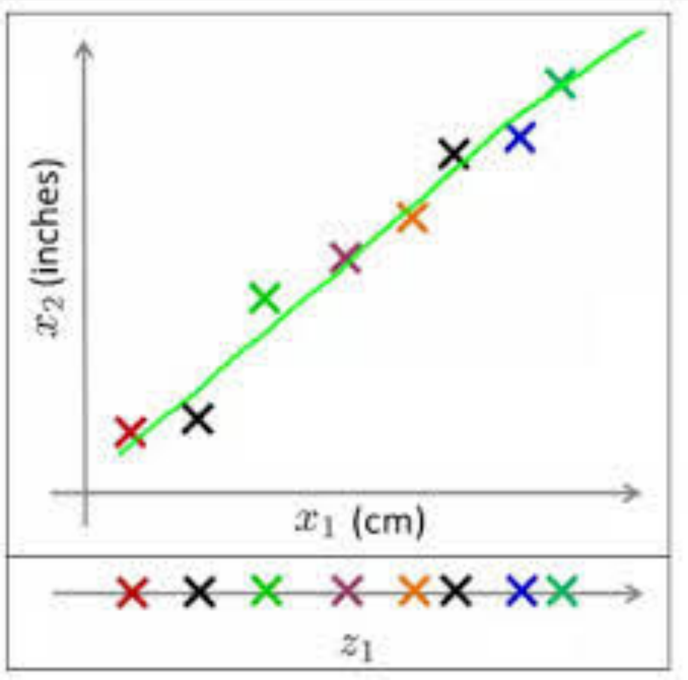
В пространстве большой размерности объекты сильно удалены друг от друга.



Источник: Bishop

Уменьшаем размерность

- Методы снижения размерности
- Отбор признаков



Источник: analyticsvidhya.com/blog/2015/07/dimension-reduction-methods

Отбираем признаки

? Задача: найти и удалить вредные признаки.

Какие признаки для нас вредные?

- Перебрать все варианты и посмотреть качество (лучший, если признаков мало)
- Посчитать корреляцию с целевой функцией и удалить шумные
- Посчитать корреляцию всех пар признаков и удалить скоррелированные
- Последовательно удалять худшие
- Последовательно добавлять лучшие

Резюме

Алгоритм:

- Наглядный, понятный
- Идеально работает, если правильно выбрана метрика
- Ленивый алгоритм, совсем не учится
- Позволяет делать **беспризнаковое распознавание**
- На признаковом распознавании, как правило, работает хуже других алгоритмов

? Какие можете придумать примеры беспризнакового распознавания?

? Какая сложность обучения алгоритма ближайшего соседа? Предсказания одного объекта?

Сложность алгоритма

Сложность обучения — $O(ND)$ (запоминаем выборку)

Сложность предсказания — $O(ND)$(считаем все расстояния)

В таком виде это в real time системах это работать не будет!

? А для k ближайших соседей?

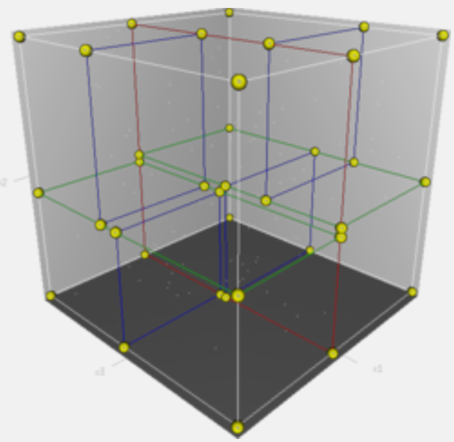
? Зачем мы тогда все это учим?

Ускоряем базовый алгоритм

Структурируем признаковое пространство, чтобы по нему быстрее искать.

- KD-tree
- Ball tree

Если признаков мало (несколько десятков), то сложность по числу объектов логарифмическая. Если много — линейная (проклятие размерности), внедрять нельзя!



Источник: en.wikipedia.org/wiki/K-d_tree

Приближенный поиск ближайших соседей

В среднем имеют логарифмическую сложность даже для больших признаковых пространств.

Примеры методов:

- ANNOY — делим пространство случайными плоскостями, строим дерево
- Navigable Small World — гуляем по графу тесного мира
- FAISS — кластеризуем пространство и ищем расстояния до центров кластеров
- LSH (Locality-sensitive hashing) — делаем хэш функцию, которая близким объектам присваивает близкие значения хэша

На семинаре разбираем ANNOY!

Применяют ли метрические алгоритмы?

Применяют!

Все большие поисковые/рекомендательные системы состоят из двух компонент:

- Грубый отбор кандидатов
- Использование финальной модели

Быстрый приближенный поиск ближайших соседей идеально подходит под задачу выборов кандидатов.

Величину $\rho(x, y)$ можно подавать в финальную модель!

Спасибо за внимание!