# Поиск ближайших соседей: некоторые проблемы и способы их решения

September 2020

## 1 Требовательность к масштабу признаков

Пусть объекты описываются двумя признаками. Евклидово расстояние между двумя объектами $x$ и $y$ считается как:

$$\rho(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}.$$

Если первый признак увеличить в k раз, то и вклад в подсчет раастояния он будет давать больше в k раз:

$$\rho(x, y) = \sqrt{k(x_1 - y_1)^2 + (x_2 - y_2)^2}.$$

**Пример.** Рассмотрим простой пример чувствительности метода ближайшего соседа к масштабу признаков. Допустим, решается задача определения пола человека по двум признакам: росту (в сантиметрах, принимает значения примерно от 150 до 200) и уровню экспрессии гена SRY (безразмерная величина от нуля до единицы; у мужчин ближе к единице, у женщин ближе к нулю). Обучающая выборка состоит из двух объектов: $x_1 = (180, 0.2)$, девочка и $x_2 = (173, 0.9)$, мальчик. Требуется классифицировать новый объект $u = (178, 0.85)$. Воспользуемся классификатором одного ближайшего соседа. Расстояния от $u$ до объектов обучения равны $\rho(u, x_1) \approx 2.1$ и $\rho(u, x_2) \approx 5$. Мы признаем новый объект девочкой, хотя это не так — высокий уровень экспрессии гена SRY позволяет с уверенностью сказать, что это мальчик. Из-за сильных различий в масштабе признаков уровень экспрессии практически не учитывается при классификации, что совершенно неправильно.

Чтобы избегать проблем с разным масштабом признаком, их можно нормировать.

- Масштабирование на отрезок $[0, 1]$:

$$\widetilde{x}^j = \frac{x^j - \min(x^j)}{\max(x^j) - \min(x^j)},$$

- Нормировка на дисперсию:

$$\widetilde{x}^j = \frac{x^j - \overline{x}^j}{\sigma(x^j)}.$$

Операции взятия среднего, дисперсии, min, max берутся по значениям признака $j$ на объектах выборки.

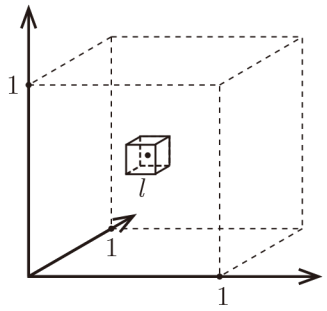
# 2 Шумовые признаки

**Вопрос** В каких еще случаях knn может находить «неправильных» соседей?

В ситуации, когда случайный шум, который есть в признаках, настолько сильно влияет на расстояние между объектами, что определение ближайшего соседа становится невозможным, knn превращается в метод случайного соседа.
Шумовые признаки могут оказать сильное влияние на метрику. Обнаружить шумовые признаки можно, удаляя поочередно все признаки и смотря на ошибку на тестовой выборке или ошибку кросс-валидации.

# 3 «Curse of dimensionality»

**Задача** Пусть есть $n$-мерный куб единичного объема, в котором равномерно распределено $N$ точек ($N$ очень большое). Внутри этого куба строится еще один, который должен покрыть долю $p$ этих точек. Какой должна быть длина ребра $l$ этого куба?




Введем плотность точек в единице объема:

$$\rho = \frac{N}{V} = N.$$

Тогда число точек внутри меньшего куба:

$$N_2 = p \cdot N = \rho \cdot V_2 = N \cdot l^n$$

Т.е. длина ребра куба:

$$e = p^{\frac{1}{n}}$$

Допустим, мы хотим покрыть 10% точек – посмотрим, как меняется $l$ с увеличением размерности:

| n | l |
|---|---|
| 10 | 0,79 |
| 50 | 0,95 |
| 100 | 0,98 |

Т.е. при больших размерностях мы должны взять почти весь куб, чтобы покрыть 10% точек.

### 3.1 «Blessing of dimensionality»

Однако с повышением размерности возникает еще один феномен, т.н. «благословение размерности». Если кратко, то с увеличением размерности, улучшается линейная разделимость классов. Связано это с тем, что в многомерном пространстве случайный вектор будет с высокой вероятностью ортогонален векторам, полученным из одного распределения (т.е. из одного класса объектов). Другими словами, становится сложнее «попасть» вектором признаков не в свой класс. Однако данный факт доказан для конкретных семейств распределений и может работать не всегда.

## 4 Эффективный поиск соседей

Искать ближайших соседей – вычислительно сложная задача (как при большой размерности признаков, так и при большом количестве объектов). Один из способов решения – искать соседей приближенно.
Подход LSH (local sensitive hashing) заключается в подборе такой хэш фукнции, которая близким объектами присваивает одинаковые значения, а удаленным – разные (с некоторой степенью вероятности). Выбор хэш функции обусловлен используемой функцией расстояния.

**Пример.** Рассмотрим меру Жаккара для множеств:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

Данная мера равна 0, когда множества состоят из разных элементов, 1, когда множества совпадают и принимает значение $[0, 1]$ в промежуточных случаях.

Пусть у нас есть некоторая хэш функция $h$, определенная на элементах множества. Определим следующую функцию:

$$h_{min}(A) = \{\min(h(x)) \mid x \in A\}.$$

Какова вероятность, что $h_{min}(A) = h_{min}(B)$? Она стремится к нулю, когда множества различны и увеличивается с увеличением числа с элементов в пересечении множеств, что очень похоже на меру Жаккара.
Однако, в случае двух конкретных множеств:

$$\begin{cases} 0,\ h_{min}(A) \neq h_{min}(B) \\ 1,\ h_{min}(A) = h_{min}(B) \end{cases},$$

промежуточных значений нет. Для того, чтобы получить более точную картину, введем $k$ разных хэш функций, где $k$ выбирается исходя из наибольшей величины ошибки $\varepsilon$:

$$k = \frac{1}{\varepsilon^2}.$$

Описанный подход называется $MinHash$ и используется, например, для эффективного поиска дубликатов среди большой коллекции текстов (с некоторой погрешностью).

**Задача.** Сконструируйте хэш функцию для поиска соседей по косинусному расстоянию (проще исходить из геометрического смысла).

Для косинусного расстояния используется следующее семейство функций:

$$F = \{f_\omega(x) = sign(\langle \omega, x \rangle)\}$$

С геометрической точки зрения, отдельный вектор $\omega$ – нормальный вектор плоскости, проходящей через начало координат. Тогда $f_\omega$ определяет, в какой полупоскости лежит рассматриваемый вектор.

Плюсы использования LSH:

- Скорость поиска
- Выигрыш в памяти (можно хранить только сигнатуры – значения хэш-функций, а не сами объекты)

## 4.1 Композиции хэш-функций

**Определение** Семейство функций $F$ называется $(d_1, d_2, p_1, p_2)$-чувствительным, если для всех $x$, $y \in X$ выполнено:

- Если $\rho(x, y) \leq d_1$, то $\mathbb{P}[f(x) = f(y)] \geq p_1$,
- Если $\rho(x, y) \geq d_2$, то $\mathbb{P}[f(x) = f(y)] \leq p_2$.

Здесь под вероятностью $\mathbb{P}$ понимается равномерное распределение на всех функциях семейства $F$.

Чтобы увеличить разницу между вероятностями $p_1$ и $p_2$, можно объединять несколько простых хэш-функций из семейства в одну сложную. Выберем для этого $m$ функций $f_1, \ldots, f_m$ из $F$ и построим новую функцию:

$$g_1(x) = (f_1(x), \ldots, f_m(x)).$$

Повторим процедуру $L$ раз и получим $L$ таких функций $g_1(x), \ldots, g_L(x)$.

Данный алгоритм имеет два параметра: число базовых функций в одной композиции $m$, и число таких композиций $L$. Увеличение параметра $m$ приводит к уменьшению вероятности того, что два непохожих объекта будут признаны схожими. Действительно, для того, чтобы значения композиции совпали на двух объектах, необходимо, чтобы совпали значения $m$ базовых хэш-функций. Если расстояние между этими объектами велико, т.е. $\rho(x, y) > d_2$, то вероятность совпадения значений $m$ базовых функций не будет превышать $p_2^m$.

Увеличение же параметра $L$ приводит к увеличению вероятности того, что два схожих объекта будут действительно признаны схожими. Действительно, объект $x$ будет рассмотрен нашим алгоритмом как кандидат в $k$ ближайших соседей для $u$, если хотя бы один из хэшей $g_1(x), \ldots, g_L(x)$ совпадет с хэшем $g_1(u), \ldots, g_L(u)$ соответственно. Если объекты $x$ и $u$ действительно схожи, то есть $\rho(x, u) \leq d_1$, то вероятность того, что они будут признаны схожими, больше или равна $1 - (1 - p_1)^L$ (в случае $m = 1$). В то же время чрезмерное увеличение параметра $L$ приведет к тому, что для нового объекта будет рассматриваться слишком много кандидатов в $k$ ближайших соседей, что приведет к снижению эффективности алгоритма.