# Лекция 22
# Метрические методы классификации

Е. А. Соколов
ФКН ВШЭ

21 марта 2017 г.

## 1 Приближенный поиск ближайших соседей (продолжение)

**Композиция хэш-функций.** Семейство хэш-функций уже можно использовать для поиска ближайших соседей. Выберем случайную хэш-функцию $f$, создадим таблицу $T$, и разместим каждый объект обучающей выборки $x$ в ячейке $f(x)$ этой хэш-таблицы [1]. Пусть теперь требуется найти $k$ ближайших соседей для объекта $u$. Вычислим для него хэш $f(u)$, возьмем все объекты из соответствующей ячейки хэш-таблицы, и вернем из них $k$ ближайших к $u$. Однако, как правило, разница между вероятностями $p_1$ и $p_2$ оказывается не очень большой, поэтому либо истинные $k$ ближайших соседей не окажутся в ячейке $f(u)$, и результат будет далек от оптимального, либо в эту ячейку попадет слишком много лишних объектов, и тогда поиск окажется слишком трудозатратным.

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 $\mathcal{F}$ и построим новую функцию $g_1(x) = (f_1(x), \ldots, f_m(x))$. Повторим процедуру $L$ раз и получим $L$ таких функций $g_1(x), \ldots, g_L(x)$. Для каждой функции $g_i(x)$ создадим свою хэш-таблицу $T_i$, и поместим каждый объект обучающей выборки $x$ в ячейку $g_i(x)$ этой таблицы. Чтобы найти $k$ ближайших соседей для нового объекта $u$, выберем объекты из ячеек $g_1(x), \ldots, g_L(x)$ таблиц $T_1, \ldots, T_L$ соответственно, и вернем $k$ наиболее близких из них.

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. $\rho(x, y) \geqslant d_2$, то вероятность совпадения значений $m$ базовых функций не будет превышать $p_2^m$. В то же время чрезмерное увеличение параметра $m$ может привести к тому, что практически все объекты попадут в разные ячейки хэш-таблицы, и для новых объектов не будет находится ни одного соседа.

[1] Поскольку множество значений хэш-функции может быть большим, обычно таблица $T$ сама является хэш-таблицей.

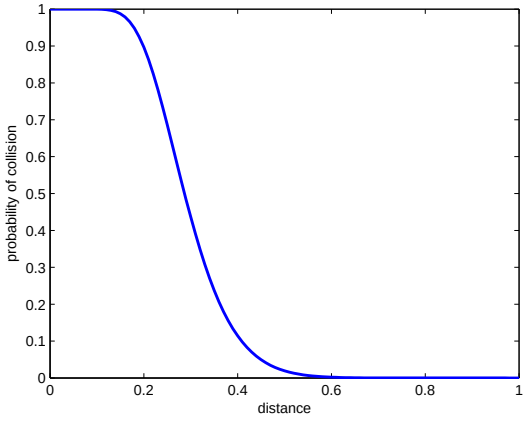


Рис. 1. Пример зависимости вероятности того, что два объекта будут признаны алгоритмом как схожие, от расстояния между этими объектами.

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 $\rho(x, u) \leqslant d_1$, 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.

Итоговый алгоритм является $(d_1, d_2, 1-(1-p_1^m)^L, 1-(1-p_2^m)^L)$-чувствительным. Вид зависимости вероятности коллизии от расстояния между объектами приведен на рис. 1 (для $m = 10$, $L = 20$). Видно, что описанный способ композиции базовых хэш-функций позволяет добиться того, что вероятность коллизии двух объектов как функция от расстояния имеет резкий скачок в определенной точке (в нашем примере 0.7). За счет выбора параметров $L$ и $m$ можно менять положение точки скачка, а также регулировать сложность алгоритма. На практике эти параметры выбирают с помощью кросс-валидации.

Отметим также, что биты хэшей $g_1(u), \ldots, g_L(u)$ можно использоваться как признаки, над которыми будет запускаться метод $k$ ближайших соседей.

**Теоретические гарантии.** Будем говорить, что алгоритм решает задачу поиска $c$-ближайшего соседа, если для нового объекта $u$ он с вероятностью $1 - \varepsilon$ возвращает объект выборки, удаленный от $u$ не более чем в $c$ раз сильнее, чем ближайший к $u$ объект выборки. Существует теоретический результат, который говорит, что можно выбрать параметры $L$ и $m$ так, что описанный алгоритм будет решать задачу поиска $c$-ближайшего соседа за $O(d\ell^r \log \ell)$, где $r$ для многих функций расстояния имеет порядок $1/c$ [1].

**Хэш-функции для косинусного расстояния.** Для косинусного расстояния используют следующее семейство функций:

$$\mathcal{F} = \{f_w(x) = \operatorname{sign}\langle w, x\rangle \mid w \in \mathbb{R}^d\}.$$

Каждая хэш-функция соответствует некоторой гиперплоскости, проходящей через начало координат, и возвращает для каждого вектора либо $+1$, либо $-1$ в зависимости от того, по какую сторону от этой гиперплоскости он находится.

**Хэш-функции для евклидовой метрики.** В данном случае хэш-функция соответствует некоторой прямой в $d$-мерном пространстве, разбитой на отрезки длины $r$. Функция проецирует объект $x$ на эту прямую и возвращает номер отрезка, в который попала проекция. Формально, семейство хэш-функций имеет вид

$$\mathcal{F} = \left\{ f_{w,b}(x) = \left\lfloor \frac{\langle w, x \rangle + b}{r} \right\rfloor \mid w \in \mathbb{R}^d, b \in [0, r) \right\}.$$

При этом, в отличие от описанных выше семейств, функции выбираются не равномерно: каждая компонента проекционного вектора $w$ выбирается из стандартного нормального распределения $\mathcal{N}(0, 1)$.

Данное семейство может быть обобщено на расстояния Минковского с $p \in (0, 2]$. В этом случае компоненты вектора $w$ должны выбираться из $p$-устойчивого распределения [2]. Например, для $p = 1$ таковым является распределение Коши.

**LSH forest.** В методе LSH присутствует два параметра: размерность хэша $m$ и число хэшей $L$. Оба следует подбирать под конкретную задачу, и от их выбора может сильно зависеть качество поиска соседей. Один из вариантов решения — алгоритм *LSH forest* [3], в котором предлагается применить каждую хэш-функцию $g_i$ к выборке и построить префиксное дерево на её выходах. Затем в качестве ближайших соседей для нового объекта объявляются те объекты из выборки, с которыми он оказался в наиболее глубоких листовых вершинах. Такой подход позволяет устранить зависимость от размера хэша и количества хэш-функций и получать хорошее качество при их фиксированных значениях.

**Рандомизированные алгоритмы.** Подходы вроде locality-sensitive hashing достаточно популярны и используются для решения многих задач. Так, *фильтр Блума* позволяет с помощью небольшого числа бит и семейства хэш-функций описать множество и проверять любой элемент на принадлежность ему. Алгоритм *HyperLogLog* может приближенно найти число различных элементов в последовательности, используя хэш-функции. Во всех этих методах используется одна и та же идея: охарактеризовать сложную структуру с помощью некоторого количества случайных признаков, и затем использовать только их для поиска нужной величины.

**Обучение хэшированию.** Рандомизированные методы простые в реализации и применении, но обладают существенным недостатком — никак не зависят от данных и от задачи, для решения которой используются. Легко представить ситуацию, в которой все объекты сосредоточены в небольшом густом облаке, и максимальный угол между двумя объектами составляет 10 градусов. В этом случае большинство хэширующих гиперплоскостей, генерируемых для косинусного расстояния, будут бесполезны. Было бы разумно выбирать их так, чтобы они попадали в облако объектов.

На решение этой проблемы направлены методы *обучения хэшированию* [4]. Их изложение выходит за рамки этого текста — отметим лишь, что они зачастую позволяют существенно уменьшить число бит в хэше при сохранении точности поиска ближайших соседей.

# 2 Обучение метрик

В методе $k$ ближайших соседей не так много параметров — число соседей, функция расстояния, ядро и его ширина. Можно выбирать метрику из числа известных — например, из евклидовой, манхэттенской и косинусной. Эти метрики фиксированы и никак не могут быть подстроены под особенности данных. Кажется, что обучение метрики под выборку могло бы увеличить число степеней свободы у метрических методов и позволить добиваться более высокого качества. Например, масштаб признаков может существенно влиять на их важность при вычислении расстояний.

Рассмотрим простой пример. 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. Евклидовы 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.

Удобнее всего обучать метрику через линейные преобразования признаков:

$$\rho(x, z) = \|Ax - Az\|^2 = (x - z)^T A^T A(x - z),$$

где $A \in \mathbb{R}^{n \times d}$ — матрица, которую можно подбирать. По сути, обучение линейного преобразования равносильно настройке параметра $\Sigma$ в метрике Махалонобиса:

$$\rho(x, z) = (x - z)^T \Sigma^{-1}(x - z),$$

если положить $A = \Sigma^{-1/2}$. Нелинейные методы часто сводятся к обучению линейных в новом признаковом пространстве (т.е. $\rho(x, z) = \|A\varphi(x) - A\varphi(z)\|^2$) либо путём ядрового перехода в линейном методе [5].

Мы разберём два подхода к обучению расстояния Махалонобиса.

## §2.1 Neighbourhood Components Analysis

Метод NCA [6] выбирает метрику так, чтобы для каждого объекта ближайшими оказывались объекты его же класса. Рассмотрим объект $x_i$ и рассмотрим следующий эксперимент: мы выбираем из оставшейся выборки случайный объект $x_j$ и относим $x_i$ к классу $y_j$. Зададим вероятности через расстояния между объектами:

$$p_{ij} = \begin{cases} \frac{\exp(-\|Ax_i - Ax_j\|^2)}{\sum_{k \neq i} \exp(-\|Ax_i - Ax_k\|^2)}, & i \neq j \\ 0, & i = j \end{cases}$$

Можно вычислить вероятность того, что объект $x_i$ будет отнесён к правильному классу. Если обозначить через $C_i = \{j \mid y_i = y_j\}$ множество индексов объектов того же класса, то данная вероятность равна

$$p_i = \sum_{j \in C_i} p_{ij}.$$

Будем максимизировать матожидание количества верно классифицированных объектов:

$$Q(A) = \sum_{i=1}^{\ell} p_i \to \max_A$$

Этот функционал можно продифференцировать по $A$:

$$\frac{\partial Q}{\partial A} = 2A \sum_i \left( p_i \sum_k p_{ik}(x_i - x_k)(x_i - x_k)^T - \sum_{j \in C_i} p_{ij}(x_i - x_j)(x_i - x_j)^T \right).$$

Далее матрицу $A$ можно обучать любым градиентным методом.

Отметим, что метод NCA можно использовать и для ускорения поиска ближайших соседей. Если взять матрицу $A \in \mathbb{R}^{n \times d}$ с небольшой первой размерностью $n$, то она будет переводить объекты в компактные представления, евклидова метрика на которых позволяет хорошо отделять классы друг от друга.

## §2.2 Large margin nearest neighbor

Метод LMNN [7] пытается обучить метрику так, чтобы $k$ ближайших соседей каждого объекта относились к нужному классу, а объекты из других классов отделялись с большим отступом. Попытаемся ввести соответствующий функционал.

Определим для каждого объекта $x_i$ набор из $k$ целевых соседей — объектов, расстояние до которых должно оказаться минимальным. В простейшем варианте это могут быть ближайшие $k$ объектов из этого же класса, но можно выбирать их и иначе. Введём индикатор $\eta_{ij} \in \{0, 1\}$, который равен единице, если объект $x_j$ является целевым соседом для $x_i$.

Выше мы поставили перед собой две цели: минимизировать расстояние до целевых соседей и максимизировать расстояние до объектов других классов. Суммарное расстояние до целевых соседей можно вычислить как

$$\sum_{i \neq j} \eta_{ij} \|Ax_i - Ax_j\|^2.$$

Для объектов других классов будем требовать, чтобы расстояние до них хотя бы на единицу превосходило расстояния до целевых соседей:

$$\sum_{i=1}^{\ell} \sum_{j \neq i} \sum_{\substack{m \neq i \\ m \neq j}} \eta_{ij} [y_m \neq y_i] \max(0, 1 + \|Ax_i - Ax_j\|^2 - \|Ax_i - Ax_m\|^2).$$

Суммируя эти два выражения, получим итоговый функционал:

$$\sum_{i \neq j} \eta_{ij} \|Ax_i - Ax_j\|^2 + \\ + C \sum_{i=1}^{\ell} \sum_{j \neq i} \sum_{\substack{m \neq i \\ m \neq j}} \eta_{ij} [y_m \neq y_i] \max(0, 1 + \|Ax_i - Ax_j\|^2 - \|Ax_i - Ax_m\|^2) \to \min_A$$

Данную задачу можно свести к стандартной задаче с линейным функционалом и ограничениями на неотрицательную определённость матрицы и решена стандартными солверами.

## Список литературы


[1] *Andoni, A., Indyk, P.* (2008). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. // Communications of the ACM, 51(1), 117.

[2] *Datar, M., Immorlica, N., Indyk, P., Mirrokni, V. S.* (2004). Locality-sensitive hashing scheme based on p-stable distributions. // Proceedings of the twentieth annual symposium on Computational geometry - SCG '04, 253.

[3] *Bawa, Mayank and Condie, Tyson and Ganesan, Prasanna* (2005). LSH Forest: Self-tuning Indexes for Similarity Search. // Proceedings of the 14th International Conference on World Wide Web.

[4] *Wang, J., Liu, W., Kumar, S., Chang, S.-F.* (2015). Learning to Hash for Indexing Big Data - A Survey. http://arxiv.org/abs/1509.05472

[5] *Kulis, B.* (2012). Metric Learning: A Survey. // Foundations and Trends in Machine Learning.

[6] *Goldberger J., Hinton G., Roweis S., Salakhutdinov R.* (2005). Neighbourhood Components Analysis. // Advances in Neural Information Processing Systems.

[7] *Weinberger, K. Q.; Blitzer J. C.; Saul L. K.* (2006). Distance Metric Learning for Large Margin Nearest Neighbor Classification. // Advances in Neural Information Processing Systems.