Семинары по метрическим методам классификации

Евгений Соколов

22 сентября 2013 г.

2 Методы поиска ближайших соседей

§2.1 Точные методы

Разберем методы поиска ближайших соседей для евклидовой метрики. Будем рассматривать задачу поиска одного ближайшего соседа, все методы несложно обобщаются на случай с $k > 1$.

Если просто перебирать все объекты обучающей выборки, выбирая наиболее близкий к новому объекту, то получаем сложность $O(\ell d)$.

Можно выбрать подмножество признаков, и сначала вычислить расстояние только по этим координатам. Оно является нижней оценкой на полноценное расстояние, и если оно уже больше, чем текущий наилучший результат, то данный объект можно больше не рассматривать в качестве кандидата в ближайшего соседа. Такой подход является чисто эвристическим и не гарантирует сублинейной сложности по размеру обучения.

kd-деревья. Одной из структур данных, позволяющих эффективно искать ближайших соседей к заданной точке, является *kd-дерево*. Оно разбивает пространство на области (каждая вершина производит разбиение по определенной координате), и каждый лист соответствует одному объекту из обучающей выборки. Обходя это дерево определенным образом, можно найти точку из обучения, ближайшую к заданной. Если размерность пространства небольшая (10-20), то данный подход позволяет находить ближайшего соседа за время порядка $O(\log \ell)$.

Экспериментально было установлено, что в пространствах большой размерности сложность поиска ближайшего соседа в kd-дереве сильно ухудшается и приобретает линейный порядок сложности [?].

§2.2 Приближенные методы

Есть два способа борьбы с высокой сложностью поиска ближайших соседей при большом числе признаков:

1. Запоминать не всю обучающую выборку, а лишь ее представительное подмножество. Существует большое число эвристических алгоритмов для отбора эталонных объектов (например, разобранный на лекции STOLP).

2. Искать k ближайших соседей приближенно, то есть есть разрешать результату поиска быть чуть дальше от нового объекта, чем k его истинных соседей. Ниже мы подробно разберем этот подход.

Опишем метод приближенного поиска ближайших соседей LSH (locality-sensitive hashing). Его идея заключается в построении такой хэш-функции для объектов выборки, которая с большой вероятностью присваивает одинаковые значения близким объектам и разные значения отдаленным объектам. Дадим формальное определение.

Опр. 2.1. *Семейство функций $\mathcal{F}$ называется (d_1, d_2, p_1, p_2)-чувствительным, если для всех $x, y \in \mathbb{X}$ выполнено:*

- *Если $\rho(x, y) \leqslant d_1$, то $\mathbb{P}_{f \in \mathcal{F}}[f(x) = f(y)] \geqslant p_1$.*
- *Если $\rho(x, y) \geqslant d_2$, то $\mathbb{P}_{f \in \mathcal{F}}[f(x) = f(y)] \leqslant p_2$.*

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

Отметим, что определение имеет смысл лишь если $d_1 \leqslant d_2$ и $p_1 \geqslant p_2$.

Пример. Рассмотрим пример семейства хэш-функций для меры Джаккарда. Пусть объекты представляют собой множества, являющиеся подмножествами универсального упорядоченного множества $U = \{u_1, \dots, u_n\}$. Выберем перестановку π на элементах этого множества, и определим хэш-функцию $f_\pi(A)$ так, чтобы она возвращала номер первого элемента в данной перестановке, входящего в A:

$$f_\pi(A) = \min\{\pi(i) \,|\, u_i \in A\}.$$

Покажем, что множество всех таких функций $\mathcal{F} = \{f_\pi \,|\, \pi \in \operatorname{Sym}(U)\}$ является (d_1, d_2, p_1, p_2)-чувствительным.

Сначала докажем следующее утверждение: вероятность того, что случайно выбранная функция $f_\pi \in \mathcal{F}$ будет принимать одинаковые значения на двух заданных множествах A и B, равна коэффициенту Джаккарда $\frac{|A \cap B|}{|A \cup B|} = 1 - \rho_J(A, B)$ этих двух множеств. Разобьем элементы u универсального множества U на три типа:

1. $u \in A$, $u \in B$.
2. $u \in A$, $u \notin B$ или $u \notin A$, $u \in B$.
3. $u \notin A$, $u \notin B$.

Обозначим число объектов первого типа через p, а число объектов второго типа — через q. Заметим, что через p и q можно выразить коэффициент Джаккарда для множеств A и B: $1 - \rho_J(A, B) = \frac{p}{p+q}$.

Вероятность того, что значения случайно выбранной хэш-функции будут одинаковыми на множествах A и B, равна вероятности того, что в случайно выбранной перестановке множества U элемент первого типа встретится раньше элемента второго типа; элементы третьего типа на значение хэш-функции никак не влияют. Последняя же вероятность равна $\frac{p}{p+q}$. Утверждение доказано.

Пусть расстояние Джаккарда между двумя множествами $\rho_J(A, B)$ не превосходит d_1. Тогда для коэффициента Джаккарда выполнено $1 - \rho_J(A, B) \geqslant 1 - d_1$, а значит, для вероятности p_1 того, что случайно выбранная функция из $\mathcal{F}$ даст одинаковые хэши для этих множеств, выполнено $p_1 \geqslant 1 - d_1$. Отсюда получаем, что $\mathcal{F}$ является $(d_1, d_2, 1 - d_1, 1 - d_2)$-чувствительным семейством.

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 $f_1, \dots, f_m$ из $\mathcal{F}$ и построим новую функцию $g_1(x) = (f_1(x), \dots, f_m(x))$. Повторим процедуру L раз и получим L таких функций $g_1(x), \dots, g_L(x)$. 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 $g_1(x), \dots, g_L(x)$ таблиц $T_1, \dots, T_L$ 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 равна p_2^k. 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.

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 $g_1(x), \dots, g_L(x)$ совпадет с хэшем $g_1(u), \dots, g_L(u)$ соответственно. Если объекты x и u действительно схожи, то есть $\rho(x, u) \leqslant d_1$, то вероятность того, что они будут признаны схожими, равна $1 - (1 - p_1)^L$ (в случае $k = 1$). 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

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

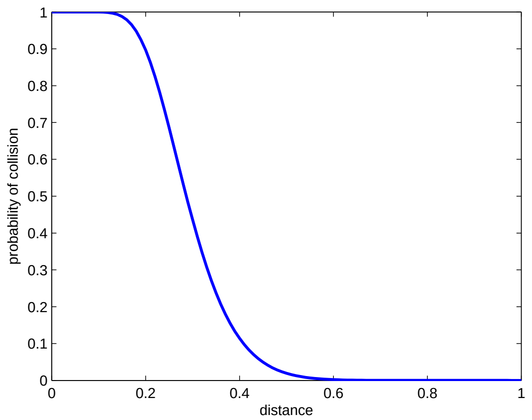

Рис. 1. Пример ависимости вероятности того, что два объекта будут признаны алгоритмом как схожие, от расстояния между этими объектами.

рис. 1 (для $m = 10$, $L = 20$, $p = 1-d$). 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 k можно менять положение точки скачка, а также регулировать сложность алгоритма. На практике эти параметры выбирают с помощью кросс-валидации.

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 k 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 $1/c$ [?].

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.

Задача 2.1. *Покажите, что данное семейство является (d_1, d_2, p_1, p_2)-чувствительным.*

Решение. В нашем случае расстояние между векторами равно углу θ между ними. Вероятность того, что точки x и y окажутся по одну сторону от случайно выбранной гиперплоскости, равна $(180 - \theta)/180$. Соответственно, семейство является $(d_1, d_2, (180 - d_1)/180, (180 - d_2)/180)$-чувствительным.

■

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 [**?**]. Например, для $p = 1$ таковым является распределение Коши.