Конспект семинара «Реализация метода ближайших соседей»

курс «Машинное обучение 1», программа OzonMasters

5 октября 2020 г.

Метод k ближайших соседей

Рассмотрим задачу многоклассовой классификации. Пусть дана обучающая выборка $X = (x_i, y_i)_{i=1}^{l}$, $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{Y} = \{1, \ldots, k\}$. Пусть также введена *функция расстояния* $\rho : \mathbb{R}^d \times \mathbb{R}^d \to [0, \infty)$. От функции не требуется соответствие всем аксиомам метрики, достаточно лишь неотрицательности. Для каждого объекта x' можно расположить объекты обучающей выборки в порядке неубывания расстояний:

$$\rho(x', x_{(1)}) \leqslant \rho(x', x_{(2)}) \leqslant \ldots \leqslant \rho(x', x_{(l)})$$

Метрический алгоритм k ближайших соседей (k nearest neighbours, kNN) относит объект x' к тому классу, представителей которого окажется больше всего среди его k ближайших соседей:

$$a(x') = \arg\max_{y \in Y} \sum_{i=1}^{k} \mathbb{I}[y_{(i)} = y]$$

Часто в качестве метрики используется стандартная евклидова метрика или косинусное расстояние ($\rho(x, x') = 1 - \cos(x, x')$).

Взвешенный метод k ближайших соседей

Рассмотренный алгоритм никак не учитывает степень близости объекта x' к его ближайшим соседям. Алгоритм k взвешенных ближайших соседей позволяет учесть расстояния до соседей с помощью введения весов объектов, например так:

$$a(x') = \arg\max_{y \in Y} \sum_{i=1}^{k} \frac{1}{\varepsilon + \rho(x', x_{(i)})} \mathbb{I}[y_{(i)} = y]$$

Быстрое вычисление евклидового рассояния

Существуют различные алгоритмы поиска ближайших соседей. Самый простой и универсальный способ — полный перебор всех объектов обучающей выборки.

Пусть $X \in \mathbb{R}^{l_1 \times d}$ — обучающая выборка, $Z \in \mathbb{R}^{l_2 \times d}$ — тестовая выборка. Подсчитаем количество операций, требующихся для подсчёта квадратов всех попарных евклидовых расстояний между объектами обучающей и тестовой выборки. Квадрат евклидова расстояния между объектами x и z записывается так:

$$\rho(x_i, z_j) = \sum_{s=1}^{d} (x_i^s - z_j^s)^2$$

Легко видеть, что расстояние между двумя объектами вычисляется за $3d$ операций: d вычитаний, d произведений (возведений в квадрат), d сложений. Тогда все попарные расстояния можно вычислить за $3dl_1l_2$ операций.

Число операций можно уменьшить с помощью простого трюка. Раскроем скобки в выражении выше:

$$\rho(x_i, z_j) = \sum_{s=1}^{d} (x_i^s)^2 + \sum_{s=1}^{d} (z_j^s)^2 - 2\sum_{s=1}^{d} x_i^s z_j^s$$

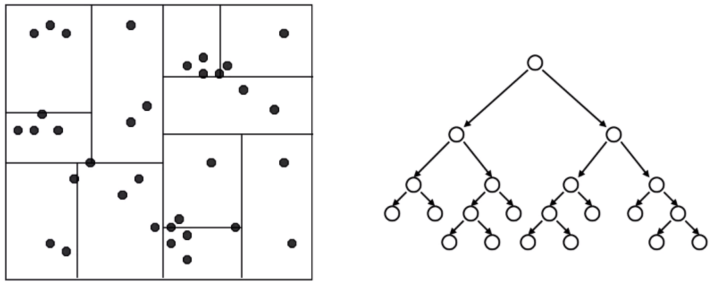
Первые две суммы достаточно вычислить для каждого объекта, а не пересчитывать отдельно по каждой паре объектов, это можно сделать за $2d(l_1 + l_2)$ операций.

Заметим, что вычисление третьей суммы по всем парам объектов можно записать как матричное произведение матриц X и Z. Вычисление этого произведения «в лоб» занимает $2dl_1l_2$ операций. При более эфективной реализации матричное умножение требует меньше операций (например, алгоритм Копперсмита — Винограда, перемножение за $O(n^{2.3727})$).

Таким образом, суммарное число операций оказывается намного меньше чем для первого способа.

Быстрый поиск ближайших соседей. KD-деревья

Для евклидовой метрики можно на этапе обучения строить различные структуры данных, позволяющие упростить сложность поиска ближайших соседей, например, KD-деревья.



С помощью KD-деревьев можно производить поиск соседа быстрее ($O(\log l)$ для одного объекта, где l – количество объектов), но возникают дополнительные затраты для его построения (возможно построение за $O(l \log l)$). Также дополнительные затраты возникают при добавлении новых объектов в обучающую выборку.

Эффективный подбор параметров в методе kNN

Одним из основных параметров метода kNN является число ближайших соседей, которое в процессе обучения приходится подбирать по отложенной выборке или кросс-валидации. В общем случае, чтобы замерить качество алгоритма при m значениях параметра на отложенной выборке, необходимо m раз обучить и применить алгоритм. В случае k ближайших соседей это будет стоить $O(mdl^2)$ операций.

Отсортируем значения параметра, которые мы хотим проверить, по неубыванию:

$$k_1 \leqslant k_2 \leqslant \ldots \leqslant k_m$$

Найдём k_m ближайших соседей в обучающей выборке для объектов отложенной выборки.

Заметим, что подсчёт ближайших соседей — самая вычислительно затратная часть алгоритма. Очевидно, что для всех остальных значений параметра не нужно проводить поиск ближайших соседей заново, достаточно выбрать из уже найденных соседей нужное количество.

Таким образом, стоимость алгоритма подбора параметров была сокращена до $O(dl^2)$ операций. Предложенная схема легко обобщается на случай оценивания алгоритма по кросс-валидации.

Преобразования объектов для улучшения качества

Рассмотрим два простых способа улучшить качество алгоритма kNN с помощью некоторых элементарных преобразований изображений. Пусть у нас есть m элементарных преобразований $\phi_j(x) : \mathbb{R}^d \to \mathbb{R}^d$, $j \in \{1, \ldots, m\}$, введём $\phi_0(x) = x$.

Создадим новую обучающую выборку:

$$X_{new} = \left(\bigcup_{j=0}^{m} (\phi_j(x_i), y_i)_{i=1}^{l} \right)$$

Качество алгоритма, обученного выборке X_{new}, при удачно выбранных преобразованиях будет превышать исходное.

Другой способ основан на преобразовании объектов тестовой выборки, а не обучающей. Для каждого объекта тестовой выборки x' получим множество объектов $\Phi(x') = \{\phi_j(x') | j \in \{1, \ldots, m\}\}$. Введём «расстояние» от множества $\Phi(x')$ до объекта обучающей выборки x:

$$\rho(x, \Phi(x')) = \min_j \{\rho(x, \phi_j(x') | j \in \{1, \ldots, m\}\}$$

Вычисление k ближайших соседей для $\Phi(x')$ можно организовать следующим образом:

1. Для каждого объекта из $\Phi(x')$ найдём его ближайших соседей из обучающей выборки X

2. Среди всех найденных соседей выберем k объектов с наименьшим расстоянием

По полученному множеству ближайших соседей вычисляется ответ алгоритма. Этот способ более эффективен по памяти чем первый.

Список литературы

[1] Википедия. KD-деревья. https://en.wikipedia.org/wiki/K-d_tree

[2] Конспект ИТМО. «K-d деревья и перечисление точек в произвольном прямоугольнике»

[3] ray-tracing.ru. kd-tree. http://www.ray-tracing.ru/articles181.html