Метрические методы - продолжение

Виктор Владимирович Китов

МГУ им.Ломоносова, ф-т ВМиК, кафедра ММП.

I семестр 2015 г.

Метод парзеновского окна

- Метод парзеновского окна:

$$\widehat{f}(x) = \arg\max_{y \in Y} \sum_{n=1}^{N} \mathbb{I}[y_n = y] K\left(\frac{\rho(x, x_n)}{h(x)}\right)$$

- Выбор $h(x)$:
  - $h(x) = const$
  - $h(x) = \rho(x, z_K)$ где z_K - K-й ближайший сосед.
    - лучше для неравномерно распределенных объектов
    - для $K(z) = \mathbb{I}[z \leq 1]$ и $h(x) = \rho(x, z_K)$ приводит к методу K-ближайших соседей.

Сложность алгоритма ближайших соседей

- Сложность обучения $O(1)$
  - нужно просто запомнить новый объект
- Сложность предсказания $O(N)$
  - нужно посчитать расстояние от x до всех объектов обучающей выборки
- Варианты упрощения поиска:
  - уменьшить число объектов обучающей выборки
    - удалить шумы (editing)
    - удалить неинформативные объекты (condensing)
  - структурировать признаковое пространство и просматривать только потенциально полезные его части
    - LAESA, KD-tree, ball-tree

Содержание

1. Уменьшение числа объектов обучающей выборки
2. Структурирование признакового пространства

Понятие отступа

- Рассмотрим обучающую выборку: $(x_1, c_1), (x_2, c_2), ...(x_N, c_N)$, где c_i - правильный класс для объекта x_i, а $\mathbf{C} = \{1, 2, ...C\}$ - множество допустимых классов.
- Определим понятие отступа:

$$M(x_i, c_i) = g_{c_i}(x_i) - \max_{c \in \mathbf{C} \setminus \{\mathbf{c_i}\}} g_c(x_i)$$

  - отступ отрицательный <=> объект x_i был неправильно классифицирован
  - величина отступа показывает, насколько классификатор уверен, что объект x_i может быть отнесен к истинному классу c_i

Удаление шумов

Основная идея

Шумы - это объекты, сильно выбивающиеся из закономерности, определяемой методом ближайших соседей. То есть их можно определить как

$$\{x_i : M(x_i, c_i) < -\delta\}$$

для достаточно большого $\delta > 0$.

Алгоритм 1: Фильтрация шумовых наблюдений

для каждого x_i, c_i в обучающей выборке TS:
    рассчитать $M(x_i, c_i)$

вернуть отфильтрованную обучающую выборку:
    $T' = \{(x_i, c_i) : M(x_i, c_i) \geq -\delta\}$

Может потребоваться несколько итераций алгоритма.

Удаление неинформативных наблюдений

Основная идея

Неинформативные наблюдения - это такие наблюдения, которые не добавляют информации о классе при их учете.

- Обычно неинформативных наблюдений - большинство для выборок большого объема с простой разделяющей кривой.

Алгоритм 2: Фильтрация неинформативных наблюдений

для каждого класса $c = 1, 2, ...C$: # добавить самый репрезента-
    $x(c) = \arg\max_{x_i : c_i = c} \{M(x_i, c_i)\}$ # тивный представитель
Инициализировать множество эталонов: $\Omega = \{x(c), c = 1, 2, ...C\}$

повторять, **пока** точность существенно улучшается:
    $x_i = \arg\min_{x_i \in T\Omega} M(x, \Omega)$ # присоединили объект
    $\Omega = \Omega \cup x_i$ # с наименьшим отступом

вернуть Ω

Содержание

1 Уменьшение числа объектов обучающей выборки

2 Структурирование признакового пространства

Структурирование признакового пространства

- LAESA:
  - отсев заведомо слишком далеких объектов, чем найденный
- Иерархическое упорядочивание объектов через систему простых вложенных фигур
  - KD-trees - вложенные прямоугольники
  - ball-trees - вложенные шары
- Комментарии:
  - вложенность понимается в том смысле, что дочерние фигуры совместно содержат все объекты родительской фигуры
  - метод ближайших соседей перестает быть онлайновым, т.к. структура признакового пространства оптимизируется под TS.
  - мера близости должна удовлетворять неравенству треугольника

LAESA - инициализация

- Выделяем систему P опорных объектов $z_1, z_2, \ldots z_P$ из объектов TS:
  - z_1 выбирается случайным образом
  - для $p = 2, 3, \ldots P$: $z_p = \arg\max_{x \in TS} \sum_{i=1}^{p-1} \rho(z_i, x)$
- Основное используемое свойство:
  1. по неравенству треугольника:
     $\rho(x, x') \geq |\rho(x', z_p) - \rho(z_p, x)|$
  2. поскольку $p = 1, 2, \ldots P$ - любое, то справедлива оценка снизу на расстояние:

$$\rho(x, x') \geq G(x') = \max_{p=1,2,\ldots P} |\rho(x', z_p) - \rho(z_p, x)| \qquad (1)$$

  3. Поскольку $x' \in \{x_1, x_2, \ldots x_N\}$, то все $\rho(x', z_k)$ в (1) можно посчитать заранее.

LAESA - алгоритм

- Обозначения:
  - x - объект, для которого требуется найти ближайшего соседа.
  - Ω - множество объектов-потенциальных кандидатов для ближайшего соседа
- Алгоритм:
  1. $\Omega = TS$
  2. находим расстояния $d_i = \rho(x, z_i)$
  3. текущий ближайший сосед $x_{n.n.} = \arg\min_{z_i} \rho(x, z_i)$, $d_{n.n.} = \rho(x, x_{n.n.})$
  4. для каждого $x' \in \Omega$ оцениваем расстояние снизу: $\rho(x, x') \geq G(x') = \max_{p=1,2,\ldots P} |\rho(x', z_k) - \rho(z_k, x)|$
  5. отсеиваем заведомо неподходящие объекты: $\Omega = \Omega \setminus \{x' \in \Omega : G(x') \geq d_{n.n.}\}$
  6. находим следующего кандидата $\tilde{x} = \arg\min_{x' \in \Omega} G(x')$
  7. если $\rho(x, \tilde{x}) < \rho(x, x_{n.n.})$,
     - то $x_{n.n.} = \tilde{x}$, $d_{n.n.} = \rho(x, \tilde{x})$, переход на шаг 5.
     - иначе $\Omega = \Omega \setminus \{\tilde{x}\}$, переход на шаг 6.

Комментарии по LAESA

- Требования по памяти: PN
- Оптимальное P - баланс между сложностью расчета $G(x')$ и сложностью расчета $\rho(x, x')$.
- Для метода K ближайших соседей - поддерживать список не 1, а K ближайших кандидатов в алгоритме.

KD-tree

- Построение субоптимального дерева:

```
построить_узел_дерева(condition, Ω):
    если |Ω| < n_min:
        вернуть лист с приписанными объектами Ω
    иначе:
        найти признак с наибольшим разбросом на Ω:
            i = arg max_i σ({x_n^i : x_n ∈ Ω})
        найти медиану на Ω: μ = median({x_n^i : x_n ∈ Ω})
        вернуть внутреннюю вершину с двумя потомками:
            левый потомок=
                построить_узел_дерева(x^i < μ, {x_i ∈ Ω : x_i^i < μ})
            правый потомок=
                построить_узел_дерева(x^i ≥ μ, {x_i ∈ Ω : x_i^i ≥ μ})
```

KD-tree

- сложность $O(DN \log_2 N)$, поскольку на уровне h с k внутренними вершинами:
  - $|\Omega_1| = n_1, \ldots |\Omega_k| = n_k$, $n_1 + n_2 + \ldots + n_k \leq N$
  - требуется посчитать σ для D признаков - сложность $O(D(n_1 + \ldots + n_k)) \leq O(DN)$
- требуется повторить процедуру для каждого уровня $h = 1, 2, \ldots H$, $H \leq \lceil \log_2 N \rceil$, поскольку каждый раз число наблюдений в листе делится пополам и дерево сбалансированное
- геометрически эффективнее брать не медиану, а среднее, но дерево может оказаться несбалансированным.
- глубина $O(\log N)$ для сбалансированного случая, а для несбалансированного в наихудшем случае глубина $O(N)$

KD-tree: расстояние до прямоугольника

- Каждой вершине дерева соответствует гиперпрямоугольник. Расстояние до вершины = расстоянию до гиперпрямоугольника.
- Расстояние от $x = [x^1, ...x^D]^T$ до прямоугольника $\{(h_1, ...h_D) : h_d^{min} \leq h_d \leq h_d^{max}\}$ считается как $\rho(x, z)$, где z - ближайшая на прямоугольнике точка со следующими координатами:

$$z^d = \begin{cases} h_d^{min} & x^d < h_d^{min} \\ x^d & h_d^{min} \leq x^d \leq h_d^{max} \\ h_d^{max} & x^d > h_d^{max} \end{cases}$$

KD-tree: поиск ближайшего соседа

- Для интересующего объекта x находим лист дерева, которому он принадлежит:
  - для текущей вершины смотрим дискриминантный признак x^i и порог вершины μ
  - если $x^i < \mu$, переходим к левому потомку, иначе - к правому, пока не дойдем до листа.
- Полным перебором среди всех объектов в листе определяем ближайшего соседа

KD-tree: поиск ближайшего соседа

- Обходим все дерево, переходя к вершине-предку и считая расстояния до всех потомков данной вершины-предка
  - если расстояние до вершины больше расстояния до текущего ближайшего соседа - пропускаем соответствующую вершину
  - если нет, то делаем проверки для вложенных вершин-потомков
  - если нашли более близкий объект, то обновляем текущего ближайшего соседа и расстояние до него.

Ball trees

- Вложенная система шаров, кажой вершине дерева приписан шар.
- Вложенность: все объекты из TS, попавшие в родительский шар, попадут в один и только один из дочерних шаров.
- Характеристики шара:
  - центр c
  - объекты, приписанные к шару $z_1, z_2, ...z_K$
  - радиус $R = \max_i \|z_i - c\|$

Ball tree: рекурсивная генерация

- для родительского шара $Ball(c, \Omega)$ (с центром c и приписанными объектами Ω):
  - выбрать $c_1 = \arg\max_{z_i \in \Omega} \|z_i - c\|$
  - выбрать $c_2 = \arg\max_{z_i \in \Omega} \|z_i - c_1\|$
  - разбить объекты Ω на 2 группы:

$$\Omega_1 = \{z_i : \|z_i - c_1\| < \|z_i - c_2\|\}$$

$$\Omega_2 = \{z_i : \|z_i - c_2\| \leq \|z_i - c_1\|\}$$

  - сопоставить родительскому шару 2 дочерних: $Ball(c_1, \Omega_1)$ и $Ball(c_2, \Omega_2)$.

Ball tree: расстояние до шара

По неравенству треугольника для любого $z \in B = Ball(c, R)$:

$$\rho(x, c) \leq \rho(x, z) + \rho(z, c)$$

откуда следует, что

$$\rho(x, z) \geq \rho(x, c) - \rho(z, c) \geq \rho(x, c) - R$$

Эта нижняя граница достигается для $z \in B$, лежащего между x и c на расстоянии R от центра.

Комментарии

- Для применения LAESA, KD-tree и ball-tree мера близости $\rho(x, z)$ должна удовлетворять неравенству треугольника.
- Чем выше D, тем менее эффективно структурирование пространства.
  - оно рассчитано разделять объекты на геометрически компактные группы
  - при больших D почти все объекты становятся одинаково далекими друг от друга
  - в KD-tree: близость по отдельным координатам не гарантирует близости по остальным.
- Для больших D ball-tree работает эффективнее, чем KD-tree, за счет того, что шары более точно оценивают нижнюю границу для расстояния до точек, чем прямоугольники.

Подробности методов

Более подробное описание отдельных методов см. в

- Statistical Pattern Recognition. 3rd Edition, Andrew R. Webb, Keith D. Copsey, John Wiley&Sons Ltd., 2011, раздел 4.2.
- Воронцов К.В. Курс лекций. Метрические методы классификации и регрессии.