Метод ближайших соседей; SVM

Зинина Анастасия

- Дана обучающая выборка $X = (x_i, y_i)_{i=1}^{l}$ и функция расстояния: $\rho : X \times X \to [0, \infty)$, и требуется классифицировать новый объект u.
- **Алгоритм ближайшего соседа** (NN) относит классифицируемый объект $u \in X^l$ к тому классу, которому принадлежит ближайший обучающий объект: $w(i, u) = [i = 1]; \quad a(u; X^l) = y_u^{(1)}$.

- **Алгоритм k ближайших соседей (kNN)**. Чтобы получить более точный результат и уменьшить влияние выбросов, будем относить объект u к тому классу, элементов которого окажется больше среди k ближайших соседей $x_u^{(i)}, i = 1, ..., k$:

$$w(i, u) = [i \leqslant k];$$

$$a(u; X^l, k) = \underset{y \in Y}{argmax} \sum_{i=1}^{l} [y_u^{(i)} = y]$$

- **Алгоритм k взвешенных ближайших соседей** Однако некоторые из k рассматриваемых объектов могут находиться слишком далеко, они должны оказывать меньшее влияние на решение алгоритма. Решение - ввести строго убывающую последовательность весов.

Веса

- **Веса как функция от номера**
  - линейная функция $w(i) = \frac{k+1-i}{k}$ - неоднозначность может встречаться
  - нелинейная функция, например, q^i, $0 < q < 1$
- **Веса как функции от расстояния d:**
  - $\frac{1}{(d+a)^b}$
  - q^d, $0 < q < 1$

Способы задания весов

- Введём функцию ядра K(z), невозрастающую на $[0, \infty)$.

- Положив $w(i,u) = K(\frac{\rho(u,x_u^{(i)})}{h})$, получим алгоритм

$$a(u, X^l, h) = \underset{y \in Y}{argmax} \sum_{i=1}^{l} [y_u^{(i)} = y] K(\frac{\rho(u, x_u^{(i)})}{h}),$$

  где h - ширина окна, определяющая окрестность, в которой находятся объекты обучающей выборки, учитывающиеся в классификации объекта u

- Окно может быть фиксированной или переменной ширины. Окно переменной ширины нужно при неравномерном распределении точек в пространстве

- Возьмём финитное ядро — невозрастающую функцию K(z), положительную на отрезке [0,1], и равную нулю вне него. Определим h как наибольшее число, при котором ровно k ближайших соседей объекта u получают ненулевые веса: $h(u) = \rho(u, x_u^{k+1})$. Тогда алгоритм принимает вид

$$a(u, X^l, h) = \underset{y \in Y}{argmax} \sum_{i=1}^{l} [y_u^{(i)} = y] K(\frac{\rho(u, x_u^{(i)})}{\rho(u, x_u^{(k+1)})})$$

Задача регрессии

Решение задачи регрессии методом kNN - усреднение значений y_i:

$$a(u; X^l; k) = \frac{\sum_{i=1}^{k} w_{(i)} y_i}{\sum_{i=1}^{k} w_{(i)}}$$

Стандартизация данных

- Представим, что есть два признака, масштаб которых отличается в 100 раз

Объект/признак	f_1	f_2	Класс
x_1	100	1	1
x_2	150	2	2
x_3	130	1.2	?

- При этом второй признак гораздо важнее
- Расстояние от 3 до 1 объекта меньше расстояния от 2 до 3, поэтому 3 объект будет отнесён к классу 2. Хотя мы считаем, что класс почти полностью определяется признаком 2. Т.е. объект должен быть отнесён к классу 1

$$\rho(x_1, x_3) = \sqrt{900 + 0.04} \approx 30$$

$$\rho(x_2, x_3) = \sqrt{400 + 0.64} \approx 20$$

- Поэтому рекомендуется стандартизировать данные методами, о которых говорилось ранее

Проклятие размерности

- Рассмотрим вектор $(x_1, ..., x_n)$ и немного отличающийся от него $(x_1 + \varepsilon, ..., x_n + \varepsilon)$. Квадрат расстояния равен $n\varepsilon^2$. При большом значении n расстояние может быть больше расстояния до объекта, существенно отличающегося в одной координате.

- В n-мерном пространстве можно найти $n + 1$ равноудаленную точку, т.е. между любыми двумя расстояния могут быть почти одинаковы.

- Рассмотрим данные, равномерно распределенные в единичном гиперкубе размерности p. Предположим, что мы хотим охватить долю r всех входных данных. Для этого длина ребра должна быть равна $r^{1/p}$. При $p = 10$ чтобы покрыть 10% данных, нужно покрыть 80% возможных значений каждого признака. Это уже сложно назвать локальным методом. И уменьшение r не поможет, так как в этом случае алгоритм будет больше подстраиваться под обучающую выборку.

Проклятие размерности

- Пусть N точек равномерно распределены в единичном шаре размерности p. Тогда среднее расстояние от нуля до точки равно

$$(1 - \frac{1}{2}^{1/N})^{1/p}$$

  т.е., например, в размерности 10 для N = 500 $d \approx 0.52$, т.е. больше половины. Большинство точек в результаты ближе к границе носителя, чем к другим точкам, поэтому может оказаться бессмысленным судить о сходстве объектов по расстоянию между ними.

- Сложность точных методов решения задачи поиска ближайших соседей становится линейной по размеру выборки по мере роста размерности.

Методы нахождения ближайших соседей

- Полный перебор -сложность нахождения одного соседа $O(ln)$,где l - размер выборки, n- размерность признакового пространства;
- Специальные методы - kd-tree, ball-tree - сложность нахождения одного соседа $O(\log l)$;
- Однако с ростом размерности сложность становится порядка $O(l)$
- Рассмотрим методы, применимые в пространствах большой размерности
  - Рассматривать только "типичных"представителей классов, а не всю выборку
  - Искать k соседей приближенно

kd-tree

- В каждом узле точки разделяются по одной из координат, пока не будет достигнуто условие остановки.
- Важно выбрать, по какой координате разделять точки и каково граничное значение координаты для разделения.
- Выбор координаты, которая больше всего варьируется, ведёт к построению меньших деревьев
- Обычно выбирают медианное значение, тогда разделение точек гиперпараллелепипедами становится более сбалансированным, а сами гиперпараллелепипеды становятся меньше в областях с большим количеством точек.

*k*d-Tree

kd-tree

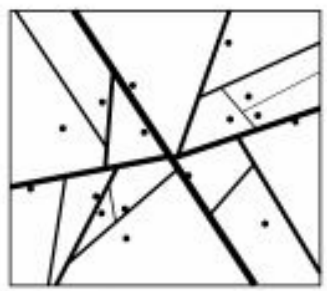
- Алгоритм эффективен в пространствах небольшой размерности.
- Недостатки:
  - Число соседей для каждого листа растёт экспоненциально с ростом размерности, поиск приобретает линейную сложность.
  - Разделение в узлах всегда параллельно одной из осей, реальное распределение точек не учитывается. Это может привести к не очень эффективной работе.
- Первую из вышеперечисленных проблем решают использованием приближенного поиска. Один из вариантов -Best Bin First. он основан на наблюдении, что большинствососедних ячеек не содержат ближайших соседей. Следовательно, ищем ячейку-кандидата в порядке увеличения расстояния и используем раннюю остановку. Метод находит 95% соседей корректно.

ball tree

- kd-tree можно назвать 'проективными деревьями", рассмотрим также "метрические деревья которые не требуют конечномерности пространства

- Рассмотрим ball tree. Кажая вершина представляет гиперсферу.Каждый внутренний узел делит точки на два непересекающихся множества, ассоциированных с разными сферами. Сами сферы могут пересекаться.

- Дочерние вершины выбираются так, чтобы расстояние между ними было максимальным.



Ball Tree

ball tree

- Это делается следующим образом:
- Сначала находится центр масс, в качестве центра первой дочерней вершины выбирается наиболее удалённая от него точка
- В качестве центра второй дочерней вершины выбирается наиболее удалённая от первой
- Остальные точки делятся на два множества в зависимости от расстояния до выбранных центров
- Получившееся разделение можно рассматривать как нахождение гиперплоскости, делящей пополам отрезок, соединяющий два центра и перпендикулярный ему
- никаких ограничений на число точек внутри сфер нет, поэтому дерево может быть очень несбалансированным. Однако такие деревья могут давать высокую скорость поиска, если они уловили истинное распределение точек пространства.

LSH

- Построим хэш-функцию, которая с большей вероятностью присваивает одинаковые значения близким объектам и разные значения отдалённым объектам.
- **Опр.** Семейство функция F называется (d_1, d_2, p_1, p_2)-чувствительным, если для всех $x, y \in X$ выполнено:
  1) Если $\rho(x, y) \leqslant d_1$, то $P_{f \in F}[f(x) = f(y)] \geqslant p_1$
  2) $\rho(x, y) \geqslant d_2$, то $P_{f \in F}[f(x) = f(y)] \leqslant p_2$,
  где $P_{f \in F}$-равномерное распределение на функциях из F
- Простой подход:выберем случайным образом f из F и поместим x в ячейке f(x) хеш-таблицы. Для поиска k ближайших соседей объекта a строим f(a) и берём k ближайших из соответствующей ячейки хэш-таблицы.Часто разница между p_1 и p_2 оказывается не очень большой, поэтому либо истинные k ближайших соседей не окажутся в ячейке $f(u)$, либо в эту ячейку попадет слишком много лишних объектов.
- Идея: объединить базовые хэш-функции в одну сложную.

LSH

- Выберем m функций $f_1, ..., f_m$ из F и построим новую функцию $g_1(x) = (f1(x), ..., fm(x))$.
- Повторим процедуру L раз и получим L таких функций $g_1(x), ..., g_L(x)$.
- 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 a, выберем объекты из ячеек $g_1(x), ..., gL(x)$ таблиц $T_1, ..., T_L$ и вернем k наиболее близких из них.

- Если $\rho(x, y) \geqslant d_2$, то $P[f(x) = f(y)] \leqslant 1 - (1 - p_1^m)^L$
  Если $\rho(x, y) \leqslant d_1$, то $P_{f \in F}[f(x) = f(y)] \geqslant 1 - (1 - p_1^m)^L$

  T.e. чем больше L и m, тем точнее алгоритм. Однако если L слишком велико,то поиск будет занимать больше времени. Если велико m, то большинство объектов попадут в разные ячейки хэш-таблицы.

Метрики и меры близости

Метрика Минковского

$$\rho(x, y) = (\sum_{i=1}^{n} \mid x_i - y_i \mid^p)^{1/p}$$

Частные случаи:

- "Считающее"расстояние при $p = 0$ - число координат, по которым вектора различаются:

$$\rho_0(x, y) = \sum_{i=1}^{n} I_{x_i \neq y_i}$$

- Манхэттенское расстояние при $p = 1$;
- Евклидова метрика при $p = 2$
- Метрика Чебышёва при $p = \infty$:

$$\rho_\infty(x, y) = \max_{i=1,\ldots,n} \mid x_i - y_i \mid$$

Метрики и меры близости

- Можно использовать взвешенное расстояние, когда одни признаки важнее других

$$\rho(x, y) = (\sum_{i=1}^{n} w_i \mid x_i - y_i \mid^p)^{1/p}$$

  - Веса могут выполнять роль нормировки признаков
  - Веса настраиваются по обучающей выборке с помощью методов оптимизации

- Косинусная мера

$$\rho(x, y) = arccos(\frac{< x, y >}{||x||||y||})$$

  - Часто используется для определения схожести текстов. Составляется словарь, текст представлен в виде вектора, координаты которого равны

$$x_i = \begin{cases} 0, & \text{i-ое слово из словаря не встречается в тексте} \\ 1, & \text{i-ое слово из словаря встречается в тексте} \end{cases}$$

  - Благодаря нормировке эта мера не зависит от длин текстов.

- Расстояние Джаккарда
  Если пользоваться выражениями множеств в виде бинарных векторов, как это было в примере с косинусной мерой, можно записать расстояние Джаккарда как

$$\rho(x, y) = 1 - \frac{< x, y >}{|| x ||^2 + || y ||^2 - < x, y >}$$

Меры близости для категориальных признаков

Для категориальных признаков не имеет смысла вводить расстояние аналогично случаю с количественными признаками. Рассмотрим, какие же меры близости можно ввести.
Введём обозначения

- $f_k(x)$ - число раз, которое k-ый признак принимает значение x
- Частота, с которой k-ый признак принимает значение x

$$p_k(x) = \frac{f_k(x)}{l}$$

- Оценка вероятности того, что у случайно выбранных объктов k-ый признак принимает значение x

$$p_k(x) = \frac{f_k(x)(f_k(x)-1)}{l(l-1)}$$

Меры близости для категориальных признаков

- Overlap

$$\rho_k(x_k, y_k) = \begin{cases} 1, & \text{если } x_k = y_k \\ 0, & \text{иначе} \end{cases}$$

- Вариант Goodall. Мера, которая приписывает больше схожести, если оба объекта принимают редкое значение. Если они принимают распространённое на выборке значение, то их схожесть оценивается меньшим значением

$$\rho_k(x_k, y_k) = \begin{cases} 1 - \sum\limits_{q:p_k(q)\leqslant p_k(x_k)} p_k^2(q), & \text{если } x_k = y_k \\ 0, & \text{иначе} \end{cases}$$

- Вариант Lin. Мера, которая приписывает меньший вес несовпадению, если значения x_k и y_k встречаются часто. При совпадении больший вес - редким значениям

$$\rho_k(x_k, y_k) = \begin{cases} \sum\limits_{q:p_k(x_k)\leqslant p_k(q)\leqslant p_k(y_k)} \log p_k(q), & \text{если } x_k = y_k \\ 2\log \sum\limits_{q:p_k(q)_k(x_k)} p_k(q), & \text{иначе} \end{cases}$$

Метод опорных векторов

- Рассиотрим линейный классификатор $a(x) = sign(< w, x > -w_0)$.
  - Предположим, что выборка линейно разделима.
  - Разделяющая гиперплоскость может быть не единственной.
  - Потребуем, чтобы разделяющая гиперплоскость максимально далеко отстояла от ближайших к ней точек обоих классов.

- Выберем константу в алгоритме так, чтобы $< w, x_i > -w_0 = y_i$ для ближайших к разделяющей гиперплоскости объектов.

- Ширина полосы $(x^+ - x^-, \frac{w}{||w||}) = \frac{2}{||w||}$ максимальна, когда норма вектора w минимальна.

- Получили задачу квадратичного прогрммирования:
$$\begin{cases} \frac{1}{2} < w, w > \to min, \\ y_i(< w, x_i > -w_0) \geqslant 1 \end{cases}$$

Линейно разделимая выборка

- Решение задачи
$$\begin{cases} \frac{1}{2} < w, w > \to min, \\ y_i(< w, x_i > -w_0) \geqslant 1 \end{cases}$$

- эквивалентно поиску седловой точки лагранжиана
$$\begin{cases} L(\mathbf{w}, b, \lambda) = 0.5(\mathbf{w}, \mathbf{w}) - \sum\limits_{i=1}^{N} \lambda_i(y_i((\mathbf{w}, \mathbf{x}_i) - w_0) - 1) \to \ \min_{\mathbf{w}, w_0} \max_{\lambda} \\ \lambda_i \geqslant 0, \ \ i = 1, ..., l \\ \lambda_i = 0, \ \text{либо} < w, x_i > -w_0 = y_i, \ \ i = 1, ..., l \end{cases}$$

Линейно разделимая выборка

- Необходимое условие седловой точки:
$$\begin{cases} 0 = \frac{\partial L}{\partial w_j} = w_j - \sum\limits_{i=1}^{N} \lambda_i y_i x_{ij}, \quad j = 1, ..., n, \ \Rightarrow \mathbf{w} = \sum\limits_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \\ 0 = \frac{\partial L}{\partial b} = \sum\limits_{i=1}^{N} \lambda_i y_i \ \Rightarrow \sum\limits_{i=1}^{N} \lambda_i y_i = 0. \end{cases}$$

- **Опр.** Если $\lambda_i > 0$ и $< w, x_i > - w_0 = y_i$, то объект обучающей выборки x_i называется опорным вектором (support vector).

- Перепишем задачу:
$$\begin{cases} -L(\lambda) = -\sum\limits_{i=1}^{l} \lambda_i + 0.5 \sum\limits_{i=1}^{l} \sum\limits_{j=1}^{l} \lambda_i \lambda_j (< x_i, x_j >) \to \min\limits_{\lambda} \\ \lambda_i \geqslant 0, \ i = 1, ..., l \\ \sum\limits_{i=1}^{l} \lambda_i y_i = 0 \end{cases}$$

Линейно неразделимая выборка

- Штрафуем алгоритм за ошибки (точки внутри разделяющей полосы):

$$\begin{cases} 0.5 < w, w > +C\sum_{i=1}^{l}\xi_i \ \rightarrow \min\limits_{w,w_0,\xi} \\ y_i(< w, x_i > -w_0) \geqslant 1 - \xi_i \\ \xi_i \geqslant 0, \ i = 1, ..., l \end{cases}$$

- $\xi \geqslant \max\{0, 1 - M_i\}$ И минимум будет достигаться при $\xi = \max\{0, 1 - M_i\}$

- Что эквивалентно задаче линейной классификации с кусочно-линейной функцией потерь и квадратичной регуляризацией:

$$Q(a, X^l) = \sum_{i=1}^{l}(1 - M_i)_+ + \frac{1}{2C} \mid\mid w \mid\mid^2 \rightarrow min$$

Линейно неразделимая выборка

- Запишем лагранжиан

$$L(\mathbf{w}, w_0, \lambda) = 0.5(\mathbf{w}, \mathbf{w}) - \sum_{i=1}^{l} \lambda_i (M_i(w, w_0) - 1 + \xi_i) - \sum_{i=1}^{l} \xi_i \eta_i + C \sum_{i=1}^{l} \xi_i$$

$$= 0.5(\mathbf{w}, \mathbf{w}) - \sum_{i=1}^{l} \lambda_i (M_i(w, w_0) - 1) - \sum_{i=1}^{l} \xi_i (\eta_i + \lambda_i - C)$$

- Двойственная задача

$$\begin{cases} L(\mathbf{w}, w_0, \lambda) \to \min_{\mathbf{w}, w_0} \max_{\lambda} \\ \lambda_i \geqslant 0, \ \xi_i \geqslant 0, \ \eta_i \geqslant 0, \ i = 1, ..., l \\ \lambda_i = 0, \ \text{либо} < w, x_i > - w_0 = y_i, \ i = 1, ..., l \\ \eta_i = 0, \ \text{либо} \eta_i = 0, \ i = 1, ..., l \end{cases}$$

Линейно неразделимая выборка

- Необходимые условия минимума:

$$0 = \frac{\partial L}{\partial w_j} = w_j - \sum_{i=1}^{l} \lambda_i y_i x_{ij}, \quad j = 1, ..., n, \; \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

$$0 = \frac{\partial L}{\partial w_0} = \sum_{i=1}^{N} \lambda_i y_i \; \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0.$$

$$0 = \frac{\partial L}{\partial \xi_i} = -\lambda_i - \eta_i + C \; \Rightarrow \eta_i + \lambda_i = C, \; i = 1, ..., l.$$

- Три типа объектов:
  1. $\lambda_i = 0; \; \eta_i = C; \; \xi = 0; \; M_i \geqslant 1$ - переферийные;
  2. $0 < \lambda_i < C; \; 0 < \eta_i < C; \; \xi = 0; \; M_i = 1$ - опорные граничные;
  3. $\lambda_i = ; \; \eta_i = 0; \; \xi > 0; \; M_i \leqslant 1$ - опорные нарушители;

Линейно неразделимая выборка

- Перепишем задачу:

$$\begin{cases} L(\mathbf{w}, w_0, \lambda) = -\sum\limits_{i=1}^{l} \lambda_i + 0.5 \sum\limits_{i=1}^{l} \sum\limits_{j=1}^{l} \lambda_i \lambda_j y_i y_j < x_i, x_j > \to \ \min_{\mathbf{w}, w_0} \max_{\lambda} \\ 0 \leqslant \lambda_i \leqslant C, \ i = 1, ..., l \\ \sum\limits_{i=1}^{N} \lambda_i y_i = 0. \end{cases}$$

- После решения задачи находим
  - $\mathbf{w} = \sum\limits_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$
  - $w_0 = med\{< w, x_i > -y_i : \ \lambda_i > 0, M_i = 1, i = 1, ..., l\}$

- Решение алгоритма:

$$a(x) = sign(\sum_{i=1}^{l} \lambda_i y_i < x_i, x > - w_0)$$

Ядра

- Можно подобрать преобразование $\psi : X \to H$ и перейти в пространство, где выборка окажется линейно разделимой.

- Теперь признаковыми описаниями являются $\psi(x_i)$, а скалярное произведение $< x, x^{'} >$ заменяется на $< \psi(x), \psi(x^{'}) >$. Т.е. в H должно быть скалярное произведение.

- **Опр.** Функция $K : X \times X \to R$ называется ядром (kernel function), если она представима в виде $K(x, x') =< \phi(x), \phi(x') >$ при некотором отображении $\phi : X \to H$, где H — пространство со скалярным произведением.

- SVM зависит только от $< x, x^{'} >$, а не от x_i. Следовательно, можно просто подбирать ядро, а не спрямляющее пространство.

Ядра

- **Теорема** (Мерсер, 1909). Функция $K(x, x')$ является ядром тогда и только тогда, когда она симметрична, $K(x, x') = K(x', x)$, и неотрицательно определена.
- **Опр.** Функция $K(x, x)$ неотрицательно определена, если для любой конечной выборки $X^p = (x_1, ..., x_p)$ из X матрица $K = || K(x_i, x_j) ||$ размера $p \times p$ неотрицательно определена: $z^T K z > 0$ для любого $z \in R^p$.
- **Утв. Свойства ядер:**
  1. константа - ядро;
  2. сумма ядер – ядро;
  3. произведение ядер – ядро;
  4. сумма равномерно сходящегося ряда ядер – ядро;
  5. композиция ядра и любого отображения (т.е. $K(\psi(\mathbf{x}), \psi(\mathbf{y}))$) – ядро.
  6. многочлен с положительными коэффициентами от ядра – ядро;
  7. композиция произвольного ядра и произвольной функции, представимой в виде степенного ряда с неотрицательными коэффициентами, - ядро.

Примеры ядер

- Линейное

$$K(x, x^{'}) =< x, x^{'} >$$

- Полиномиальное

$$K(x, x^{'}) = (< x, x^{'} > +r)^{d}$$

- Радиальное

$$K(x, x^{'}) = exp(-\gamma \mid\mid x - x^{'} \mid\mid^{2})$$