Алгоритмы кластеризации

Викулин Всеволод

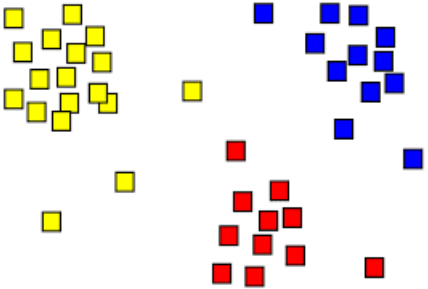
v.vikulin@corp.mail.ru

15 апреля 2019

Кластеризация. Реши задачу сам.

Задача кластеризации

Разбиение исходного набора объектов на группы таким образом, чтобы объекты в группе были похожи друг на друга, а объекты из разных групп - отличались. Обучение **без учителя**.



Примеры применения

- Сегментация
- Суммаризация
- Сжатие данные
- Обнаружение аномалий
- Тематическое моделирование
- В помощь для классификации/регрессии

На самом деле намного больше!

Мы учимся без учителя

When we're learning to see, nobody's telling us what the right answers are — we just look. Every so often, your mother says "that's a dog", but that's very little information. You'd be lucky if you got a few bits of information — even one bit per second — that way. The brain's visual system has 10^{14} neural connections. And you only live for 10^9 seconds. So it's no use learning one bit per second. You need more like 10^5 bits per second. And there's only one place you can get that much information: from the input itself. — Geoffrey Hinton, 1996

Какая бывает кластеризация?

Разнообразная, мы рассмотрим:

- Жесткая (k-means, агломеративная, dbscan)
- Мягкая (смесь распределений)
- Иерархическая (агломеративная)
- Выделяющая выбросы (dbscan)

Метрики качества кластеризации

Можно разделить на 2 типа:

1. Интутивные – свои близко, чужие подальше
2. По размеченным кластерам

Silhouette

Хотим, чтобы каждый объект к своему кластеру находился ближе, чем к соседнему.
Пусть C_i – кластер объекта i
a_i – среднее расстояние до объектов из кластера C_i,
b_i – среднее расстояние до объектов из ближайшего к i кластера (исключая C_i).

$$silhouette_i = \frac{b_i - a_i}{max(a_i, b_i)}$$

$$silhouette = \frac{1}{N} \sum_{i=1}^{N} \frac{b_i - a_i}{max(a_i, b_i)}$$

$silhouette \in [-1, 1]$

Rand Index

Пусть дана правильная кластеризация π^*, мы построили кластеризацию π.

Вопрос

Можно ли использовать метрики качества классификации?

Рассмотрим все пары объектов, проставим паре класс 1, если по π^* объекты из одного кластера и 0 – если из разных. Сделаем предскания для пар кластеризацией π.
Пусть a – число пар, где оба объекта принадлежат одному кластеру как в π^*, так и в π (True positive). b – число пар, где оба объекта принадлежат разным кластерам как в π^*, так и в π (True negative).

$$R = \frac{a+b}{C_N^2}$$

Давайте оптимизировать! Метод k-средних

Основные положения k-means

1. Жесткая кластеризация
2. Число кластеров K - структурный параметр алгоритма
3. Параметры - центры кластеров μ_k и $r_{n,k}$ - принадлежит ли кластеру k объект n.
4. Оптимизирует $Q = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} ||x_n - \mu_k||^2$,

где $r_{n,k}$ равен 1, если x_n принадллежит k кластеру, а иначе равен 0.

Вопрос

Сколько всего параметров? Как будем находить их?

Оптимизируем функционал

Смотрим на функционал качества $Q = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} ||x_n - \mu_k||^2$.
Разделим оптимизацию на 2 шага – сначала оптимизируем $r_{n,k}$ при фиксируемых μ_k, затем оптимизируем μ_k при фиксируемых $r_{n,k}$
Фиксируем центры кластеров. Как минимизировать функционал по $r_{n,k}$? Просто берем ближайший кластер!
Фиксируем, к кому кластеру какой объект принадлежит? Как посчитать средние?

$$\nabla_\mu Q = 2 \sum_n r_{n,k}(x_n - \mu_k) = 0$$

$$\mu_k = \frac{\sum_n r_{n,k} x_n}{\sum_n r_{n,k}}$$

Смысл - это просто среднее значение в кластере!

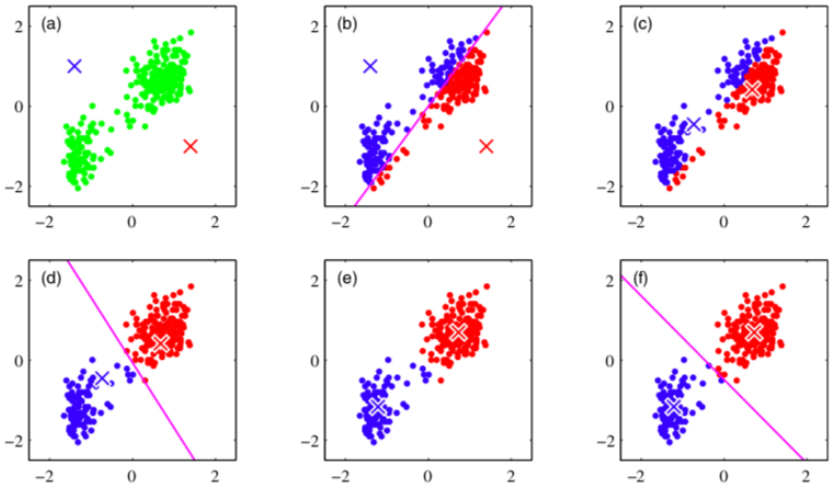
Алгоритм k-means

1. Инициализируем случайно μ_k
2. Для каждого n найти ближайший к нему кластер, то есть посчитать $r_{n,k}$
3. Пересчитать центры кластеров $\mu_k = \frac{\sum_n r_{n,k} x_n}{\sum_n r_{n,k}}$
4. Повторять (2),(3) до сходимости

Домашнее задание

Сходится ли алгоритм k-средних?

Итерации k-means



Источник: Bishop

Пример работы k-means

$K = 2$

$K = 3$

$K = 10$

Original image

Источник: Bishop

Выбор числа кластеров

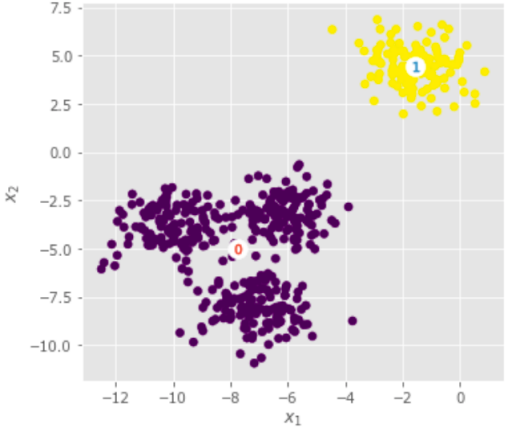
Функционал качества $Q = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} ||x_n - \mu_k||^2$. Как выбрать число кластеров? Параметры, которые нельзя настраивать во время обучения, называют **структурными параметрами**.

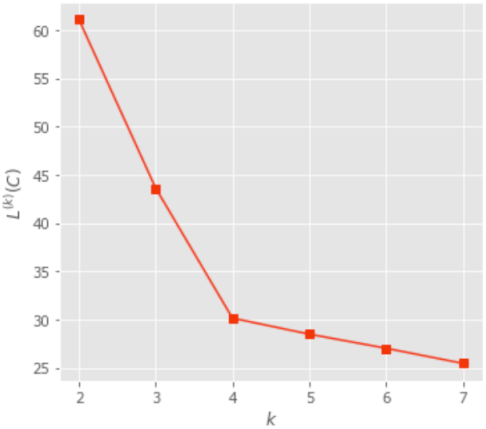
Вопрос

Какие мы уже знаем структурные параметры?

Метод локтя

Построим зависимость функции потерь от числа кластеров. Выберем точку, после которой функционал уже не сильно меняется.





Начальная инициализация

Результат работы зависит от начальной инициализации. В оригинальном алгоритме она берется случайно - результат не стабилен. Можно запустить много раз, а затем выбрать лучший вариант.
Алгоритм k-means ++.
Делаем умную инициализацию весов. Первый центроид берем случайно. Точка становится центроидом с вероятностью пропорционально удалению от ближайшего из предыдущих центроидов.

k-medoids

Функционал качества $Q = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} ||x_n - \mu_k||^2$. А если у нас есть только расстояния между объектами?

1. Инициализируем случайно медианные объекты
2. Для каждого объекта находим ближайший к нему
3. Найти медианные объекты
4. Повторять (2),(3) до сходимости

Вопрос

Как сделать k-medoids на графе городов?

Итоги k-means

Плюсы:

- Быстро считается, быстро сходится
- Можно дообучать на новых данных
- Можно предсказывать кластера для новых объектов
- Огромное количество реализаций

Минусы:

- Придется запускать много раз
- Число кластеров выбирать самому
- Выбросы могут все сломать

Иерархические алгоритмы. Долго и качественно.

Когда нужна иерархия?

Основные положения иерархических алгоритмов

1. Жесткая кластеризация
2. Результат для любого числа кластеров
3. Строит иерархию
4. Нужно уметь считать расстояние $\rho(x, x')$

Вопрос

Какое расстояние будем использовать для веб-страниц?

Вверх или вниз

Два варианта алгоритмов.
Аггломеративные алгоритмы:

1. Начинаем с ситуации, когда каждый кластер это один объект
2. На каждом шаге объединяем два ближайших кластера
3. Останавливаемся, когда все объединили в один большой кластер

Дивизивные алгоритмы:

1. Начинаем с ситуации, когда все объекты в одном кластере
2. На каждом шаге разъединяем на два самый "разъединяемый"кластер
3. Останавливаемся, когда каждый кластер это один объект

Обычно используют аггломеративные.

Расстояния между кластерами

Какая-либо агрегация расстояний между объектами кластеров. linkage - связь.

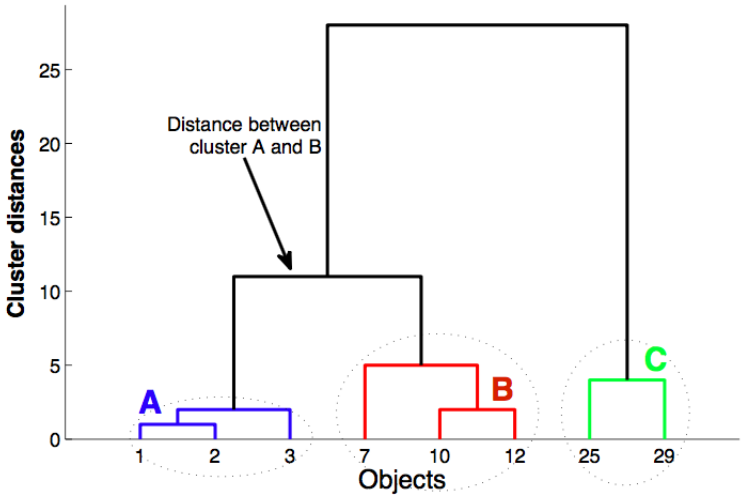
1. Single linkage - минимальное расстояние между объектами двух кластеров
2. Complete linkage - максимальное расстояние между объектами двух кластеров
3. Average linkage - среднее расстояние между объектами двух кластеров
4. Centroid linkage - расстояние между центрами кластеров
5. Все, что сами придумаете

Вопрос

Как считать расстояния для объектов с вещественными и категориальными признаками?

Дендрограмма

Наглядный образ представить данные.



Формулы Ланса-Вильямса

Хотим посчитать расстояние d между $C_i \bigcup C_j$ и C_k. Для почти всех вменяемых метрик, это можно посчитать, зная $d(C_i, C_k)$ и $d(C_j, C_k)$.

$$d(C_i \bigcup C_j, C_k) = a \cdot d(C_i, C_k) + b \cdot d(C_j, C_k) + c \cdot |d(C_i, C_k) - d(C_j, C_k)|$$

	a_i	b	c
Single link	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete link	$\frac{1}{2}$	0	$\frac{1}{2}$
Centroid	$\frac{n_i}{n_i+n_j}$	$-\frac{n_i n_j}{(n_i+n_j)^2}$	0
Median	$\frac{1}{2}$	$-\frac{1}{4}$	0
Group average link	$\frac{n_i}{n_i+n_j}$	0	0
Ward's method	$\frac{n_i+n_k}{n_i+n_j+n_k}$	$-\frac{n_k}{n_i+n_j+n_k}$	0

Домашнее задание

Докажите для некоторых linkage

Итоги иерархических алгоритмов

Плюсы:

- Иерархия кластеров
- Нужно только задать расстояния
- Выдает любое число кластеров
- Анализ данных с помощью денденограммы
- Понятно, как найти выбросы

Минусы:

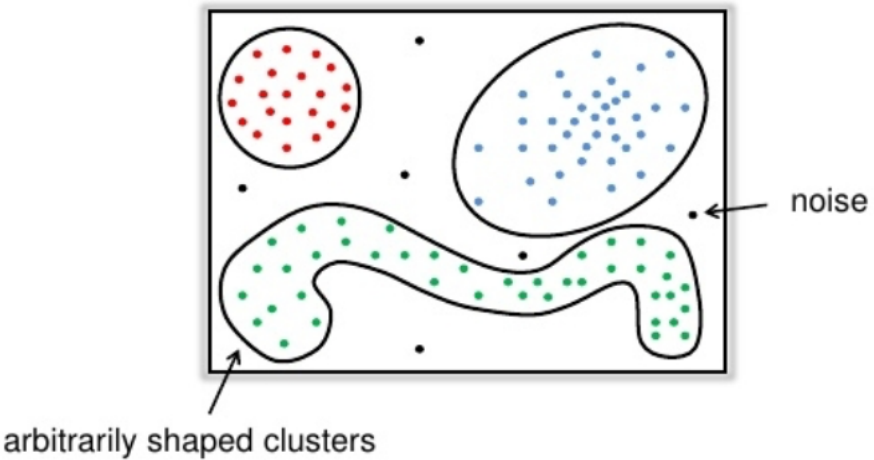
- Очень долго
- Нельзя дообучать
- Нельзя предсказывать на новых объектах
- Не всегда удобно задавать расстояния

Часть 4

Плотностные алгоритмы. Кластер моего кластера мой кластер.

Мотивация

Получаем кластеры высокой плотности, разделеные участками низкой плотности

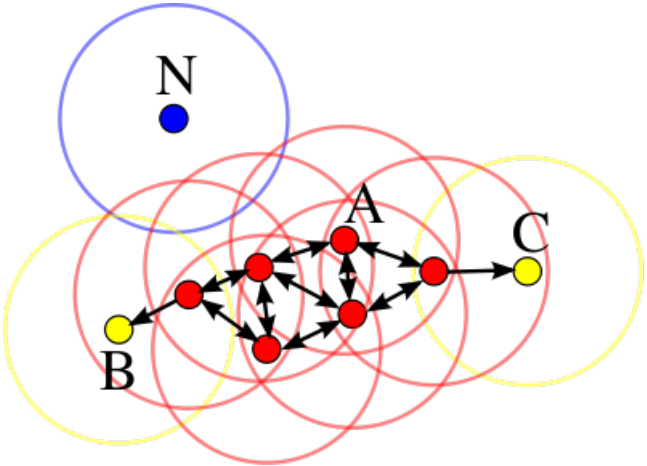


Основные положения плотностных алгоритмов

1. Жесткая кластеризация
2. Число кластеров получается само
3. Находит выбросы
4. Нужно уметь считать расстояние $\rho(x, x')$

Простые определения: core объект - объект, в ϵ окрестности которого не меньше N_{thr} объектов, граничный объект - не core, но в его ϵ окретсности есть core объект, выброс - не core и не граничный.

Более наглядно



Алгоритм DBSCAN

```
function dbscan(X, eps, min_pts):
    initialize NV = X # not visited objects
    for x in NV:
        remove(NV, x) # mark as visited
        nbr = neighbours(x, eps) # set of neighbours
        if nbr.size < min_pts:
            mark_as_noise(x)
        else:
            C = new_cluster()
.            expand_cluster(x, nbr, C, eps, min_pts, NV)
.            yield C
```

Алгоритм DBSCAN

```
function expand_cluster(x, nbr, C, eps, min_pts, NV):
    add(x, C)
    for x1 in nbr:
        if x1 in NV: # object not visited
            remove(NV, x1) # mark as visited
            nbr1 = neighbours(x1, eps)
            if nbr1.size >= min_pts:
                # join sets of neighbours
                merge(nbr, nbr_1)
         if x1 not in any cluster:
             add(x1, C)
```

Итоги плотностных алгоритмов

Плюсы:

- Сам определяет нужное число кластеров
- Находит выбросы
- Можно предсказывать для новых объектов

Минусы:

- Дольше, чем k-means
- Нельзя дообучать
- Не работает, если кластера разной плотности
- Нужно подбирать параметры ϵ, N_{thr} перед запуском

Заключение

Спасибо за внимание!