Алгоритмы кластеризации

Сергей Николенко

Машинное обучение — ИТМО, осень 2006

Outline

Суть лекции

- *Кластеризация* — типичная задача статистического анализа: задача классификации объектов одной природы в несколько групп так, чтобы объекты в одной группе обладали одним и тем же свойством.
- Под свойством обычно понимается близость друг к другу относительно выбранной метрики.

Чуть более формально

- Есть набор тестовых примеров $X = \{x_1, \ldots, x_n\}$ и функция расстояния между примерами ρ.
- Требуется разбить X на непересекающиеся подмножества (кластеры) так, чтобы каждое подмножество состояло из похожих объектов, а объекты разных подмножеств существенно различались.

Зачем это нужно

- Data mining в самом широком смысле; например, разделить веб–документы на жанры и т.п.
- Распознавание образов.
- Анализ социальных сетей.
- ...

Виды кластеризации

- Иерархические: последовательно строим кластеры из уже найденных кластеров.
  - Агломеративная (объединительная) — начинаем с индивидуальных элементов, затем объединяем.
  - Разделительная — начинаем с одного кластера, потом делим.
- Неиерархические: оптимизируем некую целевую функцию.
  - Алгоритмы теории графов.
  - Алгоритм EM.
  - Алгоритм k–средних (k–means).
  - Нечеткие алгоритмы.

Оценка кластеризации

- Действительно, пусть есть несколько разбиений на кластеры. Как их сравнить?
- Пусть нам нужно кластеризовать точки в d–мерном пространстве.

Оценка кластеризации

- Действительно, пусть есть несколько разбиений на кластеры. Как их сравнить?
- Пусть нам нужно кластеризовать точки в d–мерном пространстве.
- Логично минимизировать среднее внутрикластерное расстояние:

$$\frac{\sum_{i<j, c(x_i)=c(x_j)} \rho(x_i, x_j)}{\sum_{i<j, c(x_i)=c(x_j)} 1} \longrightarrow \min .$$

- И максимизировать среднее межкластерное расстояние:

$$\frac{\sum_{i<j, c(x_i)\neq c(x_j)} \rho(x_i, x_j)}{\sum_{i<j, c(x_i)\neq c(x_j)} 1} \longrightarrow \max .$$

Оценка кластеризации

- Действительно, пусть есть несколько разбиений на кластеры. Как их сравнить?
- Пусть нам нужно кластеризовать точки в d–мерном пространстве.
- Если алгоритм вычисляет центры кластеров μ_c, то можно более эффективно:

$$\sum_{c \in C} \frac{1}{|c|} \sum_{x \in c} \rho^2(x, \mu_c) \longrightarrow \min,$$

$$\sum_{c \in C} \rho^2(\mu_c, \mu) \longrightarrow \max,$$

  где μ — общий центр масс всей выборки.
- Чтобы учесть и стремление к минимуму, и стремление к максимуму, устремляют к минимуму отношение двух

Outline

Идея

- Есть точки $x_1, x_2, \ldots, x_n$ в пространстве. Нужно кластеризовать.
- Считаем каждую точку кластером. Затем ближайшие точки объединяем, далее считаем единым кластером. Затем повторяем.
- Получается дерево.

Алгоритм

$\texttt{HierarchyCluster}(X = \{x_1, \ldots, x_n\})$

- Инициализируем $C = X$, $G = X$.
- Пока в C больше одного элемента:
  - Выбираем два элемента C c_1 и c_2, расстояние между которыми минимально.
  - Добавляем в G вершину c_{12}, соединяем ее с вершинами c_1 и c_2.
  - $C := C \cup \{c_{12}\} \setminus \{c_1, c_2\}$.
- Выдаем G.

Результат

- В итоге получается дерево кластеров, из которого потом можно выбрать кластеризацию с требуемой степенью детализации (обрезать на том или ином максимальном расстоянии).
- Все ли понятно?

Результат

- В итоге получается дерево кластеров, из которого потом можно выбрать кластеризацию с требуемой степенью детализации (обрезать на том или ином максимальном расстоянии).
- Все ли понятно?
- Остается вопрос: как подсчитывать расстояние между кластерами?

Single-link vs. complete-link

- *Single-link* алгоритмы считают *минимум* из возможных расстояний между парами объектов, находящихся в кластере.
- *Complete-link* алгоритмы считают *максимум* из этих расстояний.
- Какие особенности будут у single-link и complete-link алгоритмов? Чем они будут отличаться?

Упражнение

Упражнение

Реализовать single-link и complete-link алгоритмы агломеративной кластеризации для точек из евклидова пространства размерности n.

Outline

Очевидный алгоритм

- Нарисуем полный граф с весами, равными расстоянию между объектами.
- Выберем лимит расстояния r и выбросим все ребра длиннее r.
- Компоненты связности полученного графа — это наши кластеры.

Минимальное остовное дерево

- Минимальное остовное дерево — дерево, содержащее все вершины (связного) графа и имеющее минимальный суммарный вес своих ребер.
- Алгоритм Краскала (Kruskal): выбираем на каждом шаге ребро с минимальным весом, если оно соединяет два дерева, добавляем, если нет, пропускаем.

Алгоритм Борувки (Boruvka)

- Начинаем с графа G и пустого множества ребер T.
- Пока $G|_T$ не связный:
  - Инициализируем множество ребер $E := \emptyset$.
  - Для каждой компоненты связности:
    - Инициализируем множество ребер $S := \emptyset$.
    - Для каждой вершины компоненты добавляем в S самое дешевое ребро, соединяющее вершину с какой-либо вершиной другой компоненты.
    - Добавляем самое дешевое ребро из S в E.
  - $T := T \cup E$.
- Он работает быстрее алгоритма Краскала (за время $O(E \log V)$).
- Впрочем, самый быстрый из известных алгоритмов MST работает за время $O(E\alpha(V))$, где α — обратная к функции Аккермана.

Кластеризация

- Как использовать минимальное остовное дерево для кластеризации?

Кластеризация

- Как использовать минимальное остовное дерево для кластеризации?
- Построить минимальное остовное дерево, а потом выкидывать из него ребра максимального веса.
- Сколько ребер выбросим, столько кластеров получим.

Упражнение

Упражнение

Реализовать алгоритм кластеризации методом минимального остовного дерева.

Outline

Идея алгоритма

- Был придуман для кластеризации генов.
- Однако, вопреки ожиданиям, не слишком эффективен; для генома нечего и думать его запускать.

Алгоритм

- Зафиксируем quality threshold — предельное расстояние между объектами кластера.
- Выбрать случайный ген и начать наращивать кластер, пока это возможно (не превышая предельного расстояния).
- Повторить для других случайных генов, пока все гены не будут покрыты (пересечения кластеров возможны).
- Выбрать кластер максимального размера и удалить его из множества тестовых объектов.
- Запускать то же самое рекурсивно до тех пор, пока размер максимального кластера не станет слишком большим.

Плюсы и минусы

- Достоинства:
  - Гарантированное качество: все кластеры соответствуют заранее заданным критериям.
  - Число кластеров задавать не нужно.
  - Рассматриваются все возможные кластеры: exhaustive search.
- Недостатки:
  - Относительно медленно работает.
  - Нужно задавать threshold; если он окажется плохо подобран, алгоритм будет работать *очень* медленно.

Упражнение

Упражнение

Реализовать алгоритм кластеризации QT.

История

- FOREL означает «формальный элемент». Единственный из популярных алгоритмов кластеризации, изобретенный в СССР.
- Предложен Загоруйко и Елкиной в 1967 году для решения одной палеонтологической задачи.

Идея

- Предположим, что мы находимся в линейном пространстве ($\mathbb{R}^n$ подходит).
- Пусть задана точка x_0 и параметр R.
- Выделим все точки из тестового набора, попадающие в сферу радиуса R с центром в x_0.
- Перенесем центр кластера из x_0 в точку центра масс точек, которые были выбраны на предыдущем шаге.
- Повторять, пока центр не останется на месте.
- Центр — это и есть «формальный параметр», т.к. он не обязательно содержится в исходной выборке.

Алгоритм

FOREL(X):

- Инициализировать множество некластеризованных точек $U := X$ и множество кластеров $C := \emptyset$.
- Пока $U \neq \emptyset$:
  - Выбрать случайную точку x_0.
  - Пока процесс не стабилизируется:
    - Образовать кластер $c = \{x \in X | \rho(x, x_0) < R\}$.
    - $x_0 := \frac{1}{|c|} \sum_{x \in c} x$.
  - $U := U \setminus c$, $C := C \cup \{c\}$.
- Выдать множество полученных кластеров C.

Упражнение

Упражнение

Реализовать алгоритм кластеризации FOREL.

Спасибо за внимание!

- Lecture notes, слайды и коды программ появятся на моей homepage:

  `http://logic.pdmi.ras.ru/~sergey/index.php?page=teaching`

- Присылайте любые замечания, коды программ на других языках, решения упражнений, новые численные примеры и прочее по адресам:
  `sergey@logic.pdmi.ras.ru`, `smartnik@inbox.ru`