



# Машинное обучение

## Лекция 10. Кластеризация

**Автор:** Рустам Азимов

Санкт-Петербургский государственный университет

Санкт-Петербург

- Отсутствует целевая переменная
- Требуется восстановить некую скрытую структуру в данных
- Примеры:
  - ▶ **Задача кластеризации**
  - ▶ Задача уменьшения размерности
  - ▶ Задача визуализации

# Задача кластеризации

- $X$  — признаки, например вещественные числа
- Требуется разбить выборку на непересекающиеся подмножества, называемые кластерами,
- Хотим, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались
- То есть построить функцию:

$$\alpha : \mathbb{X} \rightarrow \{1, \dots, K\}$$

- Число кластеров  $K$  может либо быть известно, либо являться параметром

- Кластеризовать новости по сюжетам
- Пиксели на изображении по принадлежности объекту
- Музыку по жанрам
- Сообщения на форуме по темам
- Клиентов по типу поведения
- Класстеризация бывает иерархическая (нисходящая, восходящая) или нет
- Как измерять сходство объектов?

- Существует два подхода к измерению качества кластеризации:
  - ▶ Внутренний — основан на некоторых свойствах выборки и кластеров
  - ▶ Внешний — использует дополнительные данные, например, информацию об истинных кластерах
- Для внутренних метрик качества каждый кластер зачастую характеризуется своим центром (центроидом)  $c_k$
- Внешние метрики возможно использовать, если известно истинное распределение объектов по кластерам и задачу кластеризации можно рассматривать как задачу многоклассовой классификации с любой метрикой оттуда

# Примеры внутренних метрик качества

## 1 Внутрикластерное расстояние:

- ▶  $\sum_{k=1}^K \sum_{i=1}^n [\alpha(x_i) = k] \rho(x_i, c_k)$
- ▶  $\rho(x, z)$  — некоторая функция расстояния
- ▶ Данный функционал требуется минимизировать, поскольку все объекты кластера должны быть схожи (близки)

## 2 Межкластерное расстояние:

- ▶  $\sum_{i,j=1}^n [\alpha(x_i) \neq \alpha(x_j)] \rho(x_i, x_j)$
- ▶ Данный функционал нужно максимизировать, поскольку объекты из разных кластеров должны существенно отличаться

## 3 Индекс Данна (Dunn Index):

- ▶  $\frac{\min_{1 \leq k < k' \leq K} d(k, k')}{\max_{1 \leq k \leq K} d(k)}$
- ▶  $d(k, k')$  — расстояние между кластерами  $k$  и  $k'$  (например, евклидово расстояние между их центрами)
- ▶  $d(k)$  — внутрикластерное расстояние для  $k$ -го кластера (например, сумма расстояний от всех объектов этого кластера до его центра)
- ▶ Данный индекс необходимо максимизировать

- Одним из наиболее популярных методов кластеризации является **K-Means**
- Оптимизирует внутрикластерное расстояние с квадратом евклидовой метрики в качестве расстояния
- Нужно подбирать центры кластеров  $c_k$  и распределение объектов по кластерам  $\alpha(x_i)$
- Выберем для этих величин произвольные начальные приближения, а затем будем оптимизировать их по очереди

- 1 Зафиксируем центры кластеров. В этом случае внутрикластерное расстояние будет минимальным, если каждый объект будет относиться к тому кластеру, чей центр является ближайшим:

$$\alpha(x_i) = \arg \min_{1 \leq k \leq K} \rho(x_i, c_k)$$

- 2 Зафиксируем распределение объектов по кластерам. В этом случае внутрикластерное расстояние с квадратом евклидовой метрики можно продифференцировать по центрам кластеров и вывести аналитические формулы для них:

$$c_k = \frac{1}{\sum_{i=1}^n [\alpha(x_i) = k]} \sum_{i=1}^n [\alpha(x_i) = k] x_i$$



- Повторяя эти шаги до сходимости, мы получим некоторое распределение объектов по кластерам
- Новый объект относится к тому кластеру, чей центр является ближайшим
- Не гарантируется достижение глобального минимума суммарного квадратичного отклонения, а только одного из локальных минимумов
- Число кластеров надо знать заранее
- Результат работы метода K-Means существенно зависит от начального приближения
- Существует большое количество подходов к инициализации, а одним из наиболее успешных считается **k-means++**

- Графовые методы кластеризации — это простейшие методы, которые основаны на построении графа близости
- Убираются рёбра между объектами, расстояния между которыми больше определённого порога
- Кластерами же объявляются группы объектов, попадающих в одну компоненту связности

## AFFINITY PROPAGATION

INPUT:  $\{s(i, j)\}_{i, j \in \{1, \dots, N\}}$  (data similarities and preferences)

INITIALIZE: set 'availabilities' to zero *i.e.*  $\forall i, k: a(i, k) = 0$

REPEAT: responsibility and availability updates until convergence

$$\forall i, k: r(i, k) = s(i, k) - \max_{k': k' \neq k} [s(i, k') + a(i, k')]$$

$$\forall i, k: a(i, k) = \begin{cases} \sum_{i': i' \neq i} \max[0, r(i', k)], & \text{for } k = i \\ \min \left[ 0, r(k, k) + \sum_{i': i' \notin \{i, k\}} \max[0, r(i', k)] \right], & \text{for } k \neq i \end{cases}$$

OUTPUT: cluster assignments  $\hat{c} = (\hat{c}_1, \dots, \hat{c}_N)$ ,  $\hat{c}_i = \operatorname{argmax}_k [a(i, k) + r(i, k)]$

Note:  $\hat{c}$  may violate  $\{f_k\}$  constraints, so initialize  $k$ -medoids with  $\hat{c}$  and run to convergence for a coherent solution.

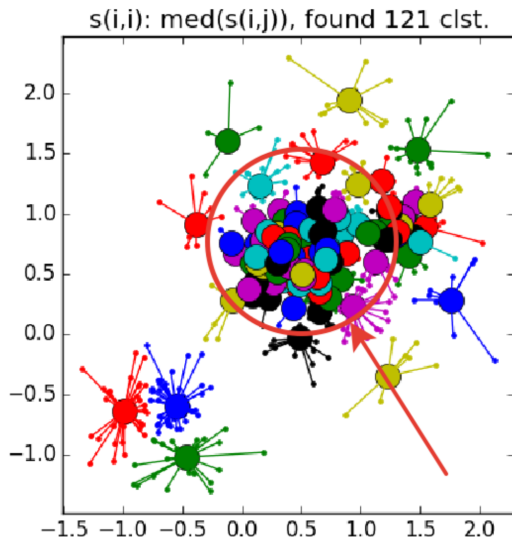
- **Affinity propagation** (AP, он же метод распространения близости) получает на вход матрицу схожести между элементами датасета  $S : n \times n$
- Возвращает набор меток, присвоенных этим элементам
- $s(i, k)$  — насколько похожа точка  $i$  на соседа  $k$ , почти никогда не меняется
- Точки хотят объединяться в группы вокруг одного лидера
- Каждая точка хотела бы видеть лидером кого-то, кто максимально на неё похож, но готова мириться с другими кандидатами, если те нравятся многим другим
- Точки сами не сильно хотят быть лидерами  $s(k, k) < 0$

## Responsibility and availability

- **Ответственность** (responsibility, таблица  $R$  с элементами  $r(i, k)$ ) отвечает за то, насколько  $i$  хочет видеть  $k$  своим предводителем
- Ответственность возлагается каждой точкой на кандидата в лидеры группы
- **Доступность** (availability, таблица  $A$ , с элементами  $a(i, k)$ ) — есть ответ от потенциального предводителя  $k$ , насколько хорошо  $k$  готова представлять интересы  $i$
- Ответственность и доступность точки вычисляют в том числе и сами для себя
- Только когда велика самоответственность (да, я хочу представлять свои интересы) и самодоступность (да, я могу представлять свои интересы), т.е.  $a(k, k) + r(k, k) > 0$ , точка может перебороть врождённую неуверенность в себе
- Точки в конечном итоге присоединяются к лидеру, для которого у них наибольшая сумма  $a(i, k) + r(i, k)$

- Affinity propagation, как и многие другие алгоритмы, можно прервать досрочно, если  $R$  и  $A$  перестают обновляться
- Affinity propagation подвержен вычислительным осцилляциям в случаях, когда есть несколько хороших разбиений на кластеры
  - ▶ В самом начале к матрице сходства добавляется немного шума порядка  $10^{-16}$
  - ▶ При обновлении и используется не простое присваивание, а присваивание с экспоненциальным сглаживанием
  - ▶ Параметр  $\gamma = 0.5$ , но если есть проблемы со сходимостью, то увеличиваем до 0.9 или 0.95 и увеличиваем количество итераций
- Параметр самоподобия  $s(k, k)$  чем меньше, тем крупнее кластеры
  - ▶ Используйте медиану по всем  $s(i, k)$  для большого количества кластеров
  - ▶ 25 перцентиль или даже минимум — для меньшего
- Пример  $s(i, k)$  — отрицательное евклидово расстояние

## Пример отказа AP

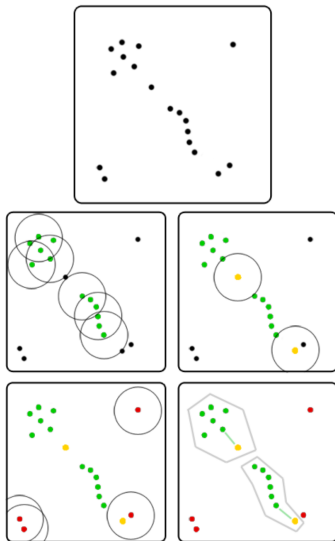


- Когда у вас не очень большой ( $n < 10^6$ ) или в меру большой, но разреженный ( $n < 10^7$ ) датасет
- Заранее известна функция близости
- Количество кластеров значения не имеет
- Свойства функции близости значения не имеют
- Вы ожидаете увидеть множество кластеров различной формы и немного варьирующимся количеством элементов



- **DBSCAN** (Density-based spatial clustering of applications with noise, плотностный алгоритм пространственной кластеризации с присутствием шума), как следует из названия, оперирует плотностью данных
- На вход он просит уже знакомую матрицу близости и два параметра — радиус  $\varepsilon$ -окрестности и количество соседей  $m$
- Обходим (например, в ширину) все точки и смотрим сколько соседей в  $\varepsilon$ -окрестности
- Если соседей  $\geq m$ , то помечаем точку зелёным цветом
- Если соседей  $< m$ , но в  $\varepsilon$ -окрестности есть зелёная точка, то помечаем жёлтым цветом, иначе красным
- Зелёные соседи в одном кластере, жёлтые — пограничные точки нужно как-то решить какому классу отдать, а красные — отшельники (тоже надо решить, что делать)

# Пример DBSCAN



- DBSCAN с неслучайным правилом обработки краевых точек детерминирован
- Однако большинство реализаций для ускорения работы и уменьшения количества параметров отдают краевые точки первым кластерам, которые до них дотянулись
- DBSCAN автоматически определяет выбросы
- Соотношение  $\frac{m}{\epsilon^n}$ , где  $n$  — размерность пространства, можно интуитивно рассматривать как пороговую плотность точек данных в области пространства
- При одинаковом соотношении  $\frac{m}{\epsilon^n}$  и результаты будут примерно одинаковы
- Главные недостатки DBSCAN — неспособность соединять кластеры через проёмы, и, наоборот, способность связывать явно различные кластеры через плотно населённые перемиčky

- <http://www.machinelearning.ru/>
- <https://scikit-learn.org>
- <https://habr.com/ru/post/321216/>
- <https://habr.com/ru/post/322034/>