

# Базовые модели машинного обучения: кластеризация

---

Гирдюк Дмитрий Викторович  
Першин Антон Юрьевич, Ph.D.  
Никольская Анастасия Николаевна

Программа «Большие данные и распределенная цифровая  
платформа»

Санкт-Петербургский государственный университет

Практика по дисциплине «Технологии ИИ»  
29 апреля 2023 г.

- *Обучение без учителя (unsupervised learning)* – это тип машинного обучения, который ищет ранее необнаруженные закономерности в наборе данных без ранее существовавших меток и с минимальным или полностью отсутствующим контролем человека.
- *Задача обучения без учителя покрывает не только кластеризацию, но и*
  - *поиск ассоциативных правил*
  - *заполнение пропущенных значений*
  - *поиск аномалий*
  - *сокращение размерности и визуализация данных*

# Постановка задачи кластеризации

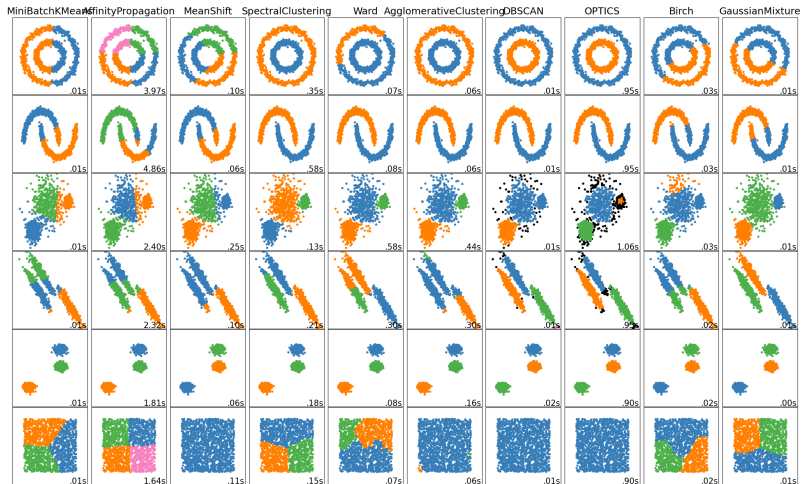
- *Кластерный анализ* или *кластеризация* – это задача группировки набора объектов таким образом, чтобы объекты в одной группе (называемой кластером) были более похожи (в некотором смысле) друг на друга, чем на объекты в других группах (кластерах).
- Проще говоря, имеем
- Пространство объектов  $X = \{x_i\}_{i=1}^n, x_i \in \mathbb{R}^m$  и выборку из него  $\hat{X}$
  - Мера расстояния между объектами  $\rho : X \times X \rightarrow R^+$

Хотим получить

- Множество групп/кластеров  $Y = \{y_i\}_{i=1}^n, y_i \in \mathbb{N}$
- Алгоритм кластеризации  $\alpha : X \rightarrow Y$

- Разделение на группы с целью упрощения работы (отдельные модели для каждой группы).
- Сокращение объемов наблюдений и сжатие данных (например, квантизация нейронных сетей).
- Выделение новизны/аномалий.
- Построение иерархии/таксономии объектов.

# Примеры кластеризации



Источник изображения

В большинстве источников выделяют пять групп алгоритмов

- *Основанные на центроидах* (centroid based): **k-means**, k-modes, k-medoids, **Meanshift**, FCM, Affinity propagation.
- *Иерархические* (hierarchical): **агломеративные** (Ward, single/average/complete linkage), BIRCH, на основе теории графов (выделение связных компонент и минимальное остовное дерево), **Spectral Clustering**, CURE, ROCK, Chameleon, Echidna, SNN, CACTUS, GRIDCLUST.
- *Основанные на плотности* (density based): **DBSCAN**, OPTICS, DBCLASD, GDBSCAN, DENCLU, SUBCLU.
- *Сеточные* (grid based): STING, Wave cluster, BANG, CLIQUE, OptiGrid, MAFLA, ENCLUS, PROCLUS, ORCLUS, FC, STIRR.
- *Основанные на модели данных* (model based): **Expectation Maximization** (EM), COBWEB, CLASSIT, SOM.

- Алгоритм k-means – один из самых простых и популярных алгоритмов кластеризации, заключающийся в поиске заранее заданного числа кластеров путем минимизации суммы квадратов внутрикластерных расстояний между точками кластеров и соответствующих им центроидами.
- Получаемая оптимизационная задача вычислительно трудна (NP-сложная), однако разработано достаточно много эвристических алгоритмов, позволяющих достаточно быстро отыскать локальный минимум.

- Пусть  $S = \{S_1, S_2, \dots, S_k\}$  есть разбиение выборки на  $k$  непересекающихся множеств.
- Тогда задача минимизации суммы квадратов внутрикластерных расстояний может быть записана следующим образом

$$\arg \min_S \sum_{i=1}^k \sum_{x \in S_i} \|x - \mu_i\|^2 = \arg \min_S \sum_{i=1}^k |S_i| \text{Var} S_i,$$

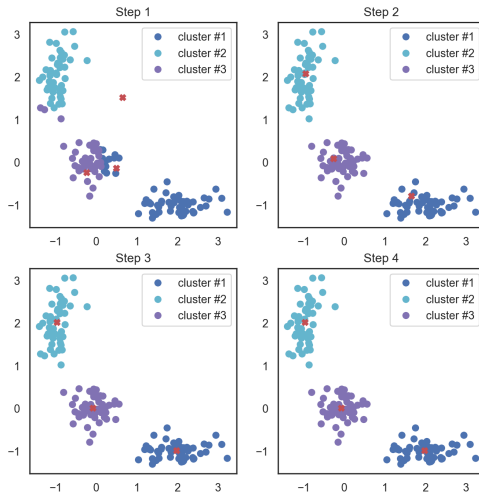
$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$$



## k-means: описание алгоритма Ллойда

1. Зафиксировав число кластеров, произвольным образом инициализируем  $k$  центроид. Это могут быть наблюдения из выборки, произвольные точки в пространстве данных или что-нибудь более умное, вроде использования эмпирической плотности распределения данных (k-means++).
2. Относим каждое наблюдение к кластеру, чей центроид (центр тяжести) находится к наблюдению ближе всего.
3. Обновляем центроиды с учетом всех входящих в каждый кластер наблюдений.
4. Повторяем шаги 2 и 3 фиксированное количество раз или до тех пор, пока все центроиды не стабилизируются (т.е. изменяются по норме не больше заранее заданного  $\varepsilon$ ).

# k-means: пример



Источник изображения

---

**Algorithm 1:** k-means

---

**input** : Выборка  $X = \{x_1, \dots, x_n\}$ , количество кластеров  $k$  и  $\varepsilon$   
Произвольным образом инициализируем  $k$  центроид  $\mu_i$   
**while** *True* **do**

    Относим наблюдения к ближайшим центроидам:

$$S_i := \{x_p : \|x_p - \mu_i\|^2 \leq \|x_p - \mu_j\|^2, 1 \leq j \leq k\}$$

    Обновляем центроиды:

$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$$

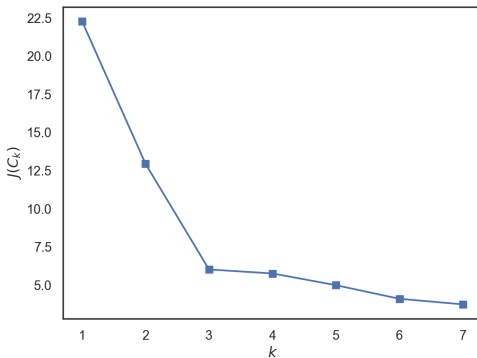
**if**  $\max_{i,j} |g_{ij} - g_{ij}^0| < \varepsilon$  **then**  
        └ Завершить работу

---

- Отличное базовое решение.
- Алгоритм метрический, потому либо нормализуем данные, либо определяем специальную метрику.
- Алгоритм прост и понятен, имеет большое количество всевозможных обобщений (k-medians, k-medoids, k-means++ и т.д.).
- Не гарантируется достижение глобального минимума суммарного квадратичного отклонения, а только одного из локальных минимумов. Кроме того, полученные кластеры зачастую стремятся иметь сферическую форму ввиду вида целевой функции.
- Результат зависит от выбора исходных центров кластеров, их оптимальный выбор неизвестен. Чаще всего алгоритм запускают несколько раз и выбирают то разбиение, которое показало наименьшее значение целевой функции.

## k-means: эвристика выбора числа кластеров

→ Число кластеров необходимо задавать заранее. На практике производят разбиения для различных значений  $k$ , строят график значений целевой функции и ищут такое значение  $k$ , после которого значение целевой функции перестает сильно изменяться (так называемый «метод локтя»).



Источник изображения

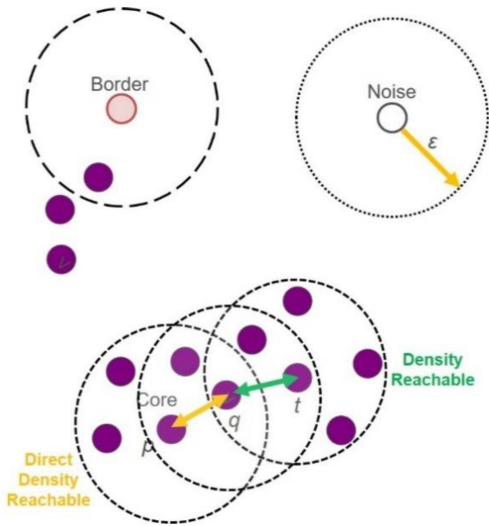
- k-means реализован в sklearn'е.
- Кроме числа кластеров `n_clusters` основными гиперпараметрами являются:
  - `init`. Метод инициализации центроид. `k-means++` (стандартный), случайный и определенный пользователем.
  - `n_init`. Количество запусков алгоритма (было описано ранее на слайдах).
  - `algorithm`. Алгоритм Ллойда или Элкана. Второй является модификацией алгоритма Ллойда: ускорение работы путем исключения некоторого числа вычислений расстояний за счет использования неравенства треугольника.

- Density-Based Spatial Clustering of Applications with Noise (DBSCAN) – алгоритм кластеризации, основанный на плотности точек, изначально разработанный с целью кластеризации в базах данных, содержащих геометрические представления наблюдений.
- Основными преимуществами алгоритма авторы выделили минимальную необходимость понимания предметной области данных при подборе гиперпараметров метода, а также способность обнаруживать кластеры произвольной формы.
- Алгоритм достаточно прост, наряду с k-means один из самых популярных.

- Алгоритм имеет 2 гиперпараметра: величина окрестности точки  $\varepsilon$  и минимальное количество наблюдений в окрестности *MinPts*
- При кластеризации точка может быть причислена к одному из 3 типов:
  - корневая: в ее  $\varepsilon$ -окрестности не менее *MinPts* точек
  - граничная: в ее  $\varepsilon$ -окрестности меньше *MinPts* точек, но среди них есть как минимум одна корневая
  - шумовая: не корневая и не граничная



# DBSCAN: иллюстрация типов наблюдений



Источник изображения

---

## Algorithm 2: DBSCAN

---

**input** : Выборка  $X = \{x_1, \dots, x_n\}$ , параметры  $\varepsilon$  и  $MinPts$

$U = X, K = \emptyset, a = 0, a_1, \dots, a_n = -1;$

**while**  $U \neq \emptyset$  **do**

    Взять  $x \in U;$

**if**  $|U_\varepsilon(x)| < MinPts$  **then**

        Пометить  $x$  как потенциально шумовую точку;

**else**

$K = U_\varepsilon(x), a = a + 1;$

**for**  $x' \in K$  **do**

**if**  $|U_\varepsilon(x')| \geq MinPts$  **then**

$K = K \cup U_\varepsilon(x');$

**else**

                пометить  $x'$  как граничную точку кластера  $K;$

**foreach**  $x_i \in K$  **do**  $a_i = a;$

$U = U \setminus K;$

---

- Общая идея состоит в построении графика, по ординате у которого расстояние до *MinPts*-го соседа, а по абсциссе – точки, отсортированные в порядке увеличения этого расстояния.
- Существенный скачок в значении идентифицирует выбросы, посему задавая некоторый процент на их число можно определить  $\varepsilon$ .
- Обычно строят несколько таких графиков для различных значений *MinPts*.
- В некоторых источниках значение *MinPts* предлагают выбирать равным  $\dim X + 1$ , Где-то встречается  $2 * \dim X$

- Есть реализация в sklearn'е. Кроме того, там же представлена модификация алгоритма под названием OPTICS, фактически отличающаяся от него тем, что задает интервал для значений  $\varepsilon$ , что позволяет выделять кластеры с различными плотностями.
- Основные гиперпараметры:
  - `eps` и `min_samples`.  $\varepsilon$  и *MinPts* соответственно.
  - `algorithm`. Способ поиска соседей: брутфорс, ball-дерево, KD-дерево, и автоматический подбор подходящего с учетом обучающей выборки (дефолтное).
  - `leaf_size`. Максимальный размер листа в дереве, если выбрано ball/KD-дерево.
  - `metric` и  $p$ . Метрику можно как реализовать самостоятельно, так и использовать из имеющегося: Минковского ( $p$  – ее параметр) и ее частные случаи (Чебышева и Манхэттенская).