Data Mining - методы анализа и обработки данных

Лекция 2

Кластеризация данных.
Иерархические алгоритмы, K-Means, Fuzzy и DBSCAN

Институт математики, механики
и компьютерных наук им. И.И. Воровича

Что такое кластеризация?

Кластеризация — это задача разбиения множества объектов на группы (кластеры) так, чтобы объекты внутри одной группы были максимально похожи друг на друга, а объекты из разных групп — максимально различны.

- **Обучение без учителя:** У нас нет правильных ответов (меток классов).
- Алгоритм сам находит структуру в данных.
- Заранее неизвестно, сколько кластеров получится.

Классификация vs Кластеризация

Классификация

- Обучение **с учителем**.
- Классы заранее известны.
- Цель: предсказать метку.
- *Пример:* Спам / Не спам.

Кластеризация

- Обучение **без учителя**.
- Классы неизвестны.
- Цель: найти структуру.
- *Пример:* Сегментация клиентов.

Области применения

Маркетинг
Сегментация клиентов по поведению, доходу и активности для таргетированной рекламы.

Биология
Группировка генов с похожими функциями или классификация видов по морфологическим признакам.

Аномалии
Поиск мошеннических транзакций. Мошенничество часто не похоже ни на один кластер нормальных операций.

Проблемы кластеризации

Главная проблема: отсутствие "правильного" ответа. Мы не знаем, сколько кластеров должно быть и какие объекты действительно похожи.

- **Проклятие размерности:** в многомерном пространстве все точки становятся одинаково далекими друг от друга.
- **Масштаб признаков:** если один признак в рублях (100000), а другой в процентах (5), метрика расстояния будет учитывать только первый.

Подготовка данных: Масштабирование

Перед кластеризацией данные **обязательно** нужно стандартизировать (Z-score), чтобы все признаки имели равный вес.

До масштабирования
Признак 1: 100, 200, 150
Признак 2: 1.5, 2.0, 1.8
Расстояние определяется только Признаком 1.

После масштабирования
Оба признака приведены к среднему 0 и отклонению 1.
Все признаки вносят равный вклад.

Метрики расстояний

Кластеризация напрямую зависит от того, как мы считаем «похожесть» объектов.

Евклидово (L2): Прямолинейное расстояние.

$$d_2(x, y) = \sqrt{\sum (x_i - y_i)^2}$$

Для числовых данных.

Манхэттенское (L1): По сетке (как такси).

$$d_1(x, y) = \sum |x_i - y_i|$$

Устойчиво к выбросам.

Типы алгоритмов кластеризации

Иерархические
Строят дерево вложенных кластеров. Не нужно заранее задавать число кластеров. (Агломеративная).

Неиерархические (Плоские)
Разбивают данные на заданное число кластеров K. (K-Means, Fuzzy C-Means).

Плотностные
Ищут области с высокой концентрацией точек. (DBSCAN).

Вероятностные
Предполагают, что данные сгенерированы смесью распределений. (GMM).

Иерархическая кластеризация

Создает иерархию кластеров (дерево). Два основных подхода:

Агломеративная (Bottom-Up)
Каждый объект — отдельный кластер. Постепенно объединяем самые близкие, пока не получим один большой.

Дивизимная (Top-Down)
Один кластер со всеми объектами. Постепенно разбиваем его на более мелкие.

Агломеративный алгоритм

1. Вычислить матрицу расстояний между всеми объектами.
2. Найти два ближайших кластера и объединить их.
3. Пересчитать расстояния от нового кластера до остальных.
4. Повторять шаги 2 и 3, пока не останется один кластер.

Результат визуализируется с помощью **дендрограммы** (древовидной диаграммы).

Методы связи (Linkage)

Как считать расстояние между двумя *кластерами*?

Single Linkage
Расстояние между *ближайшими* точками.
(Склонен к эффекту цепочки).

Complete Linkage
Расстояние между *самыми дальними* точками.
(Создает компактные кластеры).

- **Average Linkage:** Среднее расстояние между всеми парами точек.
- **Ward's Method:** Минимизирует дисперсию внутри кластеров (самый популярный).

Метод K-средних (K-Means)

Самый популярный алгоритм плоской кластеризации.

Плюсы

- Очень быстрый: $O(N \cdot K \cdot I)$.
- Легко интерпретировать.
- Хорошо работает на больших данных.

Минусы

- Нужно задавать K.
- Чувствителен к выбросам.
- Только сферические кластеры.
- Результат зависит от инициализации.

Целевая функция (Inertia)

K-Means минимизирует сумму квадратов расстояний от точек до их центроидов (WCSS - Within-Cluster Sum of Squares).

$$J = \sum_{k=1}^{K} \sum_{x \in C_k} ||x - \mu_k||^2$$

- **K** — число кластеров.
- $\mathbf{C}_k$ — множество точек в кластере k.
- μ_k — центроид кластера k.

Алгоритм K-Means

1. Выбрать **K** начальных центроидов (случайно или K-Means++).
2. **Назначение:** Каждый объект относится к тому кластеру, центроид которого ближе всего.
3. **Обновление:** Пересчитать центроид каждого кластера как среднее арифметическое всех его точек.
4. Повторять шаги 2 и 3, пока центроиды не перестанут двигаться (сходимость).

Проблема инициализации

Проблема: K-Means может застрять в локальном минимуме, если центроиды выбраны неудачно.

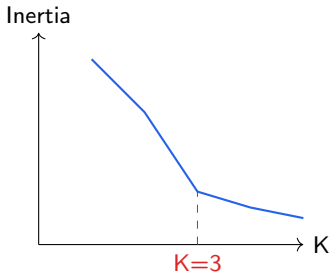
- **K-Means++:** Улучшенный алгоритм инициализации. Первый центроид выбирается случайно, а каждый следующий — с вероятностью, пропорциональной квадрату расстояния до ближайшего уже выбранного центроида.
- В `scikit-learn` параметр `init='k-means++'` используется по умолчанию.

Как выбрать K? Метод локтя (Elbow)

Строим график зависимости Inertia (WCSS) от числа кластеров K.

- При увеличении K ошибка всегда падает.
- После определенного значения K график образует «локоть» (излом).
- Это значение K считается оптимальным.

Дальнейшее увеличение K не дает значительного улучшения, но усложняет модель.



Нечёткая кластеризация (Fuzzy C-Means)

Точка может принадлежать **всем** кластерам с разной степенью уверенности.

Жесткая (K-Means)
Точка X принадлежит Кластеру 1 (100%) и Кластеру 2 (0%).

Мягкая (Fuzzy)
Точка X принадлежит Кластеру 1 (80%) и Кластеру 2 (20%).

Fuzzy C-Means: Матрица принадлежности

Вводится матрица принадлежности **U**, где элемент u_{ij} — степень принадлежности точки i кластеру j.

- Значения от 0 до 1.
- Сумма вероятностей для каждой точки равна 1: $\sum_j u_{ij} = 1$.
- Минимизируется функция с параметром нечеткости m (обычно $m = 2$).

$$J = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^m ||x_i - \mu_j||^2$$

Плотностная кластеризация (DBSCAN)

Алгоритм, основанный на плотности. Находит кластеры сложной формы и **выбросы**.

Два параметра:

- **eps (ε):** Радиус соседства (максимальное расстояние между двумя точками, чтобы считаться соседями).
- **min_samples:** Минимальное число точек в радиусе ε, чтобы точка считалась «ядром» кластера.

DBSCAN: Типы точек

Ядро (Core)
В радиусе ε находится не менее `min_samples` точек.

Граница (Border)
В радиусе ε меньше `min_samples` точек, но она находится в радиусе точки-ядра.

Шум (Noise)
Не является ни ядром, ни границей. Выброс.

DBSCAN: Алгоритм

1. Выбрать случайную необработанную точку.
2. Если она является ядром, создается новый кластер. Все точки в радиусе ε добавляются в кластер.
3. Если среди добавленных точек есть ядра, их соседи также добавляются (расширение кластера).
4. Если точка не является ядром (шум), она помечается как выброс.
5. Повторять, пока все точки не будут обработаны.

DBSCAN: Плюсы и минусы

Плюсы

- Не нужно задавать число кластеров K.
- Находит кластеры произвольной формы.
- Устойчив к выбросам (находит их).

Минусы

- Плохо работает на данных с разной плотностью.
- Чувствителен к выбору параметров ε и `min_samples`.
- Не работает в многомерном пространстве.

Кластеризация категориальных данных

Все рассмотренные алгоритмы (K-Means, DBSCAN, иерархическая) используют **метрики расстояний**, которые требуют числовых данных.

- **Проблема:** Как найти расстояние между "Москва" и "Питер"?
- **Решение 1:** One-Hot Encoding (превратить в 0 и 1 и использовать K-Means).
- **Решение 2:** Использовать специальные алгоритмы: **K-Modes** (модификация K-Means) или **K-Prototypes**.
- В `scikit-learn` для этого используют `KModes` из библиотеки `kmodes`.

Пример: K-Means на Python

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler

X, _ = make_blobs(n_samples=300, centers=3, cluster_std=0.60, random

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_scaled)

labels = kmeans.labels_

plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=labels, cmap='viridis'
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:
s=200, marker='X', c='red', label='Centroids')
plt.legend()
```

Пример: DBSCAN на Python

```
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt

# 1. Train DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan.fit(X_scaled)

# 2. Get labels (-1 means noise)
labels = dbscan.labels_

# 3. Visualize
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=labels, cmap='viridis'
plt.title("DBSCAN Clustering")
plt.show()
```

Сравнение алгоритмов (Шпаргалка)

Свойство	**K-Means**	**Иерарх.**	**DBSCAN**	**Fuzzy**
Нужно задавать K	Да	Нет	Нет	Да
Форма кластеров	Сферы	Любая	Любая	Сферы
Выбросы	Нет	Нет	Да	Нет
Скорость	Высокая	Низкая	Средняя	Высокая
Мягкое разбиение	Нет	Нет	Нет	Да

Оценка качества кластеризации

Внутренние метрики
Оценивают компактность и изолированность кластеров.
Пример: Silhouette Score.

Внешние метрики
Сравнивают с известными "истинными" метками.
Пример: Adjusted Rand Index (ARI).

Итоги лекции

- **Кластеризация** — разбиение данных на группы без учителя.
- Перед кластеризацией данные **обязательно** масштабируются.
- **Иерархические методы** строят дендрограмму (хорошо для малых данных).
- **K-Means** — быстрый, но требует задания K и ищет сферические кластеры.
- **Fuzzy C-Means** — позволяет точке принадлежать нескольким кластерам.
- **DBSCAN** — находит кластеры произвольной формы и выбросы.
- Для категориальных данных используют **K-Modes**.