Введение в машинное обучение

Кластеризация

Дьяконов А.Г.

Московский государственный университет имени М.В. Ломоносова (Москва, Россия)

Кластеризация

Кластеризация

– разбиение множества объектов на группы похожих

неформально:
маленькие внутрикластерные расстояния
большие межкластерные расстояния

Самое важное и «скользкое»

дальше предполагаем, что есть некоторая адекватная метрика!
С помощью её и будем осуществлять кластеризацию.

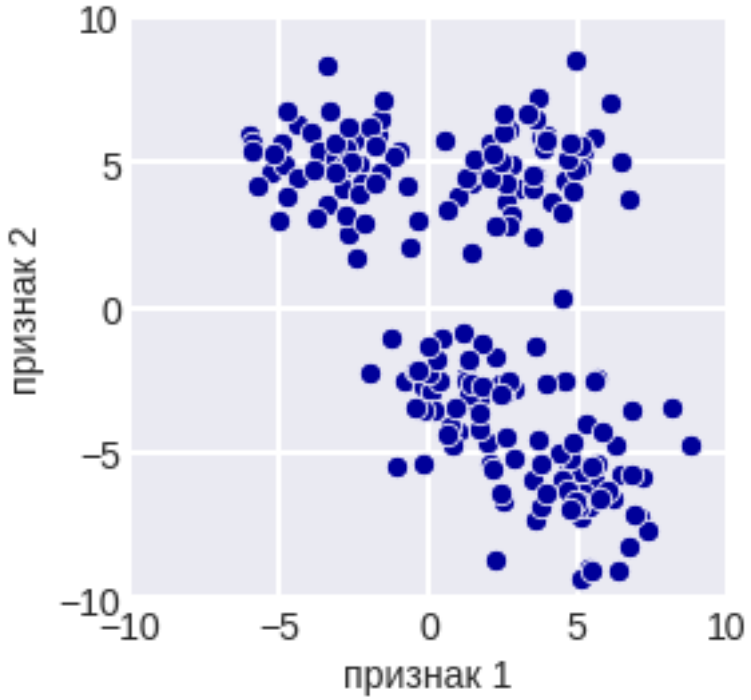
Примеры

пользователи со схожим поведением

Входная информация для алгоритмов

1. (Feature-based) Признаковые описания объектов
2. (Dis/similarity-based) Попарные сходства/различия

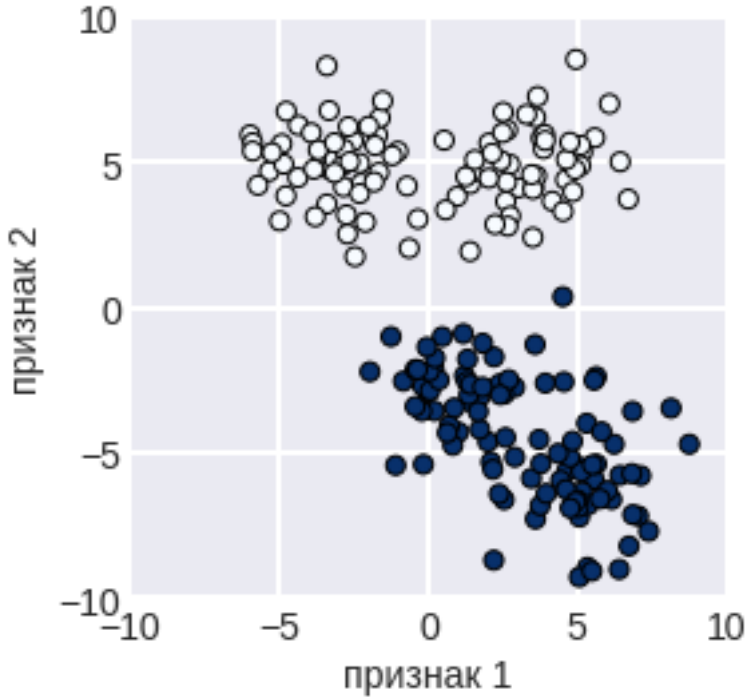
Модельная задача кластеризации



Даны объекты (без меток)

Надо разбить на группы похожих – кластеры

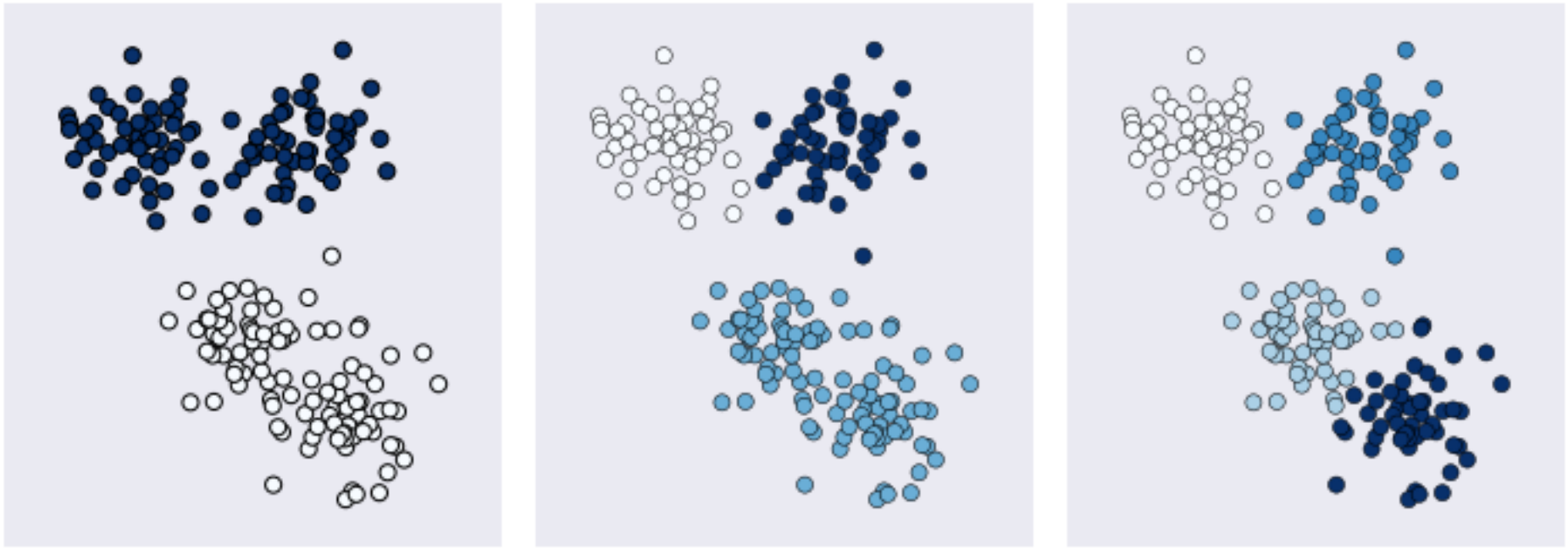
Модельная задача кластеризации



Даны объекты (без меток)

Надо разбить на группы похожих – кластеры

Модельная задача кластеризации



Много допустимых решений…

Зачем

- **кластеризация клиентов**

(выработка таргетированной политики)

- **сжатие данных**

(при модерации проверять несколько представителей кластера, устранение однотипности вопросов)

- **сообщества клиентов**

(эффективное распространение новостей, предложение услуг)

- **анализ данных / признаков**

(классическая идея математики – факторизация)

- **важная составляющая решения других задач**

(semi-supervised, outlier detection, community detection)

k-средних

$$\sum_t \frac{1}{|C_t|} \sum_{x_i, x_j \in C_t} \rho(x_i, x_j) \to \min_{\{C_t\}},$$

– NP-полная задача

вместо этого

$$\sum_t \frac{1}{|C_t|} \sum_{x_i \in C_t} \rho(x_i, \mu_t) \to \min_{\{C_t\},\{\mu_t\}}$$

Если зафиксировать $\{C_t\}$, то оптимальное решение

$$\mu_t = \sum_{x_i \in C_t} x_i$$

(центроид кластера)

Если зафиксировать $\{\mu_t\}$, то оптимальное решение

$$C_t = \{i \,|\, \| x_i - \mu_t \| = \min_j \| x_i - \mu_j \|\}$$

Такую минимизацию делают итеративно

k-средних

Вход: $\{x_1, \ldots, x_m\} \subseteq \mathbf{R}^n$

Инициализация: k **центров кластеров** $\{\mu_1, \ldots, \mu_k\} \subseteq \mathbf{R}^n$

(случайные точки или случайные объекты из обучающей выборки)

Итерация:

1. (assignment) Каждый объект приписать к тому кластеру, к центру которого он ближе:

$$C_t = \{i \mid k = \arg\min_t \| x_i - \mu_t \|^2\}$$

2. (update) Пересчитать центры кластеров:

$$\mu_t = \frac{1}{|C_t|} \sum_{i \in C_t} x_i$$

при неопределённости вида «деление на ноль» оставляем центр неизменным

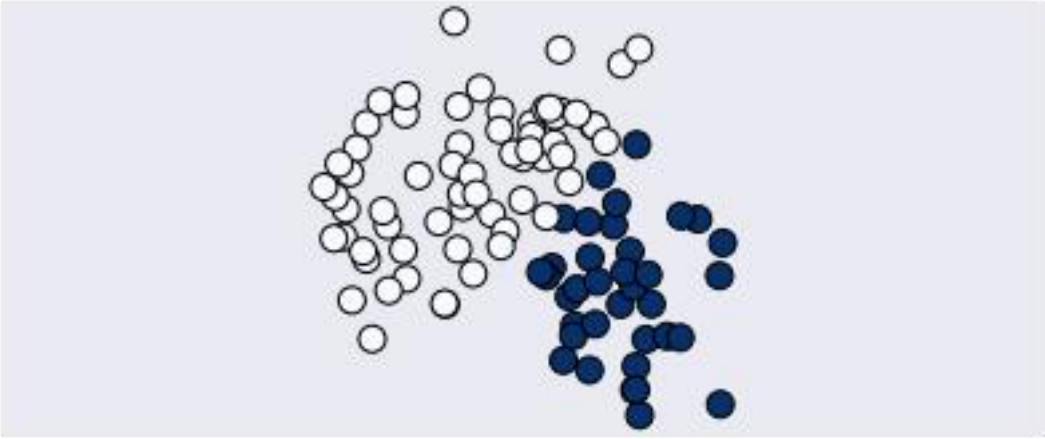
Повторять итерации до сходимости (пока центры станут неподвижными)

Ничьи разрешаются произвольно.

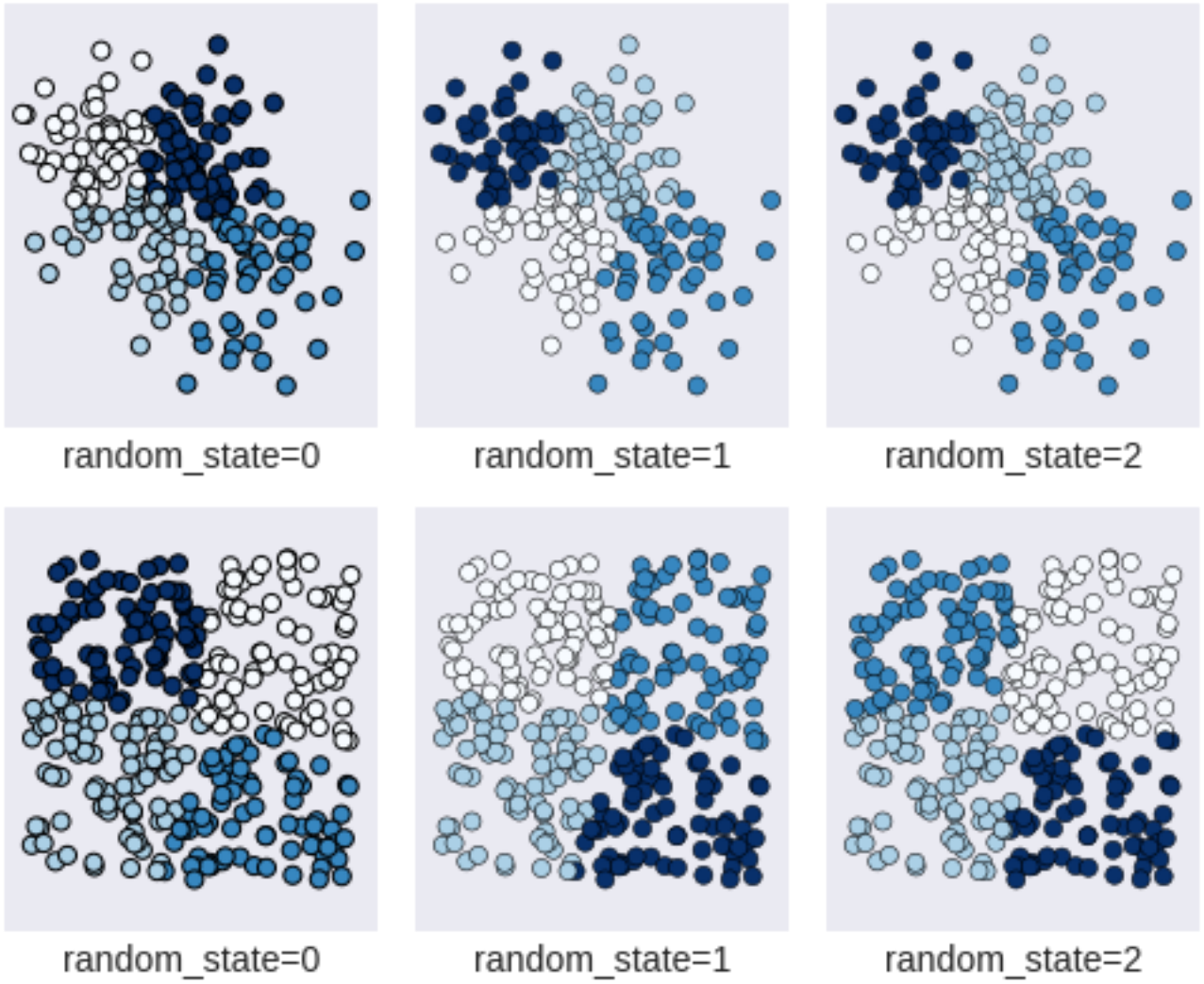
Центры изначально лучше выбирать случайными точками выборки

Код

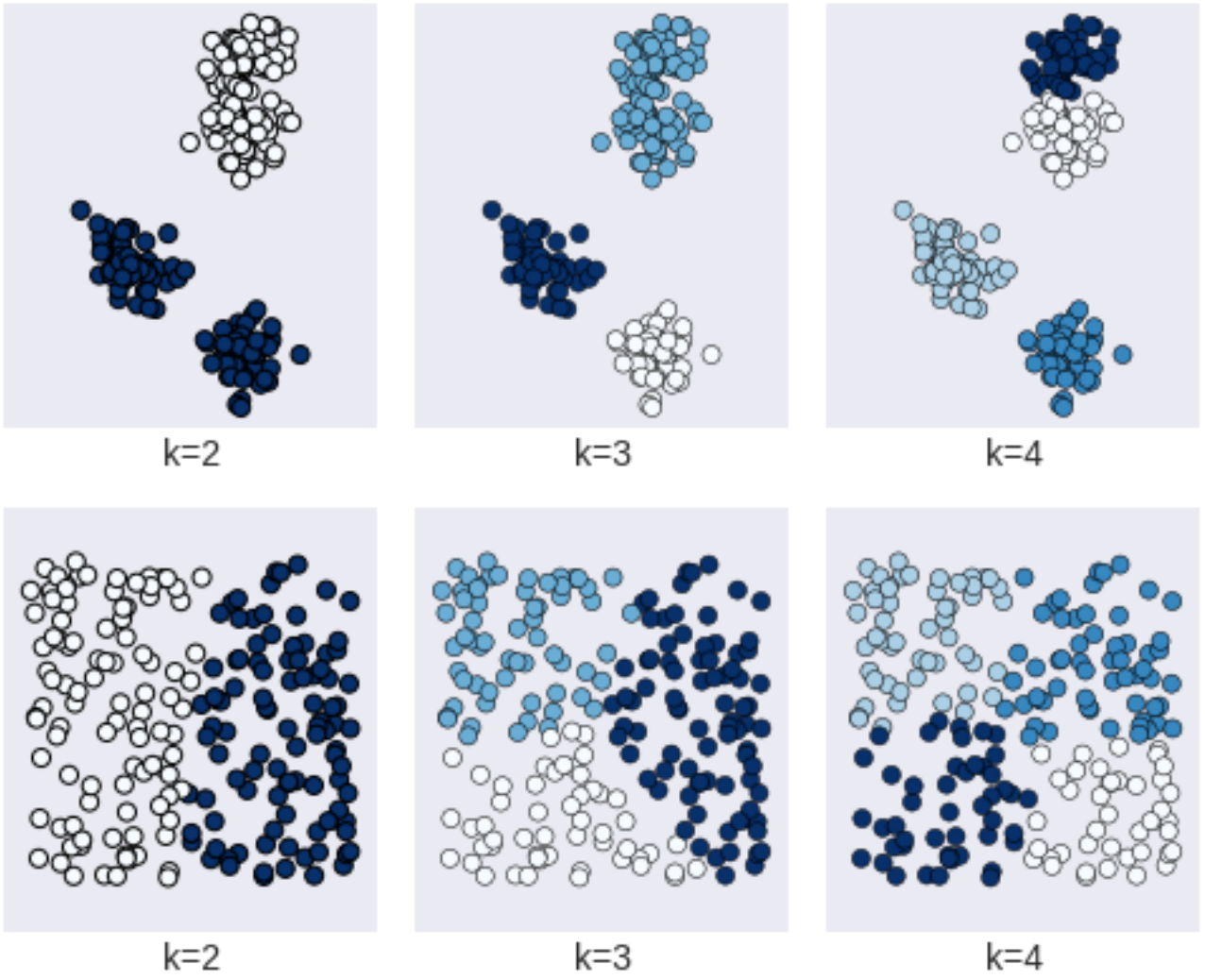
```
# создание датасета
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=100, n_features=2,
                  centers=3, random_state=0)

# запуск алгоритма
from sklearn.cluster import KMeans
a = KMeans(n_clusters=2, random_state=0).fit_predict(X)

# визуализация
import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=a)
```



Разная начальная инициализация



Заранее задаётся число кластеров k



Свойства k-means

– теоретически может сходиться долго, но на практике быстро

– чувствителен к начальной инициализации

(могут получаться разные ответы):

- качество кластеризации
- скорость сходимости

несколько разных инициализаций

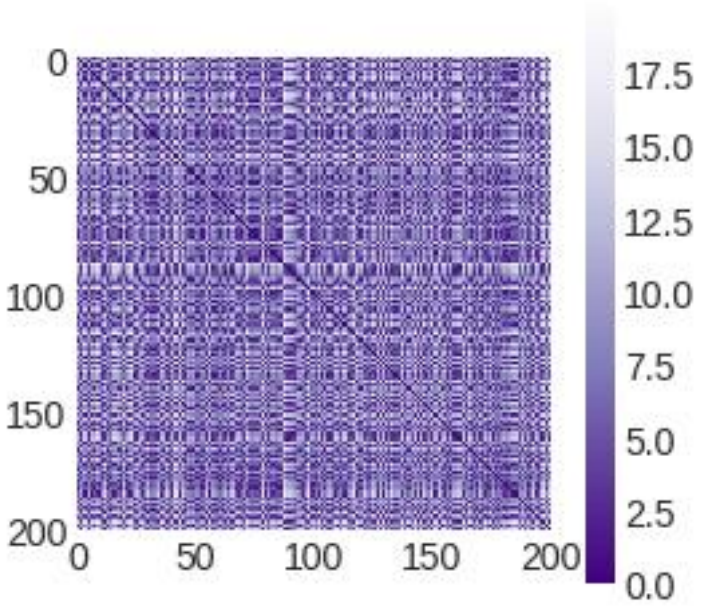
эвристика: 1й центр – один из объектов, 2й – максимально удалённый от первого, 3й – максимально удалённый от предыдущих центров и т.д.

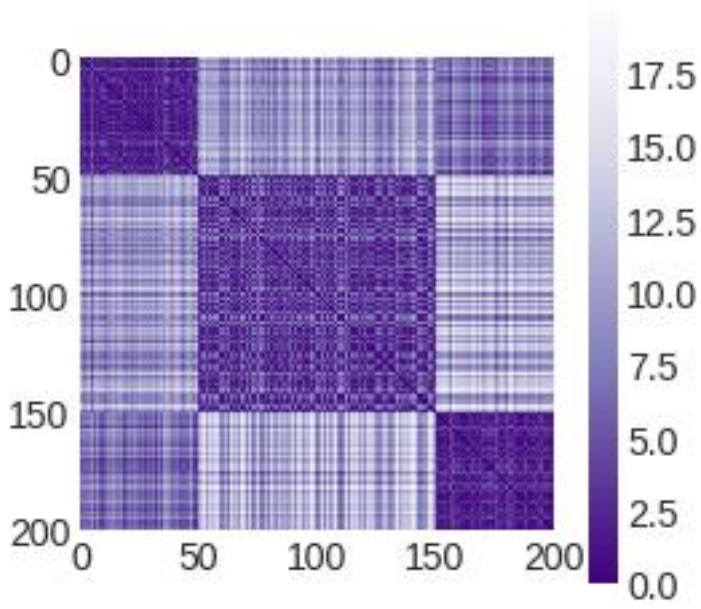
см. также k-means++

– хорошо работает для примерно равномощных кластеров с «шаровой формой»

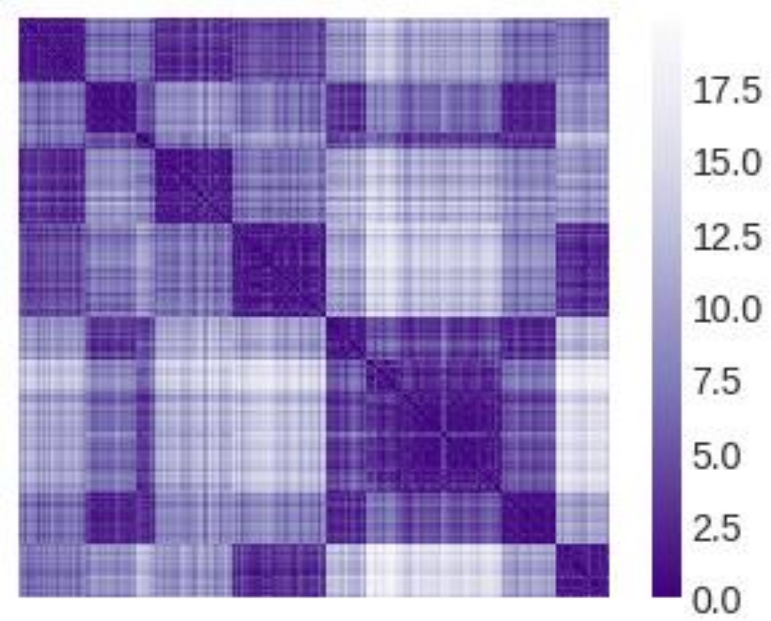
Иллюстрация

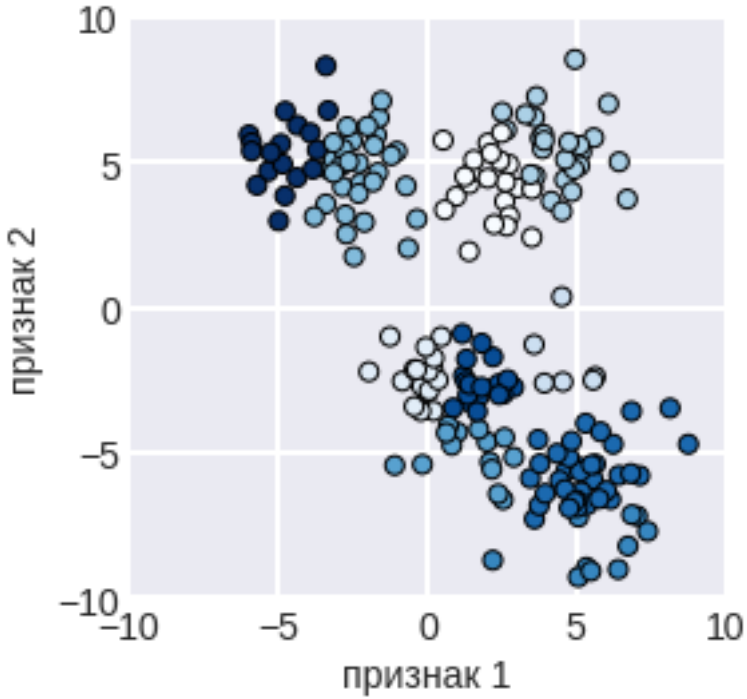
Матрица попарных расстояний до и после кластеризации (объекты упорядочены по кластерам)





Иллюстрация





ДЗ Как логично упорядочить кластеры?

ДЗ

Реализовать k-means,
используя numpy + matplotlib

1. Исследовать зависимость от стратегии начальной инициализации
2. Исследовать, для каких задач подходит / не подходит
3. Предложить и исследовать стратегию выбора числа кластеров
4. Исследовать зависимость (скорости настройки) от объёма данных / сложности задачи
5. Предложить эвристику для визуализации матрицы попарных расстояний

При реализации помнить о векторизации