Кластеризация

Лекция 9

Задачи машинного обучения

- Обучение с учителем (supervised learning)
- Обучение со слабым контролем (weakly supervised learning)
- Обучение с подкреплением (reinforcement learning)
- Обучение без учителя (unsupervised learning)

Задачи машинного обучения

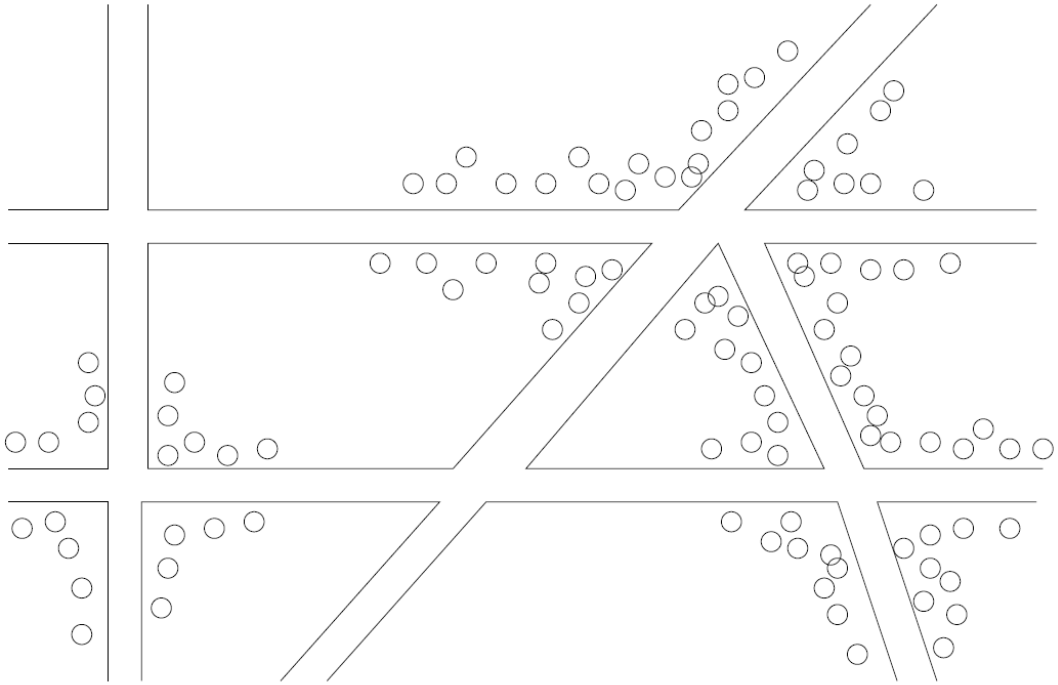
- Обучение с учителем (supervised learning)
- Обучение со слабым контролем (weakly supervised learning)
- Обучение с подкреплением (reinforcement learning)
- **Обучение без учителя (unsupervised learning)**
  - Кластеризация (clustering)
  - Снижение размерности (dimension reduction)
  - Визуализация (visualization)
  - Обучение представлений (representation learning)

Задача кластеризации

- *Кластеризация* (кластерный анализ, cluster analysis, clustering) – разделение исследуемого множества объектов на группы «похожих» объектов, называемых *кластерами* (clusters)
- Множество классов заранее неизвестно
- Применение:
  - анализ текстов, изображений, видео, аудио
  - маркетинг
  - социология
  - генетика
  - …

Задача кластеризации

- Leskovec et al. Mining of Massive Datasets (2019), p. 4
  - https://en.wikipedia.org/wiki/1854_Broad_Street_cholera_outbreak



Задача кластеризации

- Пусть дана выборка объектов:

$$X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_l\}, \qquad \boldsymbol{x}_i = (x_{i1}, \dots, x_{id}) \in \mathbb{X}$$

- Требуется выявить в данных K кластеров – таких областей, что объекты внутри одного кластера похожи друг на друга, а объекты из разных кластеров друг на друга не похожи
- Формально: требуется построить алгоритм

$$a: \mathbb{X} \to \{1, \dots, K\}$$

- Количество кластеров K может быть задано или неизвестно

Функции расстояния

- Степень различия между объектами определяется на основе *функции расстояния* (метрики, metric, distance function)
- Функция расстояния ρ должна удовлетворять трем условиям (аксиомам):

  1) $\rho(x, y) = 0 \Leftrightarrow x = y$ (аксиома тождества)

  2) $\rho(x, y) = \rho(y, x)$ (аксиома симметрии)

  3) $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$ (аксиома треугольника)

  Следствие: $\rho(x, y) \geq 0$ (неотрицательность расстояния)

Функции расстояния

- Евклидово расстояние:

$$\rho(x, y) = \sqrt{\sum_{j=1}^{d} \left(x_j - y_j\right)^2}$$

- Квадрат евклидова расстояния:

$$\rho(x, y) = \sum_{j=1}^{d} \left(x_j - y_j\right)^2$$

– не является расстоянием (не выполняется аксиома треугольника)

Функции расстояния

- Манхэттенское расстояние (расстояние городских кварталов):

$$\rho(x, y) = \sum_{j=1}^{d} |x_j - y_j|$$

- Расстояние Чебышёва:

$$\rho(x, y) = \max_{j=1,\ldots,d} |x_j - y_j|$$

Функции расстояния

- Расстояние Минковского (обобщенная функция расстояния):

$$\rho(x, y) = \left(\sum_{j=1}^{d} |x_j - y_j|^p \right)^{1/p}$$

- является метрикой для $p \geq 1$
- при $p = 1$ становится манхэттенским расстоянием
- при $p = 2$ становится евклидовым расстоянием
- при $p = \infty$ становится расстоянием Чебышёва

Функции расстояния

- Косинусное сходство (cosine similarity):

$$sim(\vec{x}, \vec{y}) = \cos\theta = \frac{(\vec{x}, \vec{y})}{\|\vec{x}\|\|\vec{y}\|} = \frac{\sum_{j=1}^{d} x_j y_j}{\sqrt{\sum_{j=1}^{d} x_j^2}\sqrt{\sum_{j=1}^{d} y_j^2}}$$

– косинусное расстояние: $1 - sim(\vec{x}, \vec{y})$

  – не является метрикой (не выполняются 2 аксиомы)

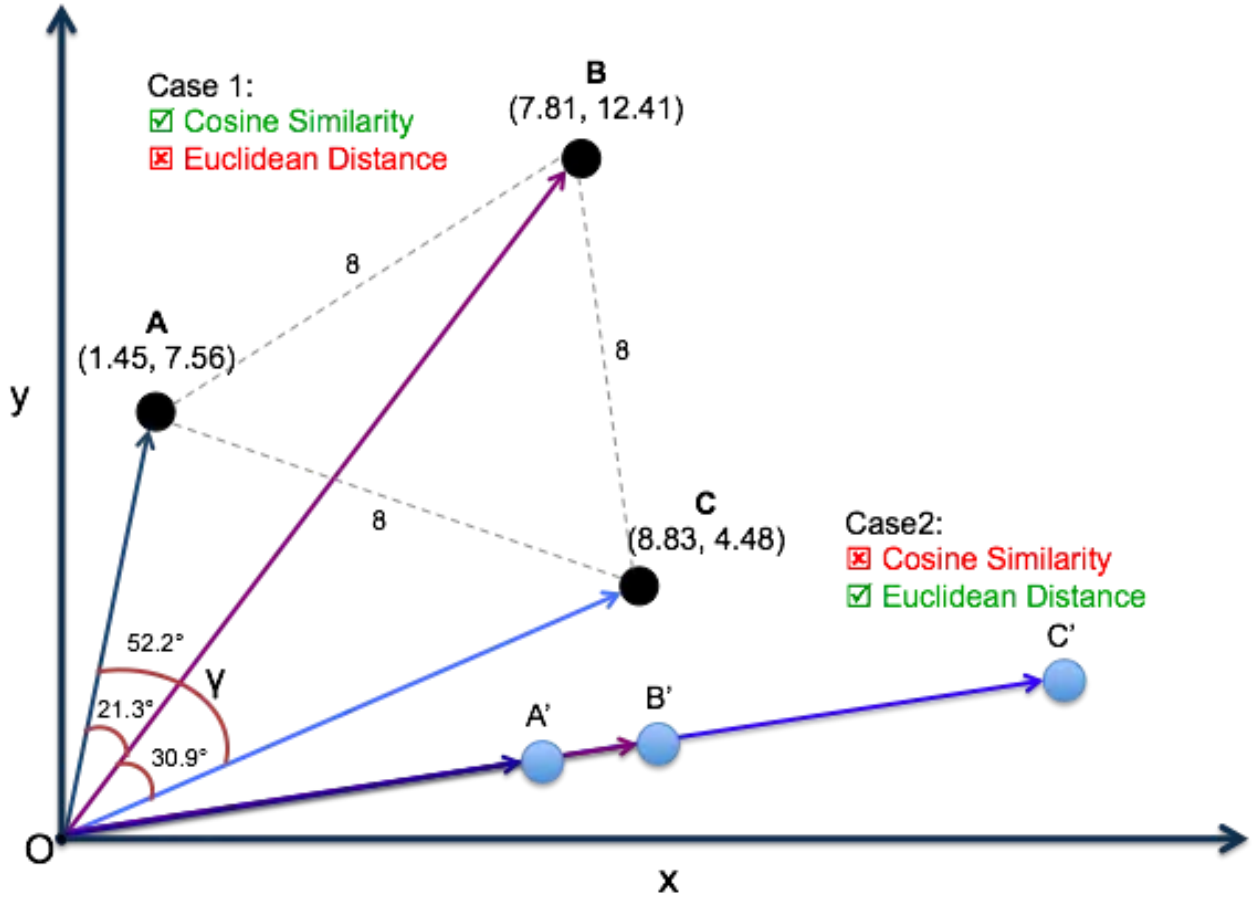
– $sim(\vec{x}, \vec{y}) \in [-1,1]$

– $sim(\vec{x}, \vec{y}) \in [0,1]$ при неотрицательных векторах

– используется при анализе текстов

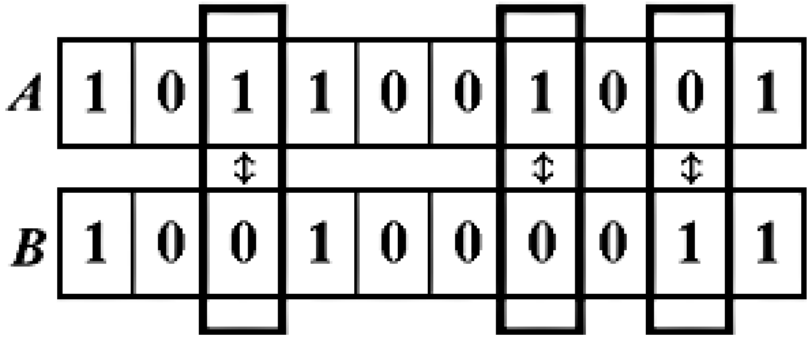
Функции расстояния

- Косинусное сходство vs. Евклидово расстояние



Функции расстояния

- Расстояние Хэмминга – количество позиций, в которых различаются символы в двух строках одинаковой длины



Расстояние Хэмминга = 3

  - используется при анализе строк и качественных признаков

Критерии качества кластеризации

Критерии качества:

- внутренние – основаны на свойствах выборки и кластеров
- внешние – используют информацию об истинных кластерах

Критерии качества кластеризации

- Внутрикластерное расстояние (компактность кластеров) (минимизировать):

$$\sum_{k=1}^{K}\sum_{i=1}^{l}[a(x_i)=k]\rho(x_i,c_k),$$

где c_k – центр кластера k (центроид):

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

Критерии качества кластеризации

- Межкластерное расстояние (отделимость кластеров) (максимизировать):

$$\sum_{i,j=1}^{l} \left[a(x_i) \neq a\left(x_j\right)\right] \rho\left(x_i, x_j\right)$$

Критерии качества кластеризации

- Индекс Данна (Dunn Index) (максимизировать):

$$\frac{\min\limits_{1\le k<k\prime\le K}\rho(k,k')}{\max\limits_{1\le k\le K}\rho(k)},$$

где $\rho(k,k')$ – расстояние между кластерами k и k',

$\rho(k)$ – внутрикластерное расстояние для k-го кластера

Критерии качества кластеризации

- Silhouette Coefficient
- Для одного объекта:

$$s = \frac{b - a}{\max(a, b)},$$

где a – среднее расстояние между объектом и всеми другими объектами в том же кластере,

b – среднее расстояние между объектом и всеми другими объектами в ближайшем кластере

https://scikit-learn.org/stable/modules/clustering.html#silhouette-coefficient

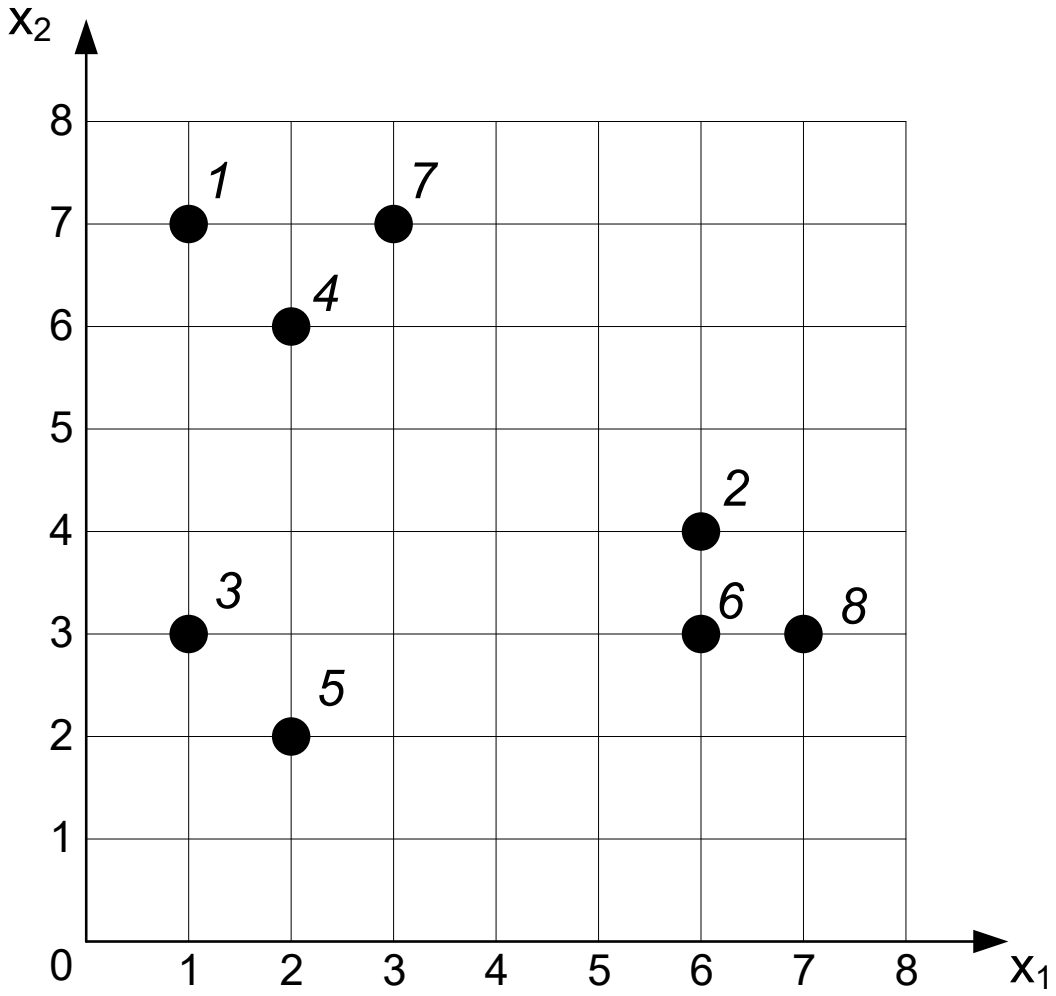
Алгоритмы кластеризации

- Алгоритм k-средних (k-means)
- Алгоритм DBSCAN
- Иерархические алгоритмы
- Графовые алгоритмы (спектральная кластеризация, минимальное остовное дерево)
- Вероятностные алгоритмы (EM-алгоритм)
- Сдвиг среднего значения (mean shift)
- Распространение похожести (affinity propagation)

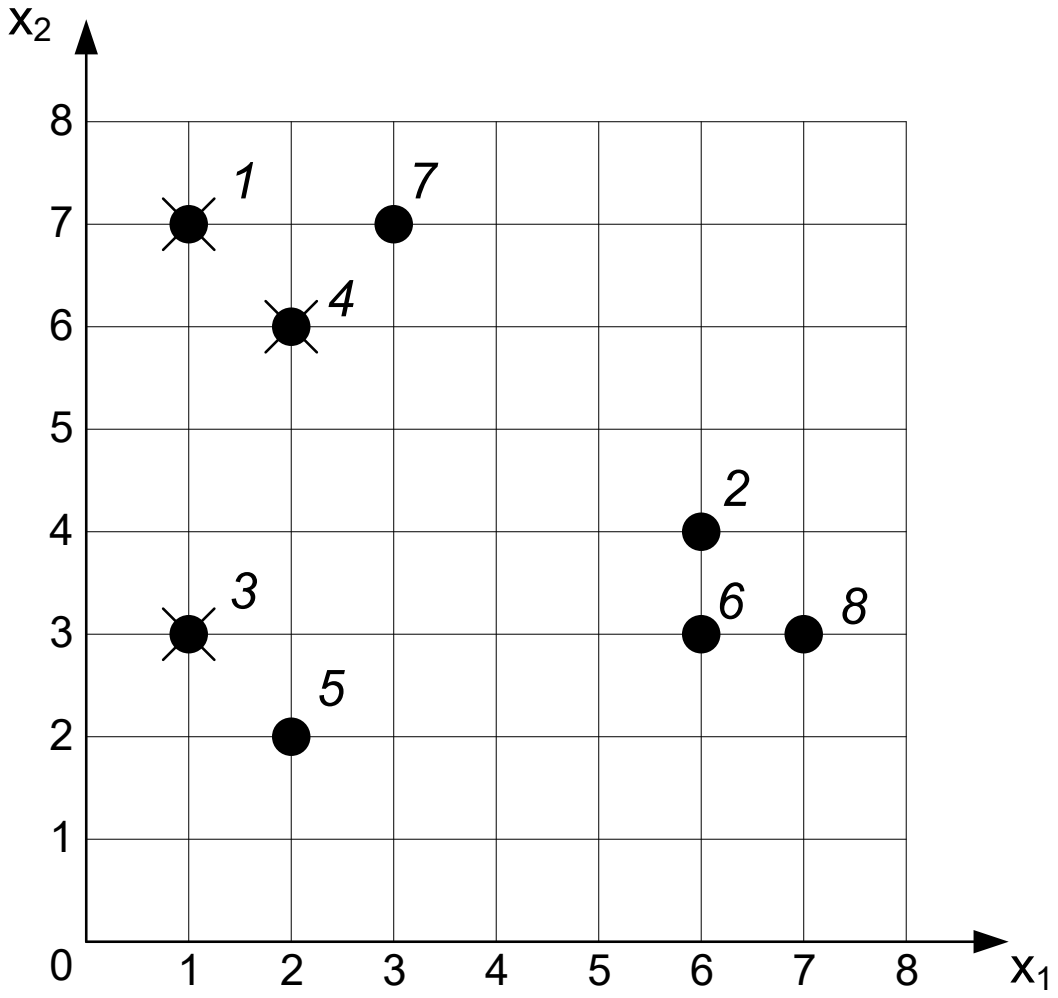
Алгоритм k-means

- Алгоритм k-means минимизирует внутрикластерное расстояние, в котором используется квадрат евклидовой метрики (т. н. “inertia” – инерция)
- Количество кластеров K должно быть задано заранее
- Алгоритм:
  1. Выбираются K начальных центров кластеров
  2. Объекты распределяются по кластерам
  3. Центры кластеров пересчитываются
  4. Шаги 2-3 повторяются до сходимости

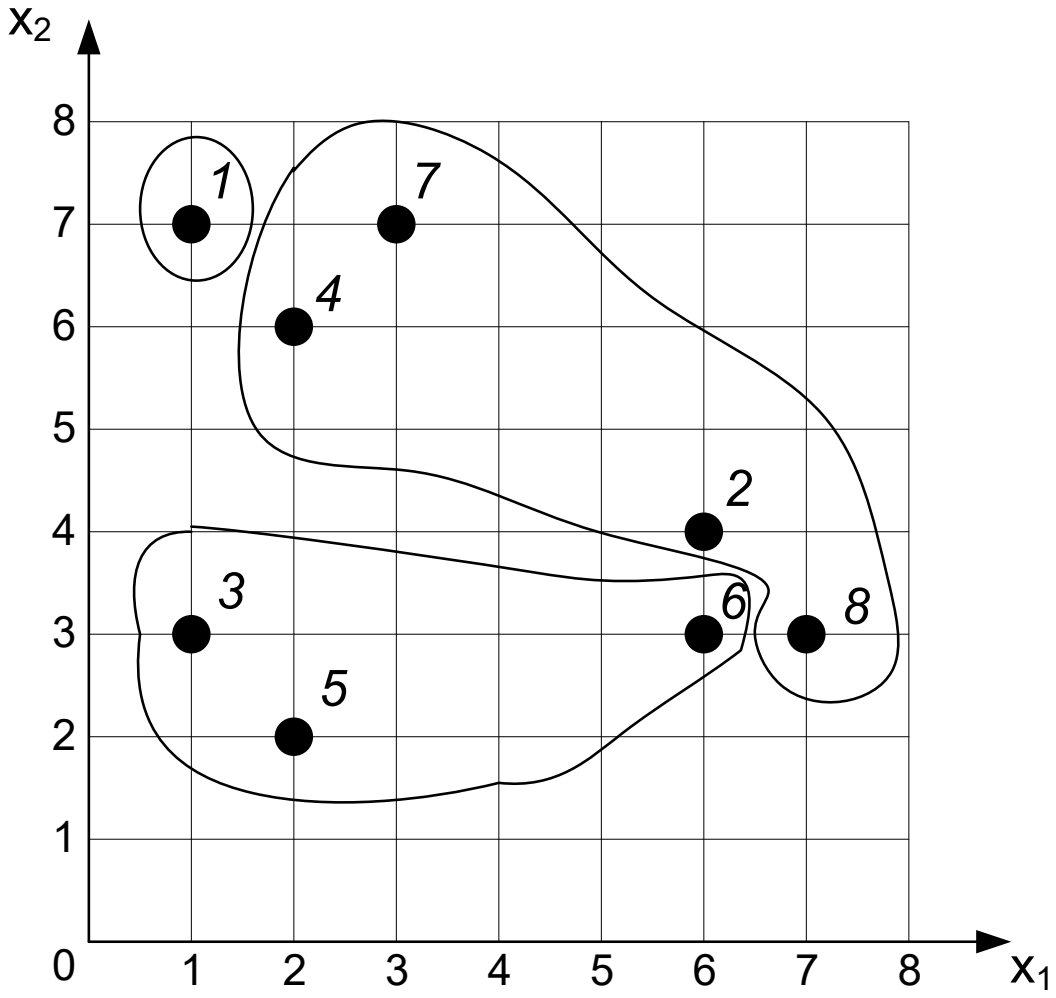
Алгоритм k-means



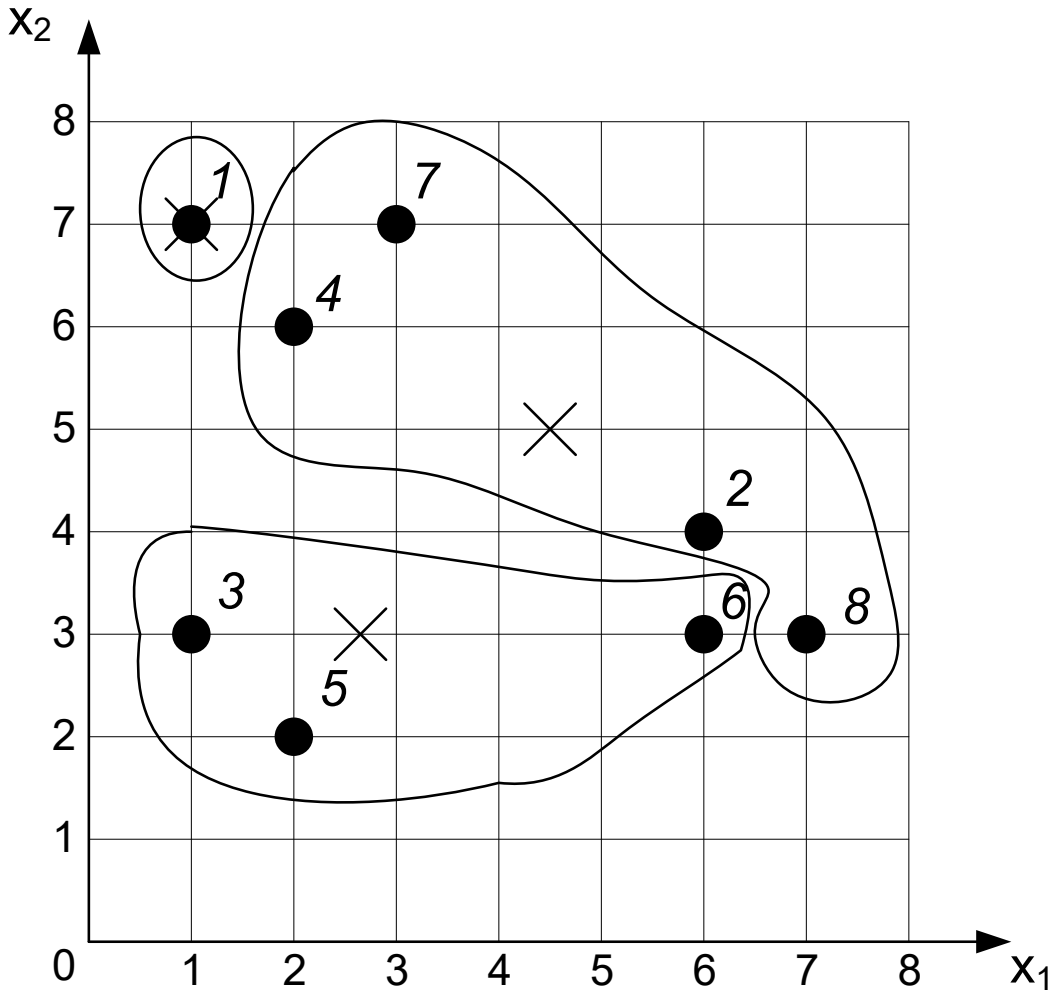
Алгоритм k-means



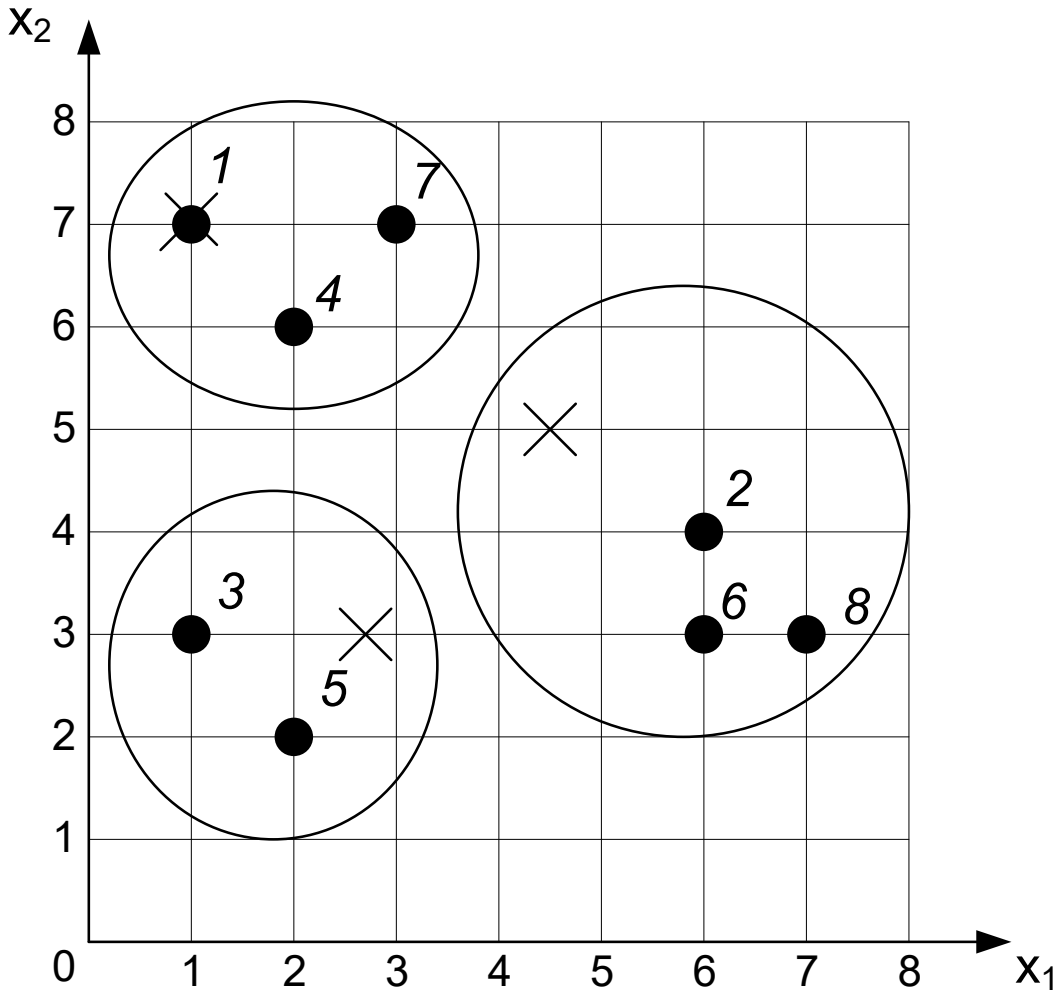
Алгоритм k-means



Алгоритм k-means



Алгоритм k-means



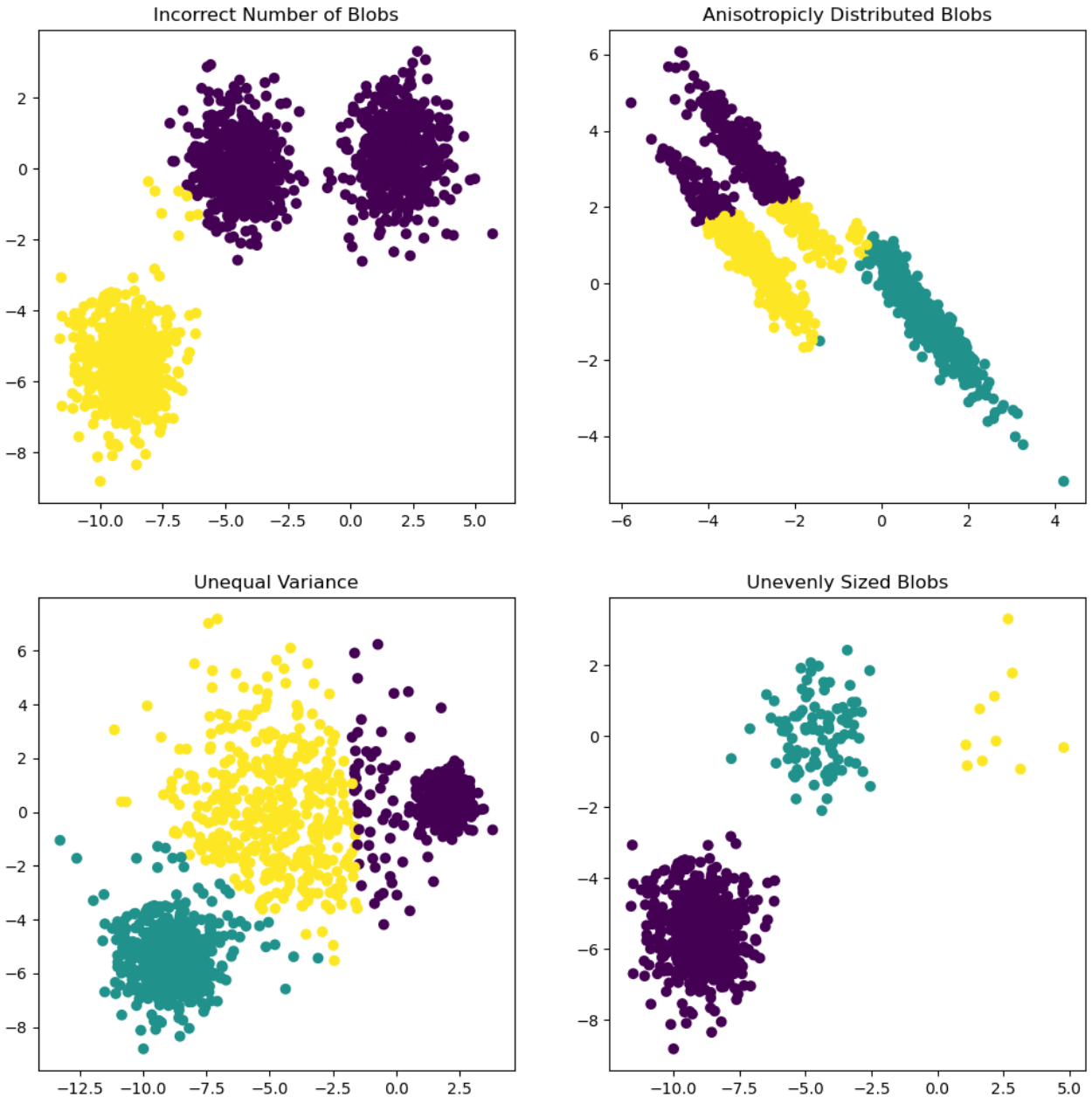
Алгоритм k-means

- Mini Batch k-means – каждый шаг осуществляется со случайно выбранным подмножеством объектов
- K-means всегда сходится (за достаточное количество шагов), но не гарантируя глобальный минимум
  - Точка минимума сильно зависит от выбора начальных центров
- Поэтому алгоритм часто выполняется несколько раз с разными начальными центрами и возвращается вариант с наименьшим значением внутрикластерного расстояния

Алгоритм k-means

- Выбор начальных центров:
  - случайные объекты
  - K-means++:
    - первый центроид выбирается случайным образом из объектов
    - для каждого объекта вычисляется квадрат расстояния до ближайшего центроида
    - следующий центроид выбирается с вероятностью, пропорциональной вычисленному квадрату расстояния

Алгоритм k-means – проблемы



Алгоритм k-means

- Визуализация:
  - https://www.naftaliharris.com/blog/visualizing-k-means-clustering

Алгоритм k-means

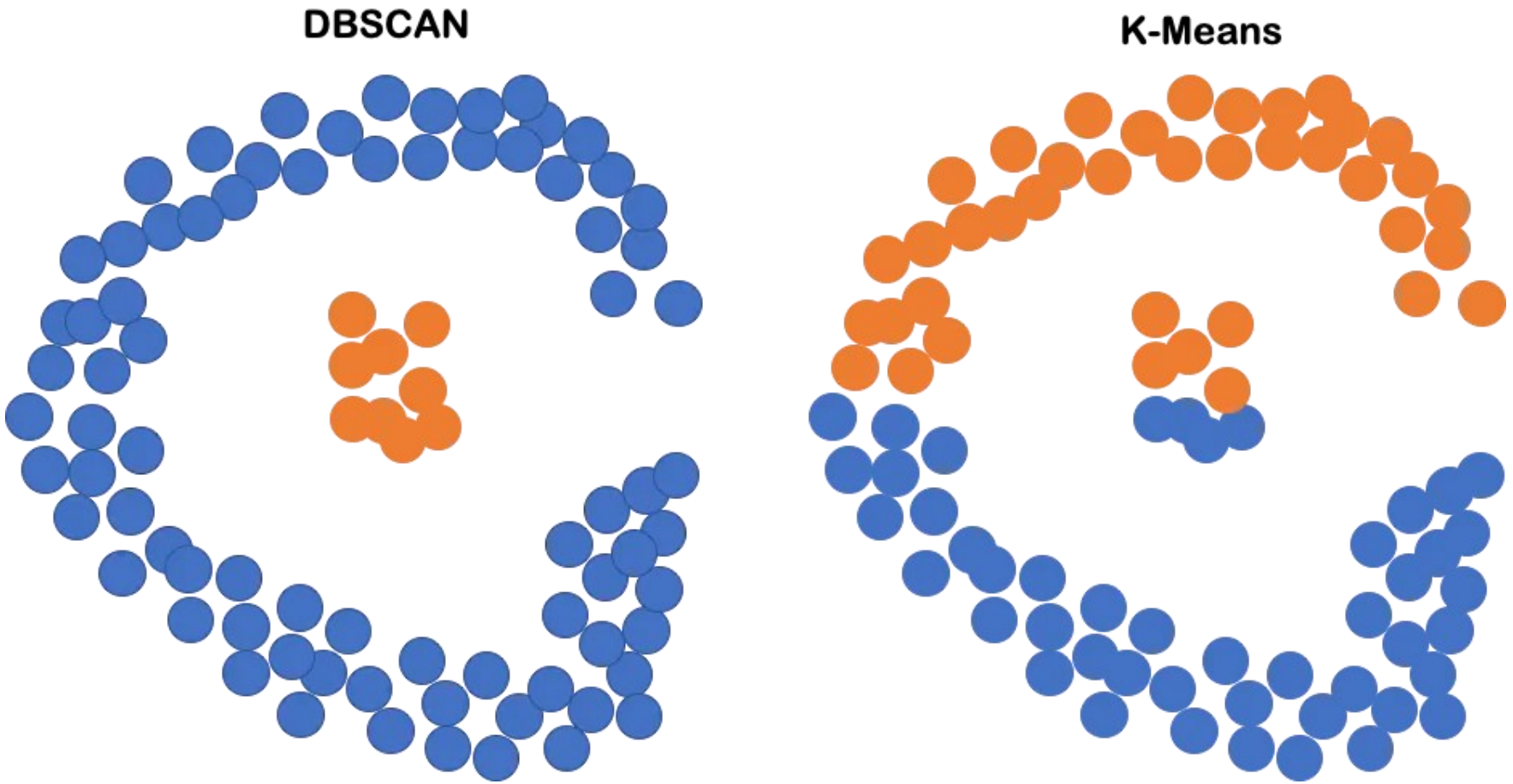
```
class sklearn.cluster.KMeans(
     n_clusters=8,
     *,
     init='k-means++',    # 'random', array, callable
     n_init=10,           # number of runs
     max_iter=300,
     tol=0.0001,
     verbose=0,
     random_state=None,
     copy_x=True,         # X -= X_mean
     algorithm='lloyd'    # 'elkan'
)
```

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

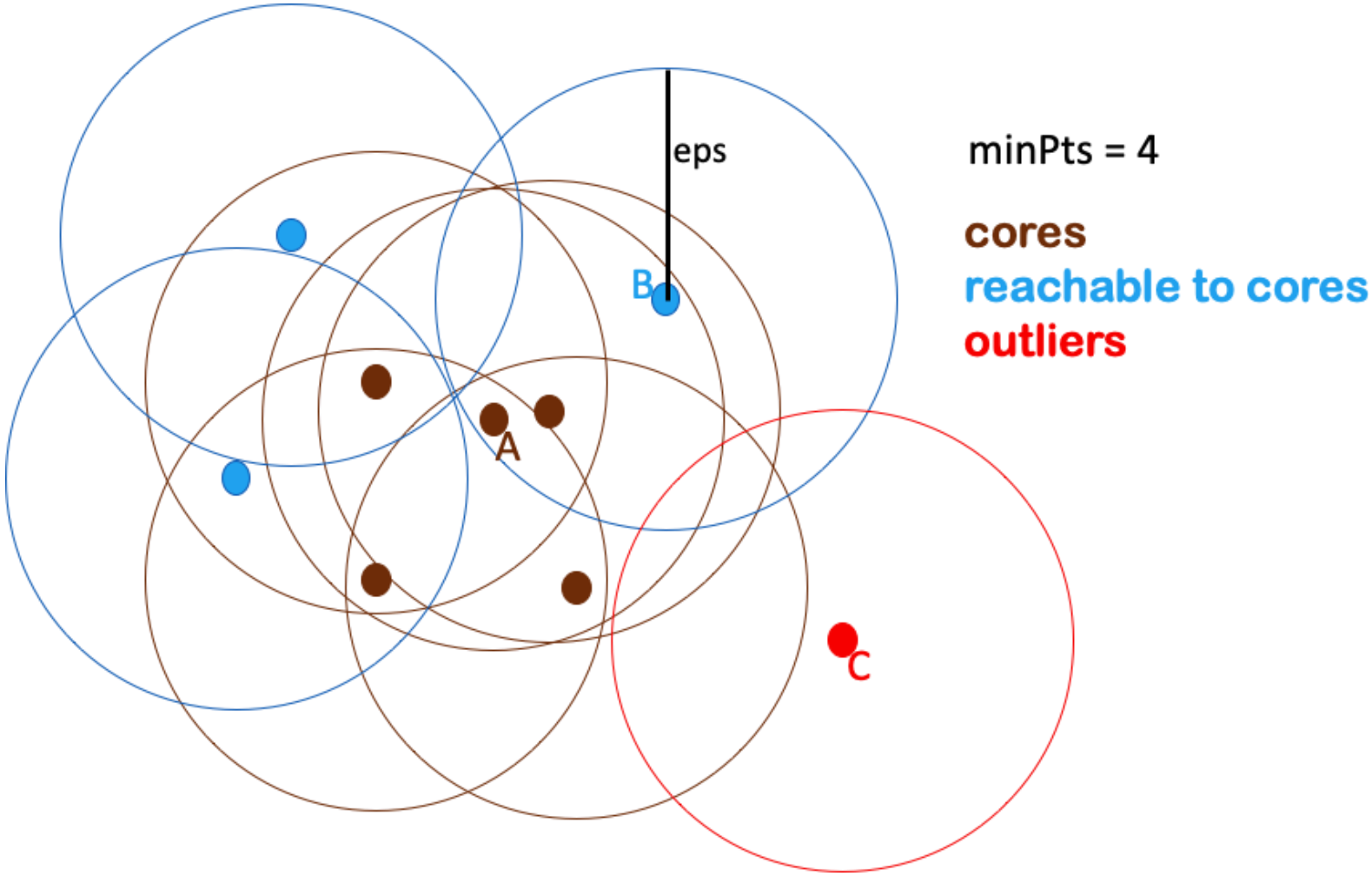
Алгоритм DBSCAN

- Основанная на плотности пространственная кластеризация для приложений с шумами (DBSCAN – Density-based spatial clustering of applications with noise)
  - Ester M., Kriegel H.-P., Sander J., Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise // Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), 1996. P. 226–231.
- Требует на входе два параметра:
  - `eps` – радиус окрестности
  - `MinPts` – минимальное количество соседей
- Не требует задания количества кластеров

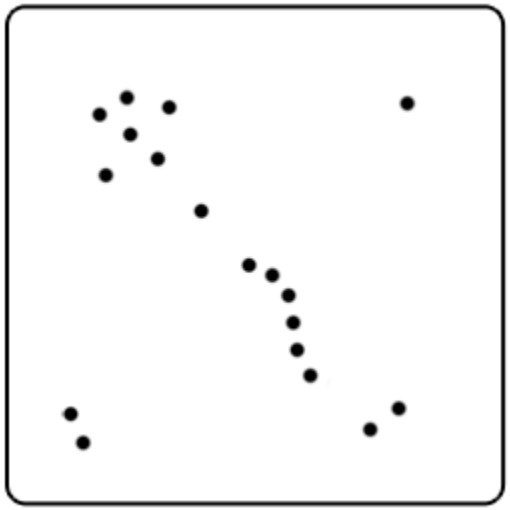
DBSCAN vs. K-Means



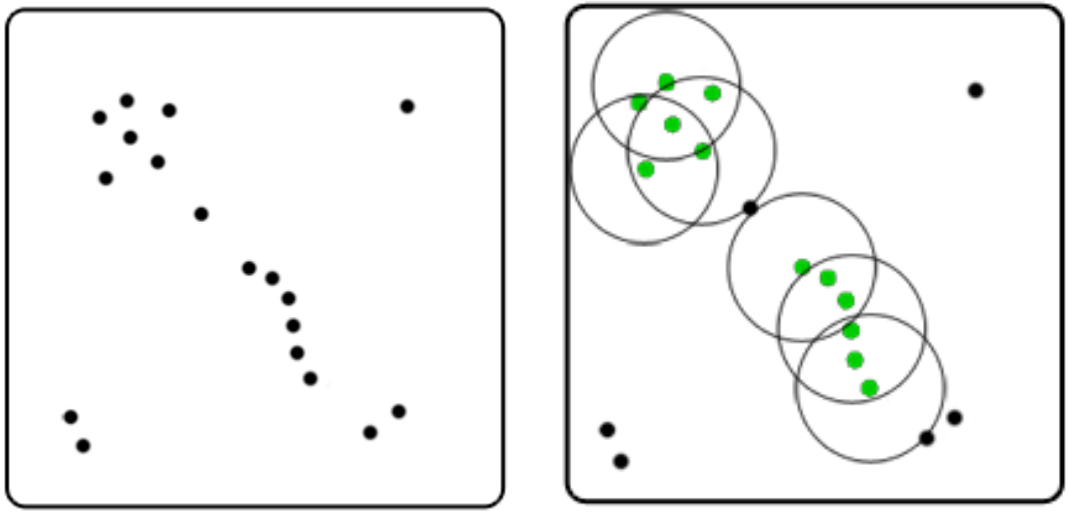
Алгоритм DBSCAN



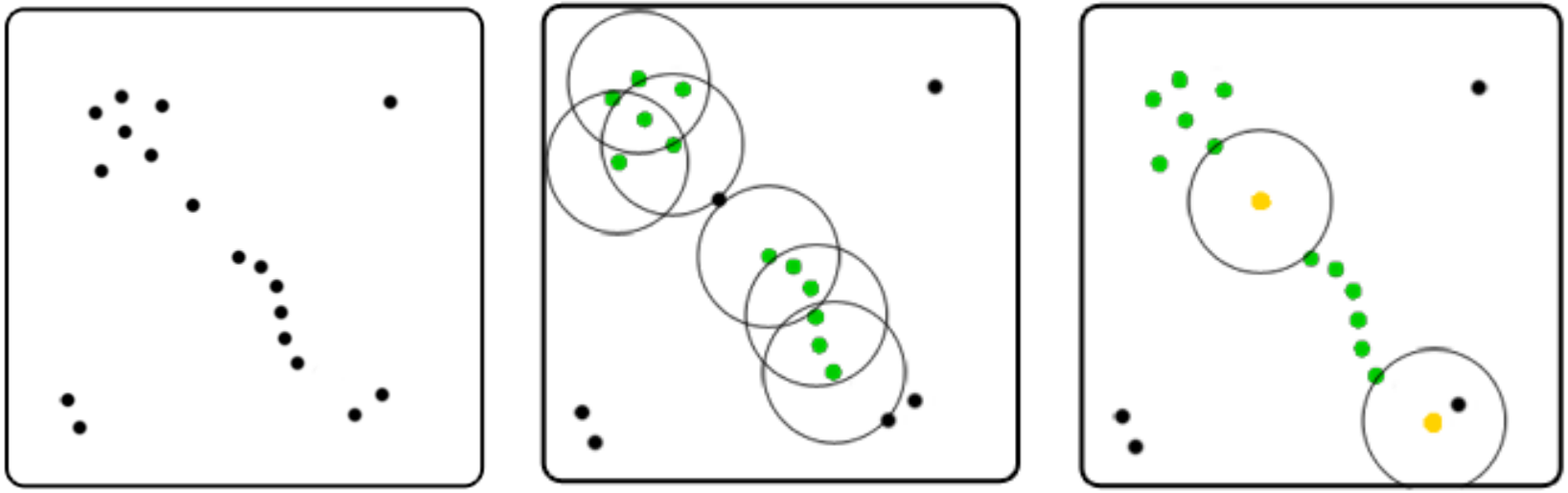
Алгоритм DBSCAN



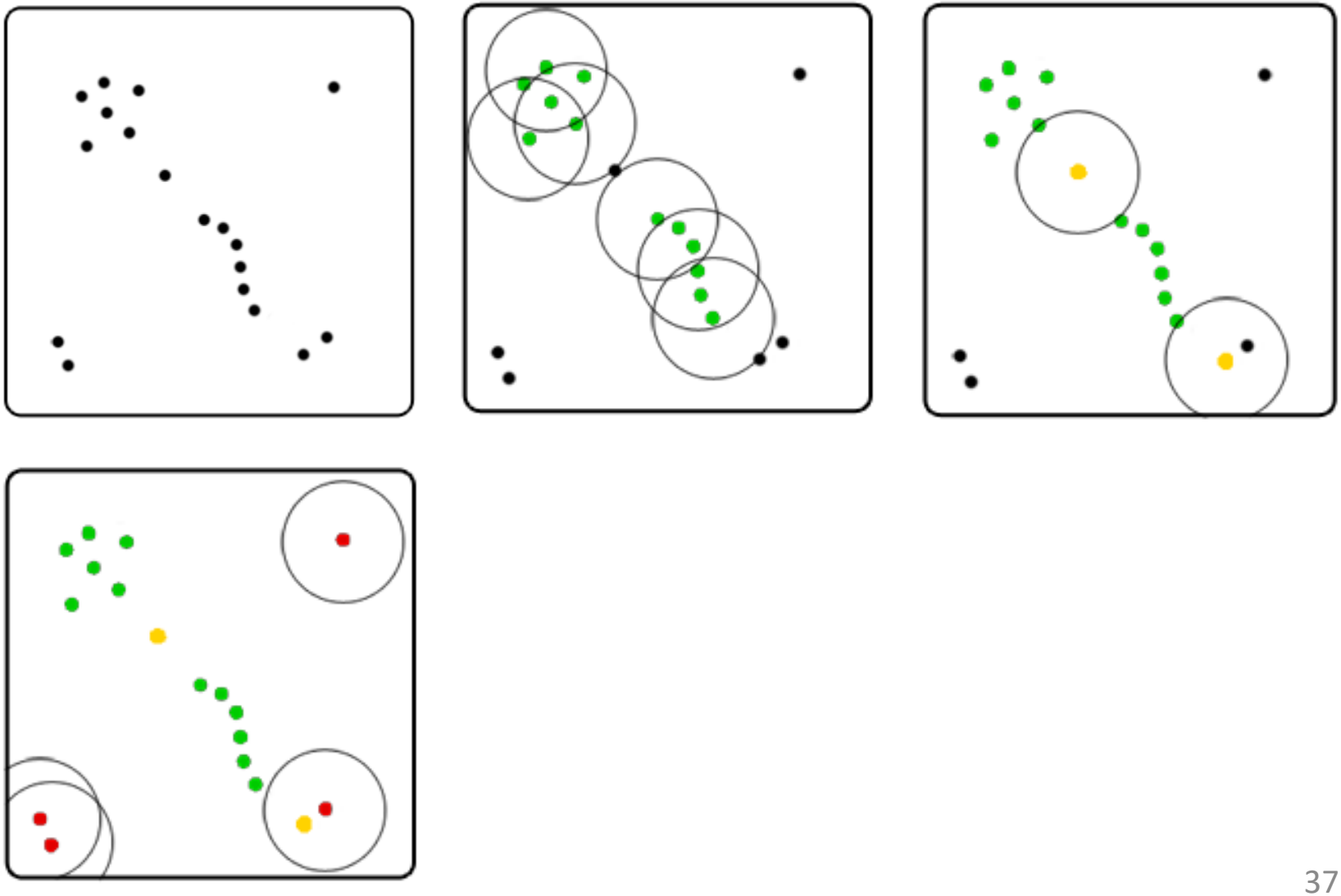
Алгоритм DBSCAN



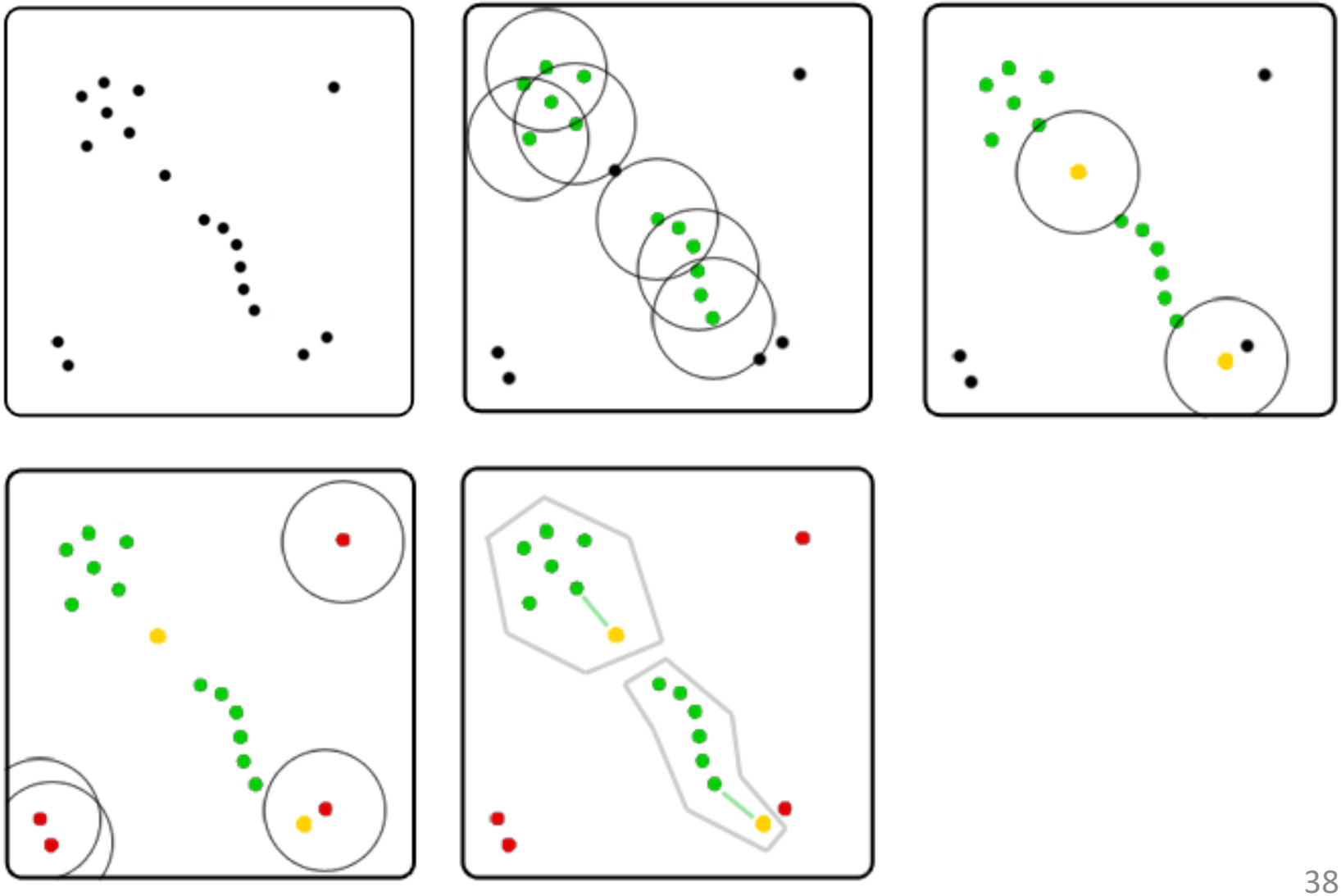
Алгоритм DBSCAN



Алгоритм DBSCAN



Алгоритм DBSCAN



Алгоритм DBSCAN

```
DBSCAN(DB, distFunc, eps, minPts) {
   C=0                                                  /* Счётчик кластеров
   for each point P in database DB {
      if label(P) ≠ undefined then continue             /* Точка уже была просмотрена
      Neighbors N = RangeQuery(DB, distFunc, P, eps)    /* Находим соседей
      if |N|< minPts then {                             /* Проверка плотности
         label(P) = Noise                               /* Помечаем как шум
         continue
      }
      C = C + 1                                         /* Следующая метка кластера
      label(P) = C                                      /* Помечаем начальную точку
      Seed set S = N \ {P}                              /* Соседи для расширения
      for each point Q in S {                           /* Обрабатываем каждого соседа
         if label(Q) = Noise then label(Q) = C          /* Заменяем метку Шум на Край
         if label(Q) ≠ undefined then continue          /* Была просмотрена
         label(Q) = C                                   /* Помечаем соседа
         Neighbors N = RangeQuery(DB, distFunc, Q, eps) /* Находим соседей
         if |N| ≥ minPts then {                         /* Проверяем плотность
            S = S ∪ N                                   /* Добавляем соседей в набор
         }
      }
   }
}
```

Алгоритм DBSCAN

- Визуализация:
  - https://www.naftaliharris.com/blog/visualizing-dbscan-clustering

Алгоритм DBSCAN

```
class sklearn.cluster.DBSCAN(
    eps=0.5,
    *,
    min_samples=5,          # includes the point itself
    metric='euclidean’,
    metric_params=None,
    algorithm='auto’,       # NearestNeighbors
    leaf_size=30,           # for NearestNeighbors
    p=None,                 # power of the Minkowski metric
    n_jobs=None
)
```

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

Алгоритм DBSCAN

- Достоинства:
  - не требует задания количества кластеров
  - не чувствителен к выбросам
  - может обнаруживать кластеры любой формы
- Недостатки:
  - сложно определять параметры `eps` и `MinPts`
  - для кластеров с большой разницей в плотности оптимальными являются разные значения параметров `eps` и `MinPts`

Иерархические алгоритмы

Подходы:

- агломеративные алгоритмы (agglomerative)
- дивизимные алгоритмы (divisive)

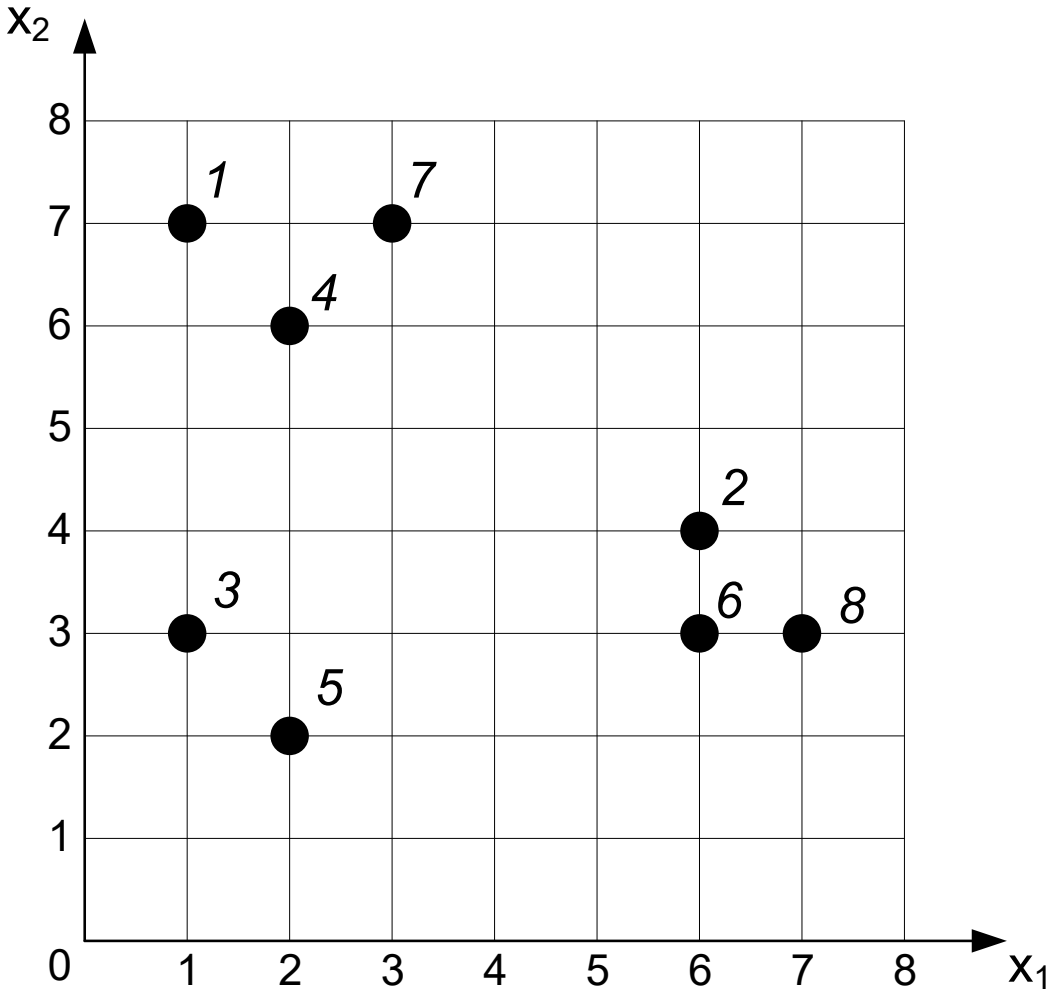
Иерархические алгоритмы

Агломеративный алгоритм:

- В начале работы алгоритма все объекты являются отдельными кластерами
- На первом шаге наиболее похожие (близкие) два кластера объединяются в один кластер
- На последующих шагах объединение продолжается до тех пор, пока все объекты не будут составлять один кластер
- На любом этапе объединение можно прервать, получив нужное число кластеров

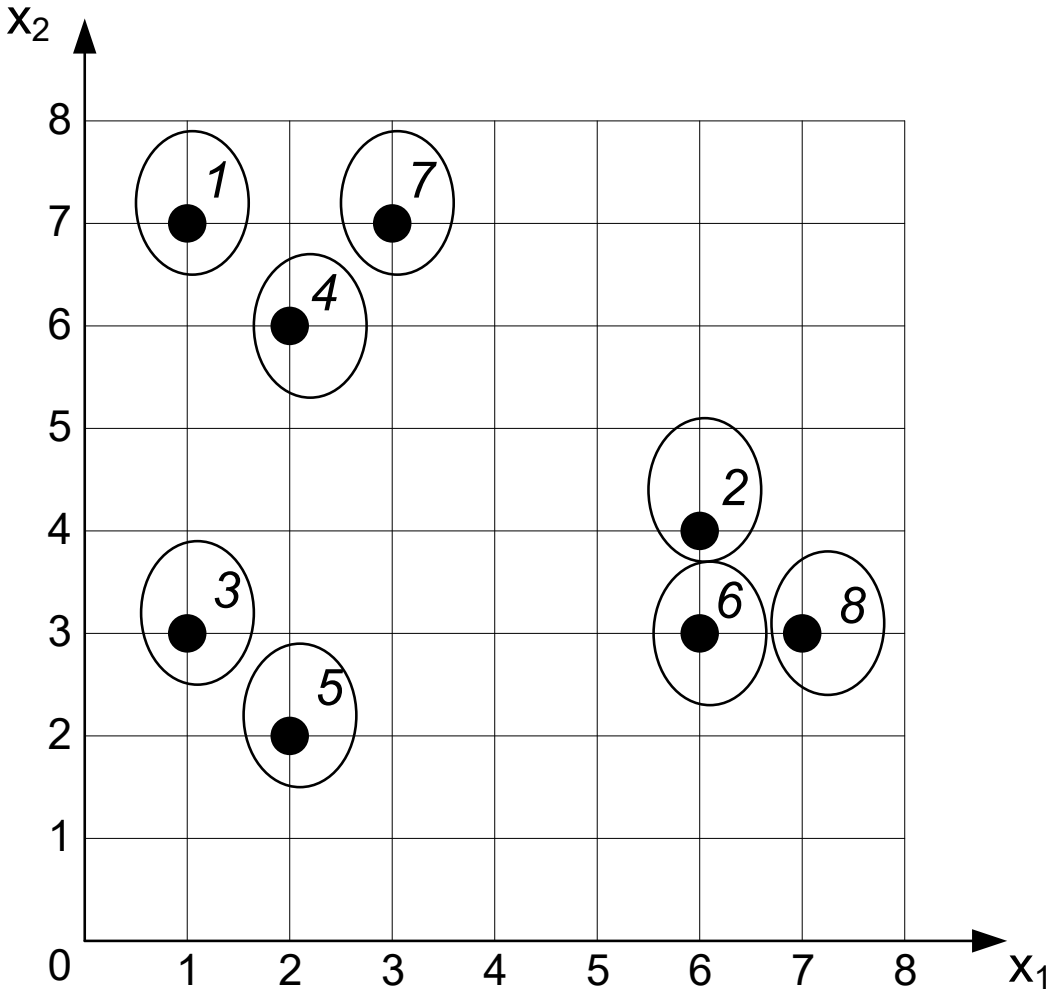
Иерархические алгоритмы

Агломеративный алгоритм:



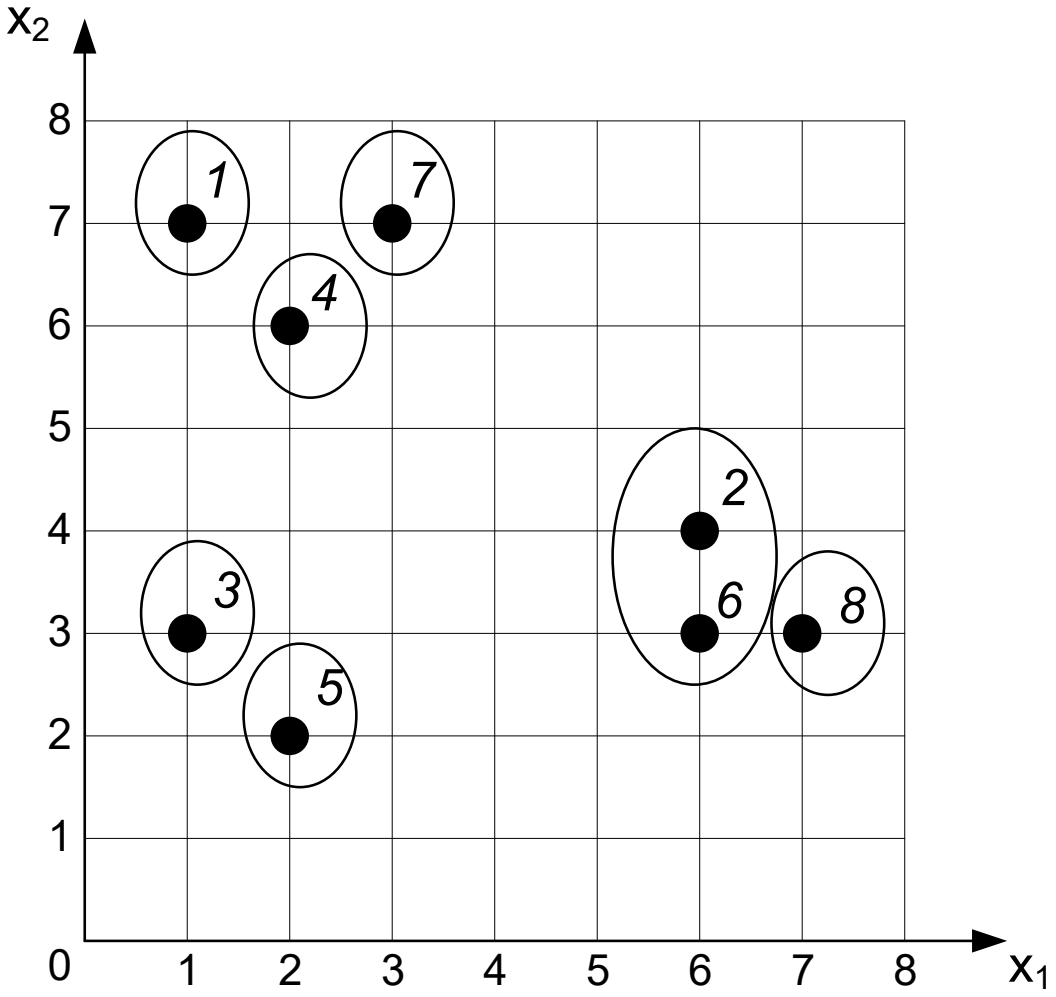
Иерархические алгоритмы

Агломеративный алгоритм:



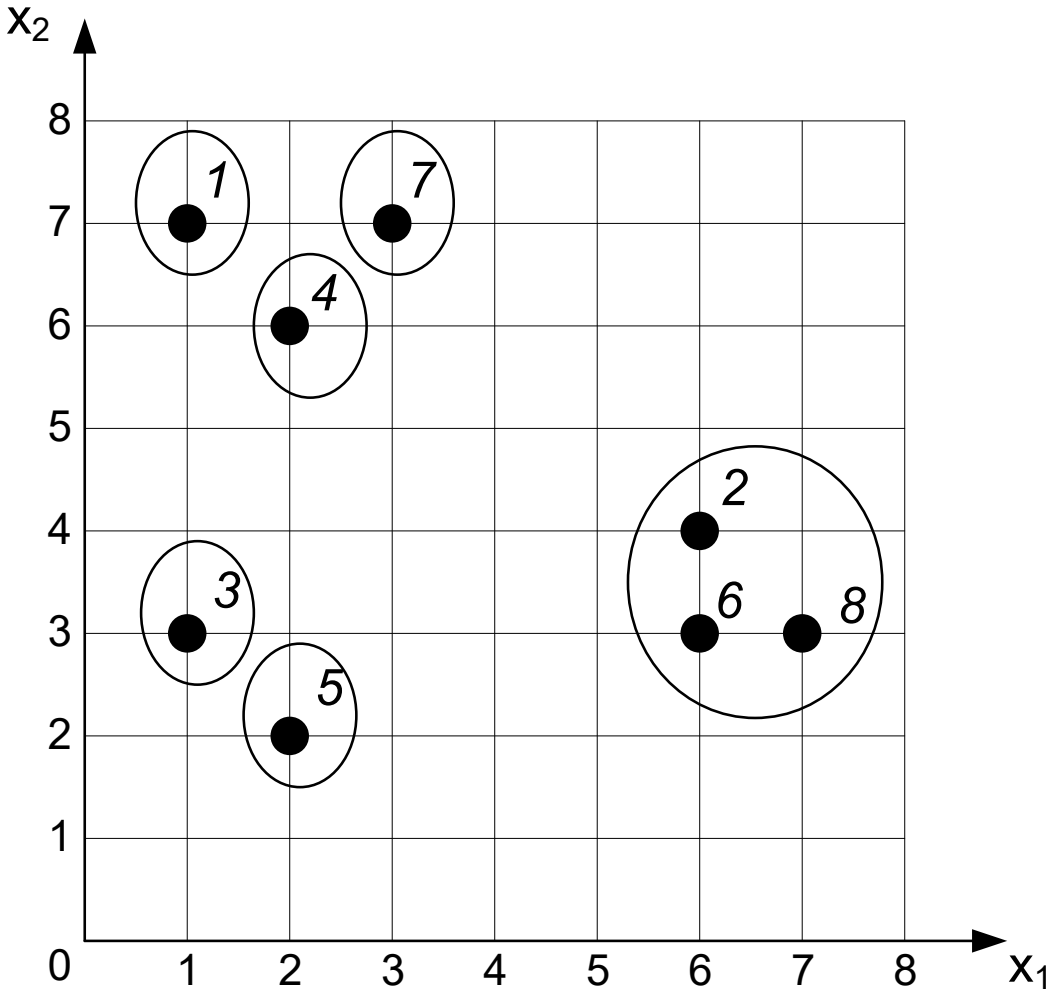
Иерархические алгоритмы

Агломеративный алгоритм:



Иерархические алгоритмы

Агломеративный алгоритм:



Иерархические алгоритмы

Расстояния между кластерами (linkage criteria):

- Maximum linkage / complete linkage:

$$d(A,B) = \max\{\rho(x,y): x \in A, y \in B\}$$

- Minimum linkage / single linkage:

$$d(A,B) = \min\{\rho(x,y): x \in A, y \in B\}$$

- Average linkage:

$$d(A,B) = \frac{1}{|A||B|}\sum_{x\in A}\sum_{y\in B}\rho(x,y)$$

Иерархические алгоритмы

Расстояния между кластерами (linkage criteria):

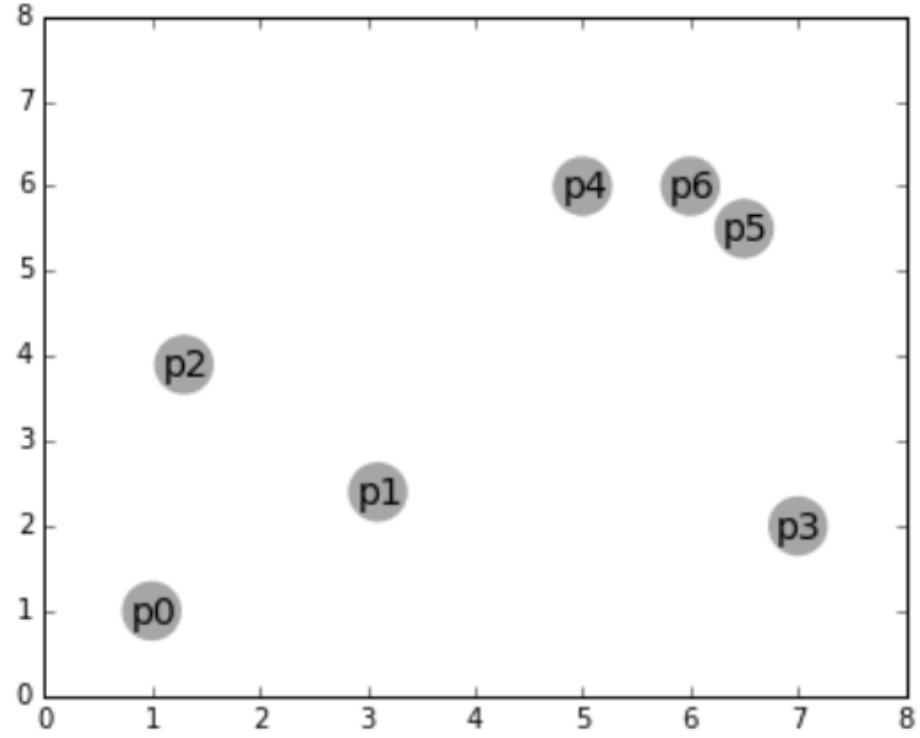
- Ward's method:

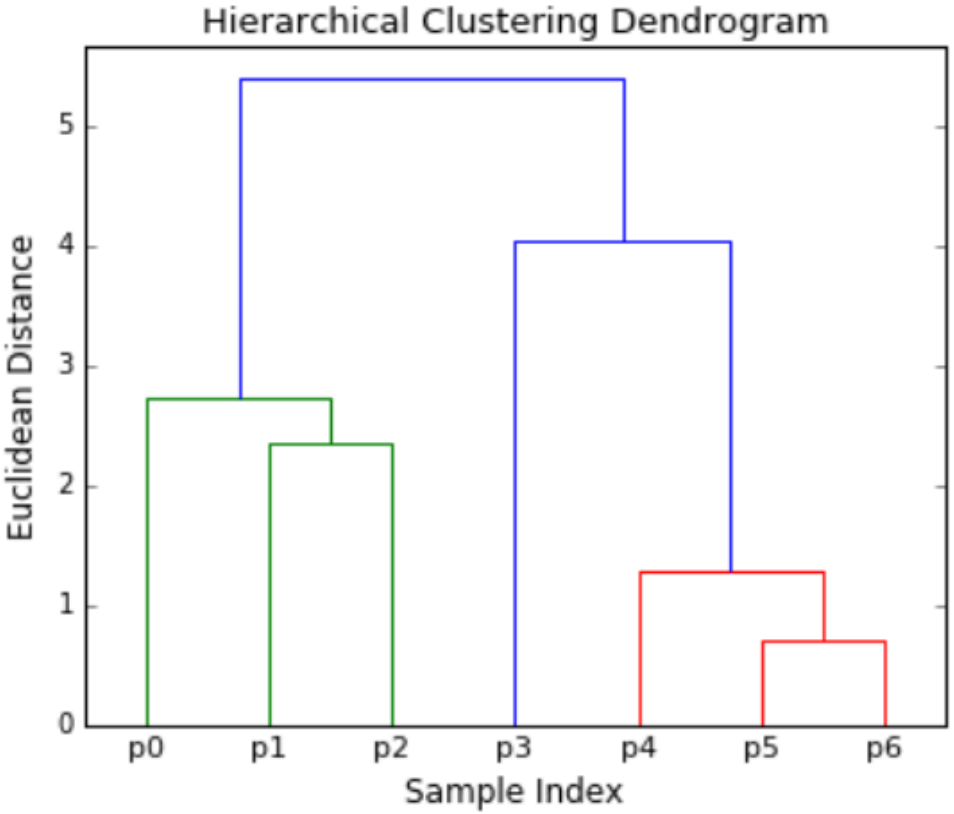
$$d(A,B) = \frac{N_A N_B}{N_A + N_B}\rho(c_A, c_B)$$

где c_A, c_B – центры кластеров A и B,

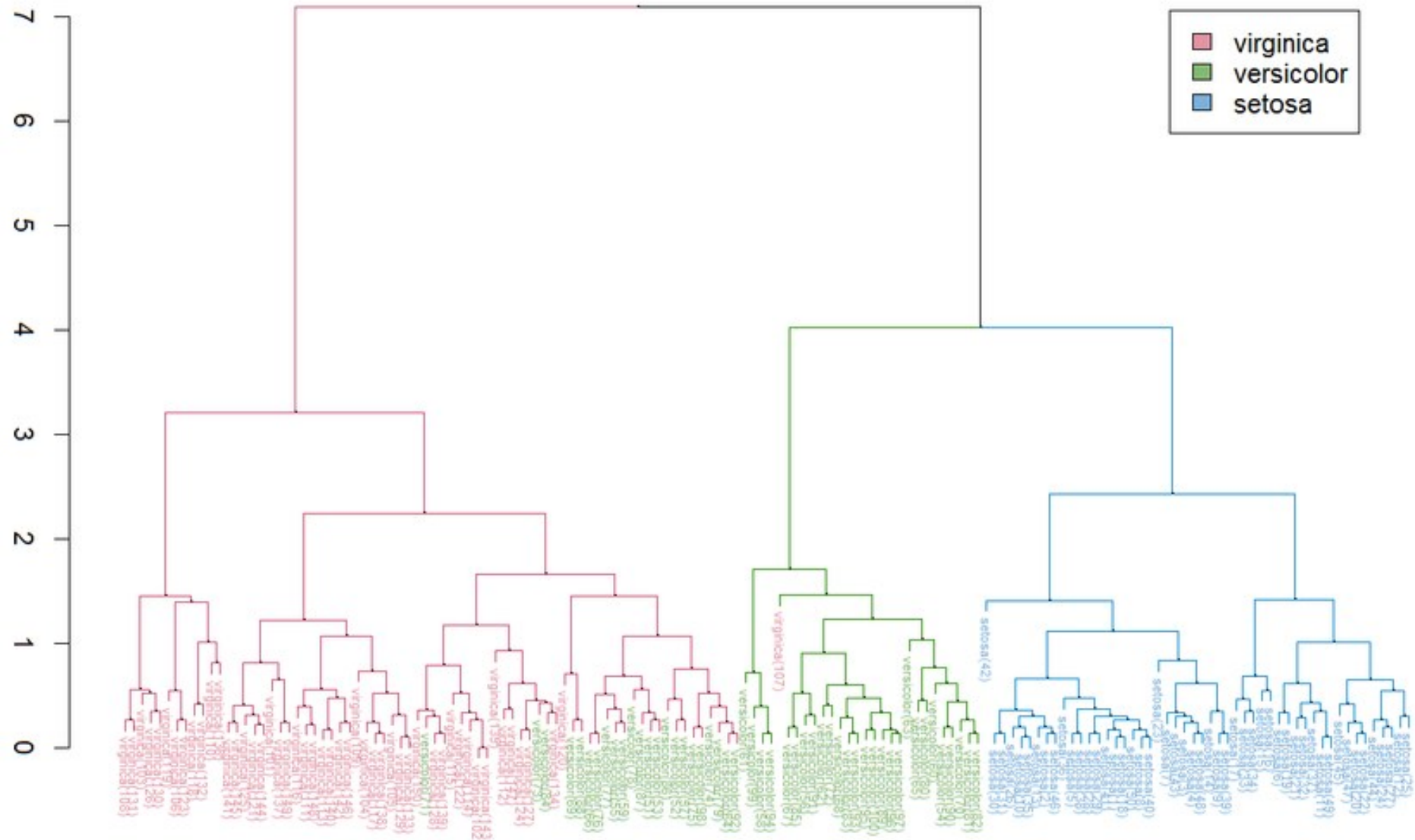
N_A, N_B – количество объектов в кластерах A и B

Иерархические алгоритмы – дендрограммы





Ирисы Фишера



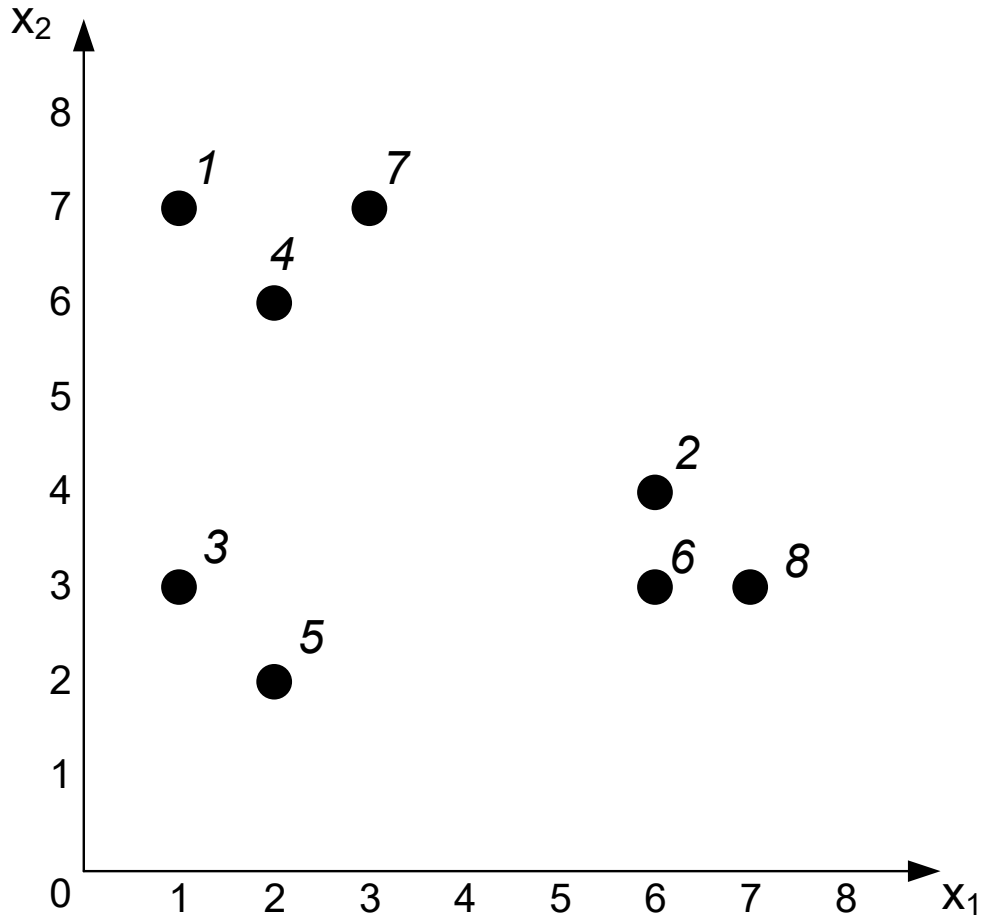
Алгоритм минимального остовного дерева

- Остовное дерево графа – ациклический связный подграф данного графа, в который входят все его вершины
- Минимальное остовное дерево (minimal spanning tree, MST) – это остовное дерево графа, имеющее минимальный возможный вес
- Вес дерева – сумма весов входящих в него рёбер
- Алгоритмы поиска минимального остовного дерева:
  - Алгоритм Прима: $O(E + V \log V)$, $O(E \log V)$
  - Алгоритм Краскала: $O(E \log E)$
  - Алгоритм Борувки: $O(E \log V)$

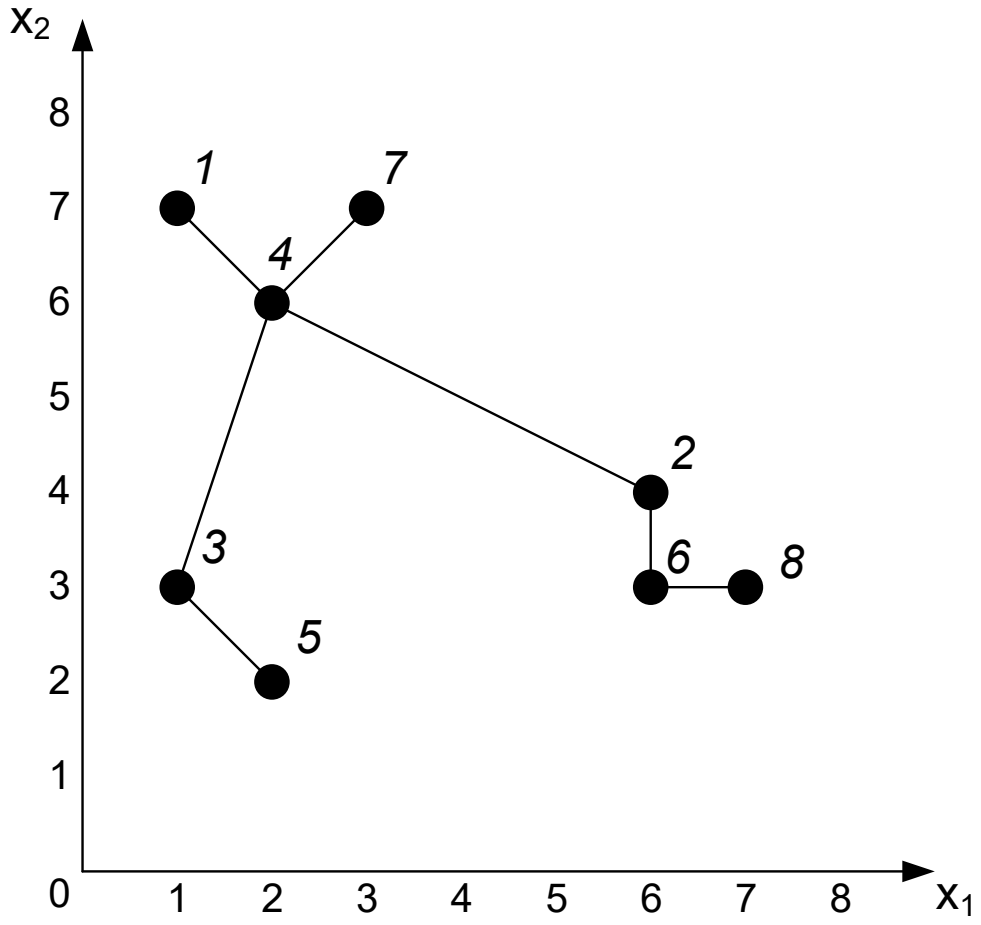
Алгоритм минимального остовного дерева

- Для кластеризации нужно построить минимальное остовное дерево, а затем удалять из него ребра максимального веса
- Сколько ребер удалим, столько кластеров (+1) получим

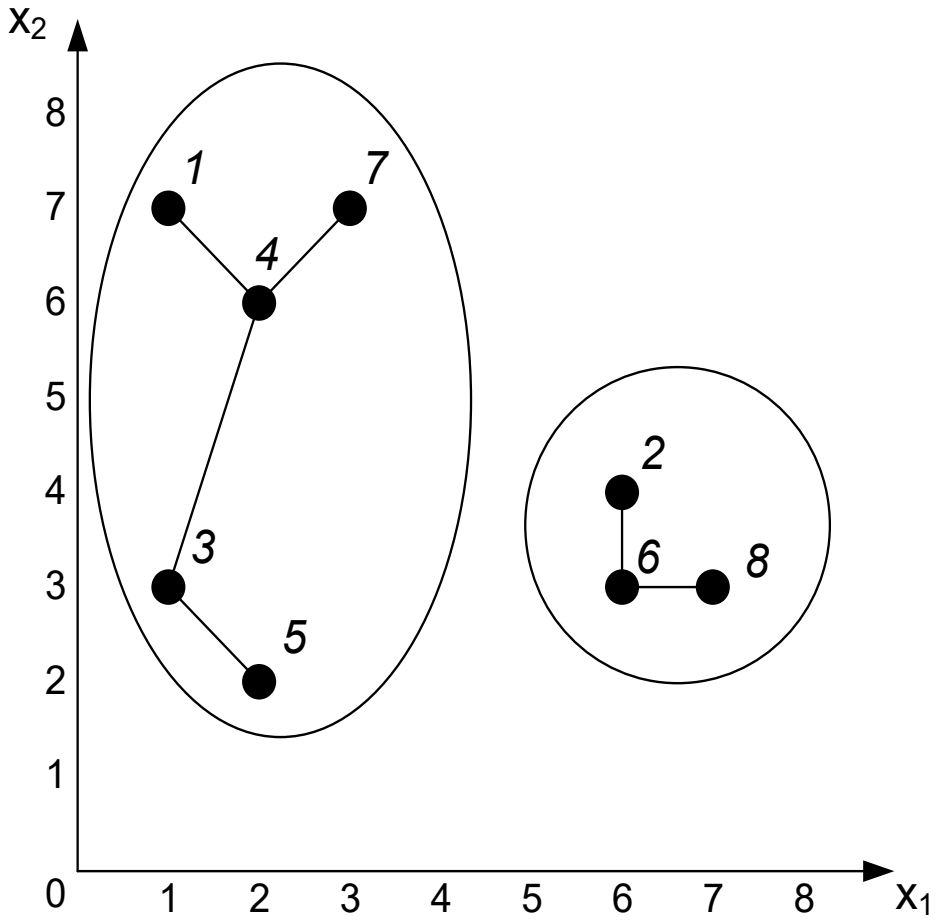
Алгоритм минимального остовного дерева



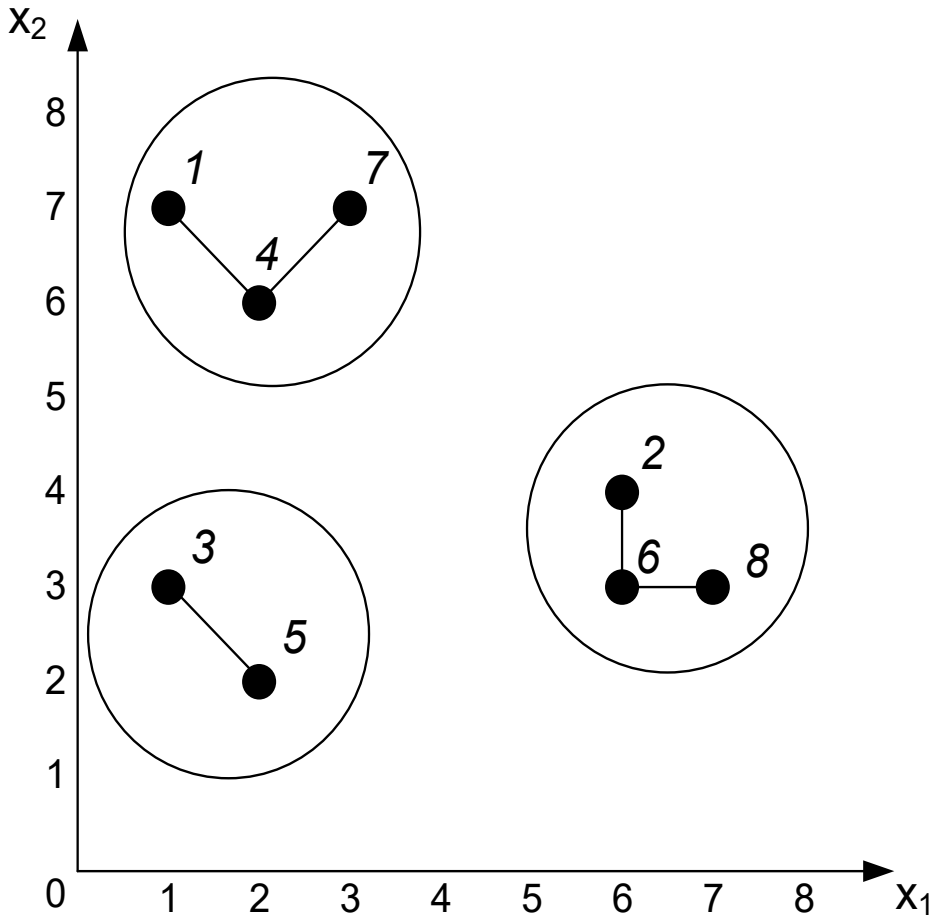
Алгоритм минимального остовного дерева



Алгоритм минимального остовного дерева



Алгоритм минимального остовного дерева



Сравнение алгоритмов (scikit-learn)