Лекция 5

Кластеризация

Типы методов кластеризации

- Ранее был рассмотрен метод кластеризации, основанный на измерении расстояния до центроидов.
- Для устранения недостатков, можно выделить следующие виды класт.:
  - Основанные на плотности
  - Агломеративная
  - Основанная на графах расстояния
  - Оценке законов распределения

DBSCAN

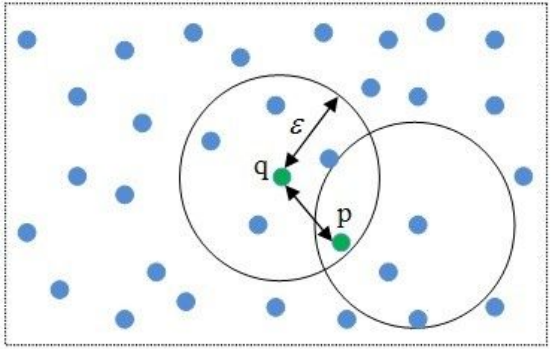
- DBSCAN - Density-based spatial clustering of applications with noise - Основанная на плотности пространственная кластеризация для приложений с шумами
- Алгоритм кластеризации - который оценивает плотность расположения точек, выделяя плотные группы точек в отдельный кластер
- Количество кластеров не фиксировано, также алгоритм выделяет точки, которые являются выбросом (шумом)
- Алгоритм определяется двумя параметрами: ε - радиус поиска, minPts - минимальное количество точек

Виды точек в DBSCAN

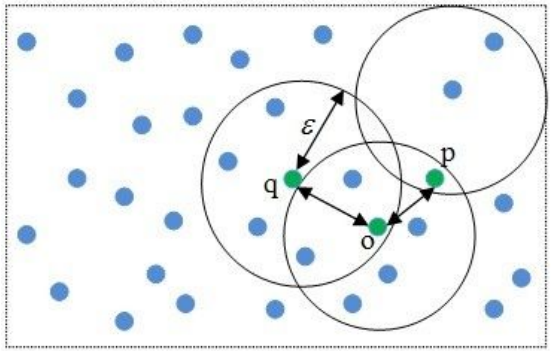
- Свойство: ε-соседство - если расстояние между двумя точками меньше ε
- Основная (core) точка - такая точка, у которой в радиусе ε находятся не меньше minPts точек, включая саму основную точку.
- Граничная (border) точка - такая точка, у которой в радиусе ε находится меньше minPts точек, но среди них есть основная точка. То есть сохраняется ε-соседство с основной точкой.
- Шум (noise) - все точки, которые не являются основными или граничными

Связи в DBSCAN (1)

- Точки q и p являются прямо достижимыми по плотности, если они основные точки и ε-соседи



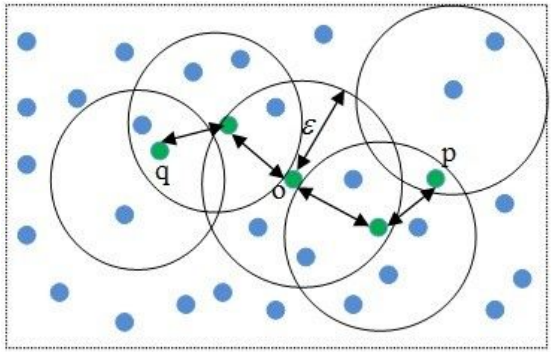
- Точки q и p являются достижимыми по плотности, если есть цепочка точек p_1, p_2, ..., p_n, где $p_1 = q$ и $p_n = p$. И каждая p_i и p_{i+1} точка являются прямо достижимыми по плотности точками



Связи в DBSCAN (2)

- Точки q и p являются связанными по плотности, если у каждой в радиусе ε есть основные точки, которые достижимы по плотности

- Все точки, которые связаны между собой по плотности образуют единый кластер



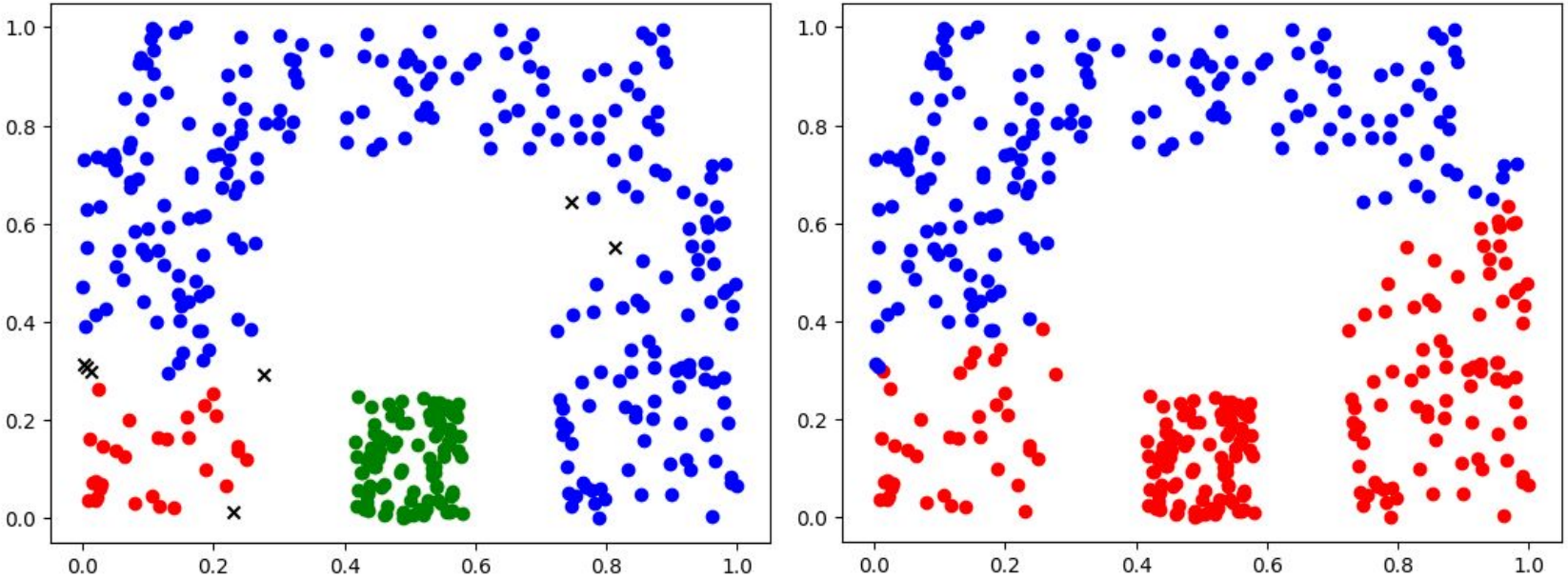
Алгоритм DBSCAN

1. Формируется множество всех основных непроверенных точек D по параметрам ε и minPts
2. Если множество D не пустое, то берется случайная основная точка и добавляется в очередь Q. Начинает формироваться новый кластер. Если множество D пустое, то алгоритм завершается.
3. Если очередь Q пустая, то шаг 2, иначе, достается следующая точка P из Q:
    a. Если P основная, то добавляется к текущему кластеру. P удаляется из D. Все ε-соседи точки P добавляются в очередь Q.
    b. Если P граничная, то добавляется к текущему кластеру.
4. Точки, которые не попали ни в один кластер помечаются как шум

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

Сравнение DBSCAN и K-Means

```
dbscan = DBSCAN(eps = 0.08, min_samples = 6)
dbscan_clust = dbscan.fit_predict(dens_data)
```



Особенность в SKlearn - отсутствие метода predict

OPTICS

- Основной недостаток DBSCAN - предполагается то, что у всех кластеров схожая плотность
- OPTICS - Ordering points to identify the clustering structure - Упорядочение точек для обнаружения кластерной структуры
- OPTICS - также как и DBSCAN оперирует с понятием основной точки, но оперирует с понятиями основного расстояния, которое определяет плотность кластера, и расстоянием достижимости.

OPTICS расстояние

- Основное расстояние - показывает минимальный радиус от основной точки p в котором лежит minPts точек:

$$\text{core-dist}_{\varepsilon, MinPts} = \begin{cases} \text{UNDEFINED} & |N_\varepsilon(p)| < MinPts \\ MinPts\text{-th}N_\varepsilon(p) & |N_\varepsilon(p)| \geqslant MinPts \end{cases}$$

- Достижимое расстояние для точек p и o равно максимуму из основного расстояния и обычного расстояния между ними:

$$\text{reachability-dist}_{\varepsilon, MinPts}(o, p) = \begin{cases} \text{UNDEFINED} & |N_\varepsilon(p)| < MinPts \\ \max(\text{core-dist}_{\varepsilon, MinPts}(p), \text{dist}(p, o)) & |N_\varepsilon(p)| \geqslant MinPts \end{cases}$$

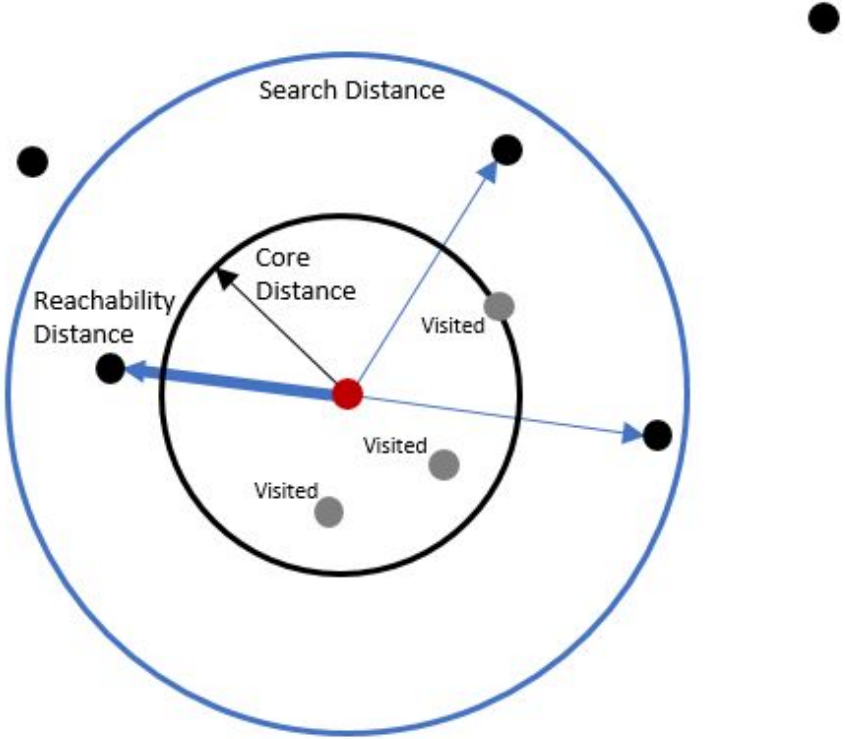
- Таким образом, для каждой основной точки, основное расстояние обозначает плотность кластера, в который она должна быть отнесена

OPTICS расстояние

Одним из параметров OPTICS является max_eps - максимальное расстояние для поиска соседних точек.

По умолчанию = бесконечность

Позволяет оптимизировать алгоритм



Алгоритм OPTICS

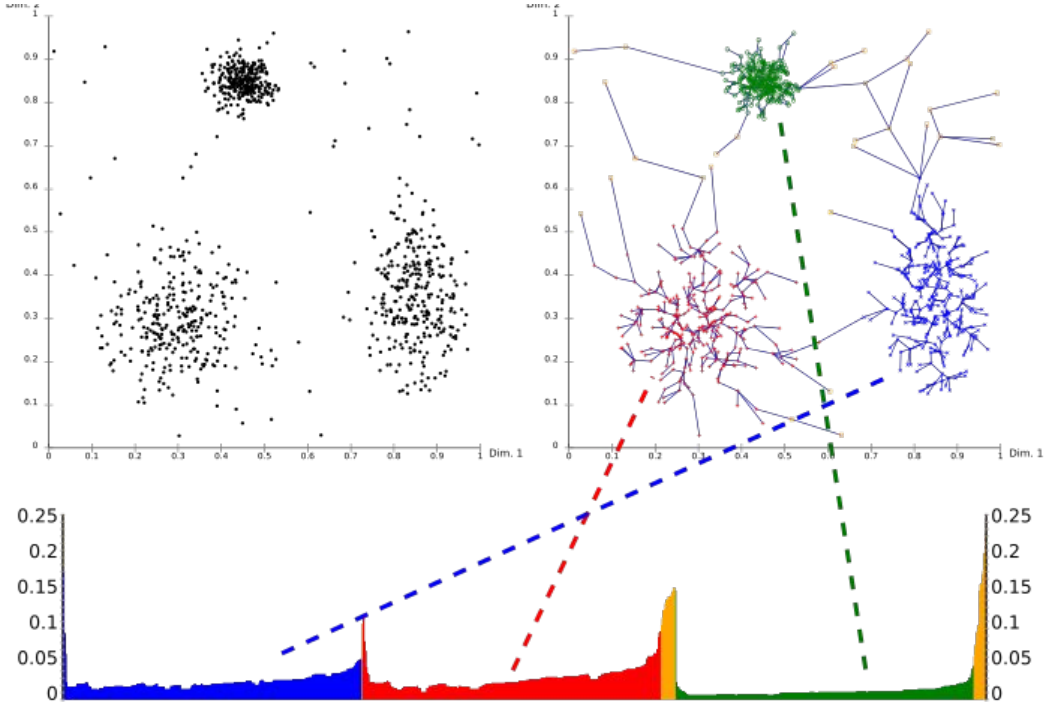
1. Для всех точек принять. Достижимость - не определена
2. Для каждой не посещенной точки:
    a. Рассчитать кол-во соседей. Если точка не основная, то перейти на шаг 2.
    b. Если точка основная. Рассчитать основное расстояние.
3. Для основной точки сделать очередь с приоритетом, добавив туда всех соседей из радиуса поиска. Если для соседа уже была рассчитана достижимость, то обновить по минимальному значению.

Таким образом, алгоритм не строит кластеры на прямую, а только хранит дерево достижимости

Особенность OPTICS

По дереву достижимости, можно построить график достижимости, где по x отмечены точки в порядке их обработки, а по y значение достижимости.

Далее варьируя значение по y, можно разделять кластеры и отделять шумы.



Применение OPTICS

```
opt = OPTICS(min_samples = 30,
             cluster_method = 'dbscan',
             eps = 0.159)
```

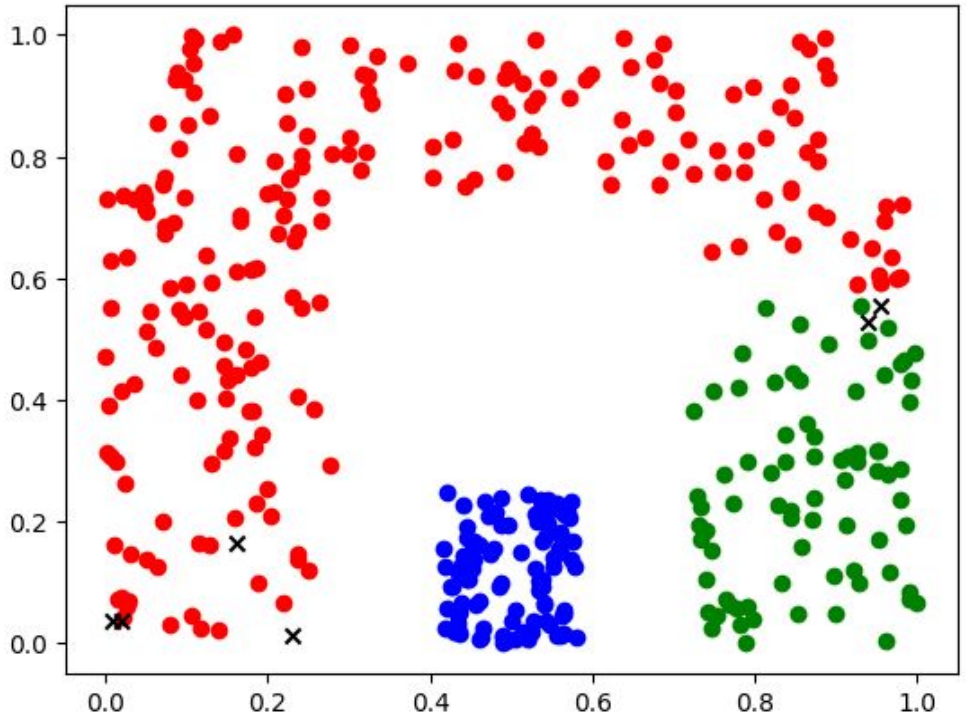
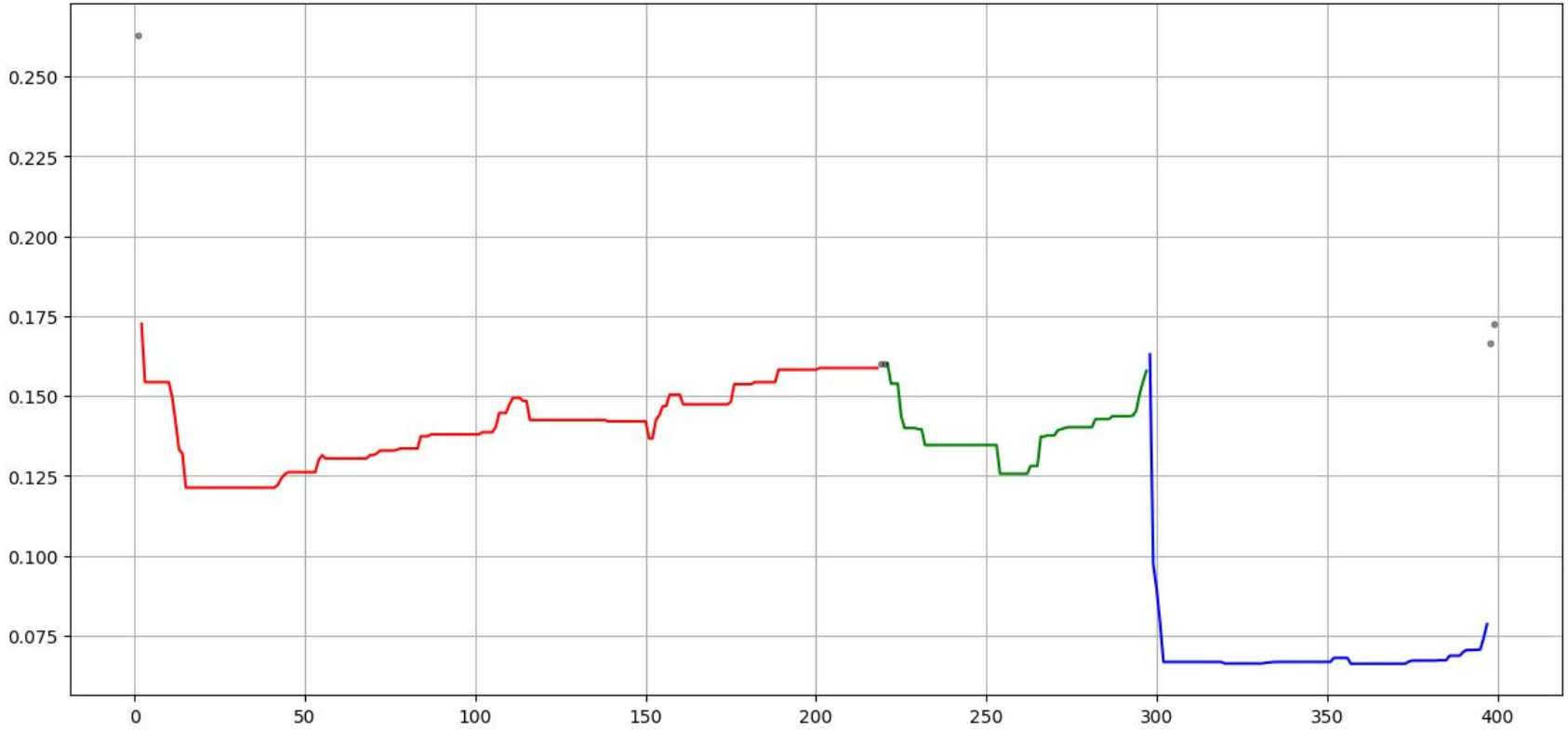


График достижимости



Иерархическая кластеризация

- Иерархическая (агломеративная) кластеризация - алгоритм кластеризации, при котором объединяются малые кластеры в один большой.
- Изначально, каждое наблюдения представляет отдельный кластер. Кластеры объединяются до тех пор, пока не будет получен один большой кластер (если не было задано других условий на остановку алгоритма)
- В качестве условий на остановку может быть указано:
    - Было получено N кластеров
    - Превышение предельной дистанции между кластерами

Иерархическая кластеризация - пример [1]

- Используется Манхэттенское расстояние
- Расстояние между кластерами - расстояние между центроидами

Набор данных.
На 1 шаге, каждая точка кластер

№	x	y
1	0	0
2	1	1
3	0	3
4	0	4
5	2	3

Матрица расстояний

0	2	3	4	5
	0	3	4	3
		0	1	2
			0	3
				0

Иерархическая кластеризация - пример [2]

- Объединили 3 и 4 кластер, получили новый кластер с центром [0, 3.5]
- Получили новые кластеры - пересчитали попарные расстояния

№	x	y
1	0	0
2	1	1
(3, 4)	0	3.5
5	2	3

Матрица расстояний

0	2	3.5	5
	0	3.5	3
		0	2.5
			0

Иерархическая кластеризация - пример [3]

- Объединили 1 и 2 кластер, получили новый кластер с центром [0.5, 0.5]
- Получили новые кластеры - пересчитали попарные расстояния

№	x	y
(1, 2)	0.5	0.5
(3, 4)	0	3.5
5	2	3

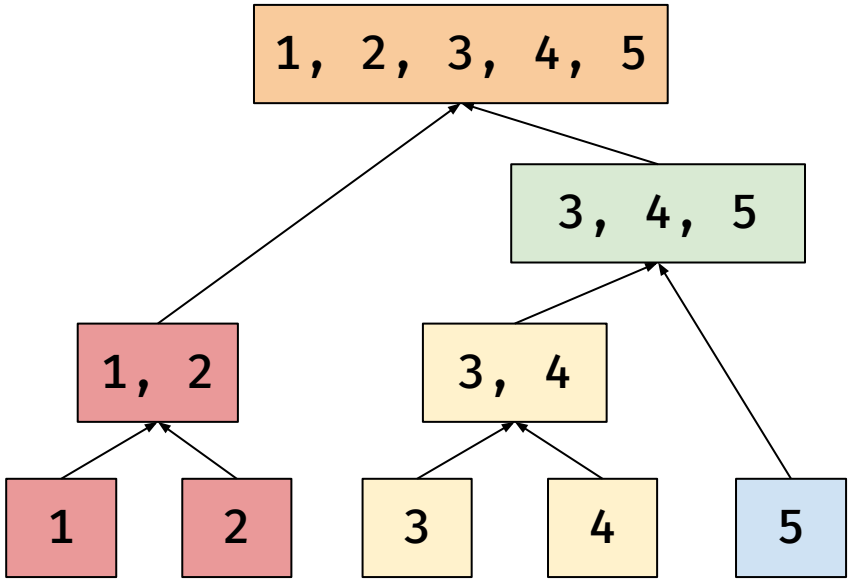
Матрица расстояний

0	3.5	4
	0	2.5
		0

Иерархическая кластеризация - пример [4]

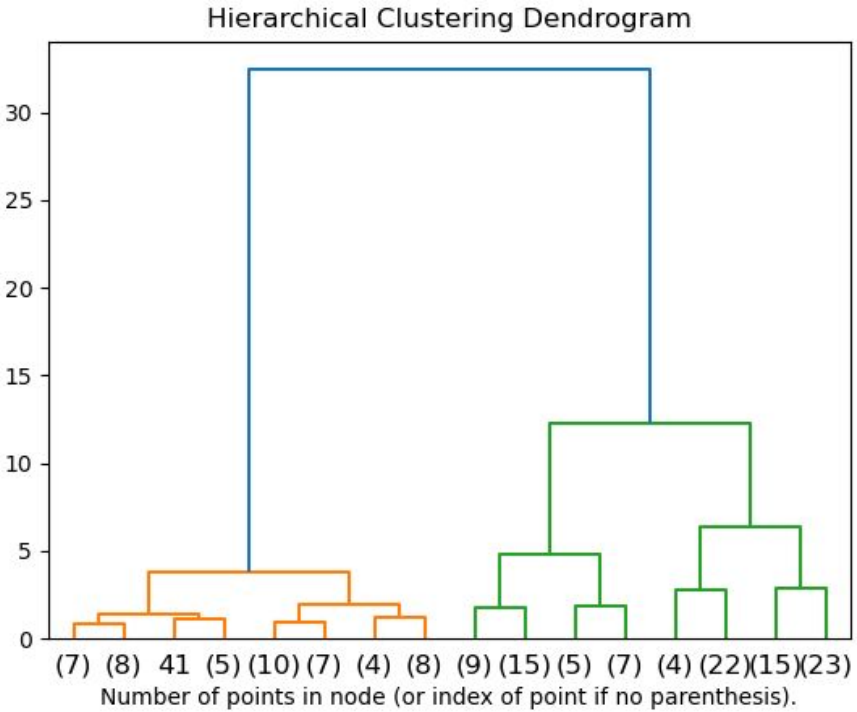
- Объединили (3, 4) и 5 кластер, получили новый кластер с центром [1, 3.25]
- Осталось 2 кластера, последний шаг тривиальный

№	x	y
(1, 2)	0.5	0.5
((3, 4), 5)	1	3.25



Дендрограмма

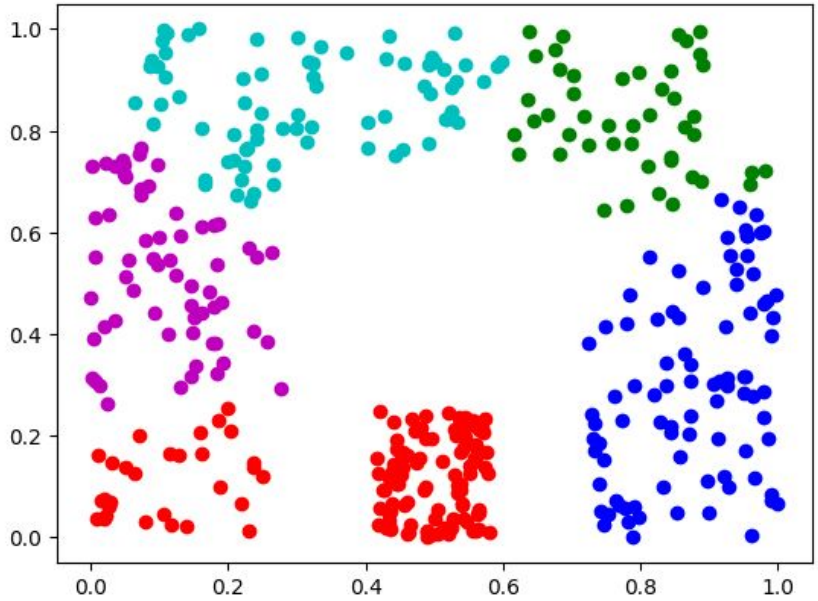
- Для визуализации иерархической кластеризации рисуется дендрограмма
- Дендрограмма отображает - какие кластеры объединялись.
- Для объединения обычно указывается номер шага, либо расстояния между кластерами.



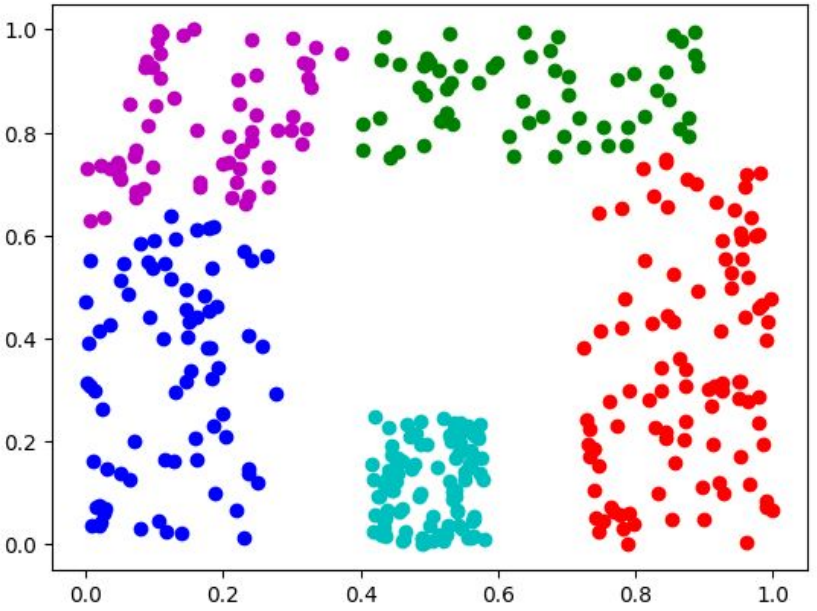
https://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_dendrogram.html#sphx-glr-auto-examples-cluster-plot-agglomerative-dendrogram-py

AgglomerativeClustering

```
agc = AgglomerativeClustering(n_clusters = 5,
                              linkage = 'average')
ag_clust = agc.fit_predict(dens_data)
```




```
agc2 = AgglomerativeClustering(n_clusters = 5,
                               linkage = 'ward')
ag_clust2 = agc2.fit_predict(dens_data)
```



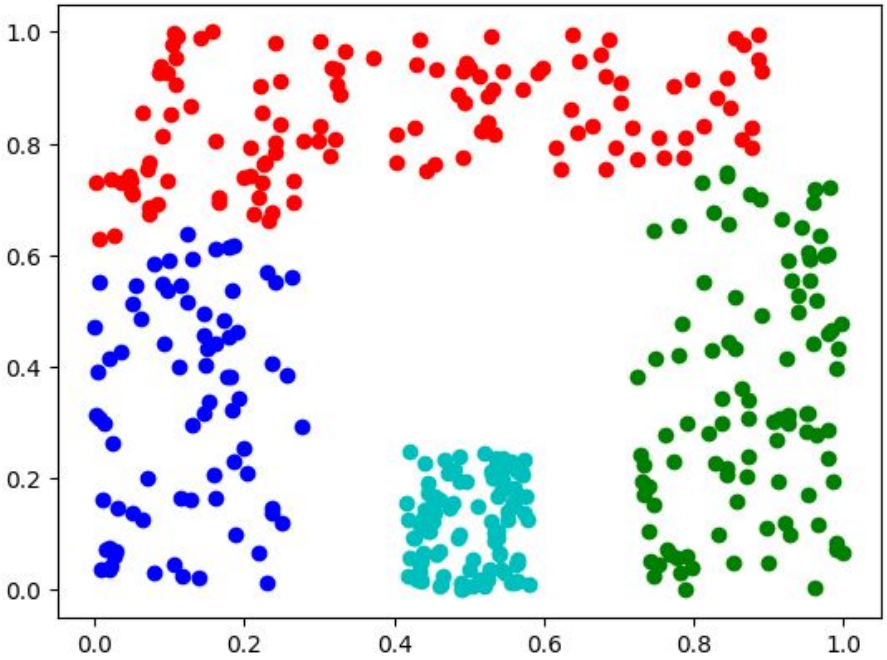
Расстояние между кластерами

- Пока кластеры состоят из одной точки, между ними легко рассчитать расстояние
- Если кластеры состоят из множества точек, то необходимо учитывать их все
- Параметр linkage позволяет определить то, как считается расстояние между класт.
- ‘ward’ - минимизация дисперсии в кластерах. Аналог инерции из K-means. Параметр по умолчанию, работает только с евклидовым расстоянием.
- ‘average’ - минимизирует среднее расстояние между каждыми точками двух кластеров. Альтернатива для не евклидовых расстояний.
- ‘complete’ - минимизирует максимальное расстояние между точками двух кластеров
- ‘single’ - минимизирует расстояние между ближайшими точками двух кластеров. Эффективна только в явно разделяемых кластерах со сложной формой. Не устойчива к шумам.

Кластеризация с ограничением по расстоянию

```
agc3 =
     AgglomerativeClustering(
          n_clusters = None,
          linkage = 'ward',
          distance_threshold = 4)
ag_clust3 = agc3.fit_predict(dens_data)

set(ag_clust3) # {0, 1, 2, 3}
```

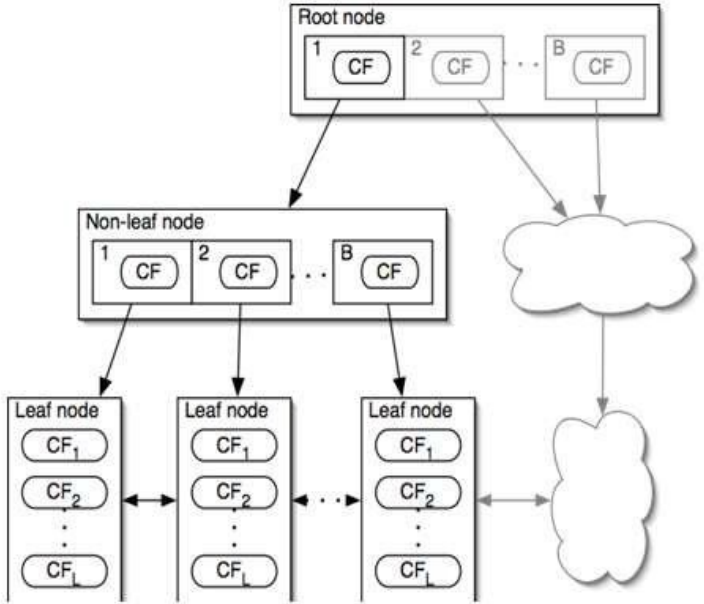


BIRCH

- BIRCH - balanced iterative reducing and clustering using hierarchies - Сбалансированное итеративное сокращение и кластеризация с помощью иерархий
- Алгоритм кластеризации, который эффективно применяется на больших наборах данных.
- Алгоритм не работает с признаками напрямую, а вычисляет для каждого кластера информацию, которая хранится в CF-дереве, и которую легко обновлять

CF-дерево

- CF-дерево - характеризуется:
  - B - фактор ветвления. Сколько дочерних узлов может в быть не в листе.
  - T - порог. Определяет максимальный радиус кластера.
  - L - максимальное кол-во листов одного узла.
- Каждый узел дерева хранит следующую информацию: $CF = (N, \overrightarrow{LS}, SS)$



Узел CF-дерева

- N - количество наблюдений во всех подкластерах
- Линейная сумма всех наблюдений в подкластерах: $\overrightarrow{LS} = \sum_{i=1}^{N} \overrightarrow{X_i}$
- Сумма квадров всех наблюдений в подкластерах: $SS = \sum_{i=1}^{N} (\overrightarrow{X_i})^2$

На основе хранимой информации можно легко рассчитать:

- Центроид кластера $\vec{C} = \frac{\sum_{i=1}^{N} \overrightarrow{X_i}}{N} = \frac{\overrightarrow{LS}}{N}$
- Радиус кластера $R = \sqrt{\frac{\sum_{i=1}^{N} (\overrightarrow{X_i} - \vec{C})^2}{N}} = \sqrt{\frac{N \cdot \vec{C}^2 + SS - 2 \cdot \vec{C} \cdot \overrightarrow{LS}}{N}} = \sqrt{\frac{SS}{N} - (\frac{\overrightarrow{LS}}{N})^2}$
- Среднее расстояние между кластерами

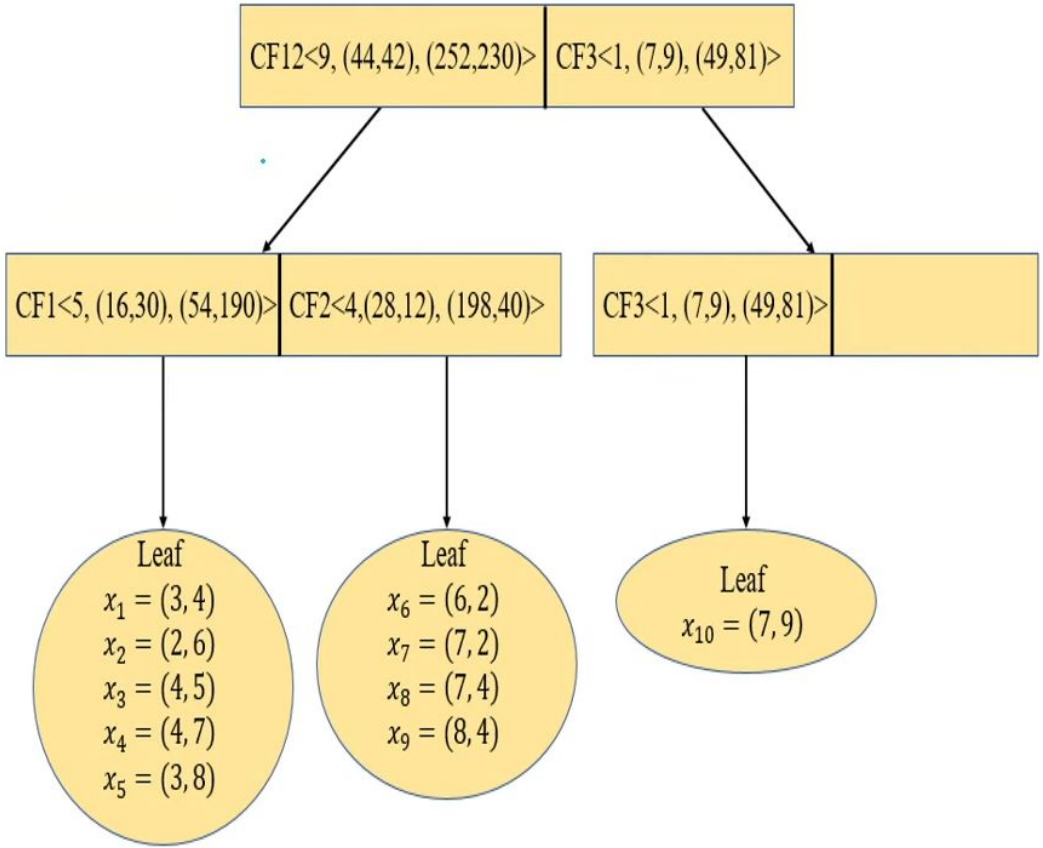
$$D_2 = \sqrt{\frac{\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} (\overrightarrow{X_i} - \overrightarrow{Y_j})^2}{N_1 \cdot N_2}} = \sqrt{\frac{N_1 \cdot SS_2 + N_2 \cdot SS_1 - 2 \cdot \overrightarrow{LS_1} \cdot \overrightarrow{LS_2}}{N_1 \cdot N_2}}$$

Алгоритм BIRCH

- Алгоритм кластеризации заключается в итеративном построении CF-дерева
- При добавлении новой точки в дерево. Определяется, в какую ветвь узла добавить точку, путем расчета LS и SS, путем простого добавления в сумму значения признаков и их квадратов соответственно.
- Оценивается параметр T. Выбирается ветвь с минимальным радиусом.
- Если радиус превышает T, то создается новая ветвь в узле. Новый узел также создается, если достигнут максимум листов L в узле.
- Если достигнут предел по количестве ветвей B, то создается новая ветвь в ближайшем родительском узле, где это возможно.
- Так как дерево не дает кластеры в явном виде, то после построения применяется более простой алгоритм кластеризации для узлов дерева на определенном уровне.

Пример CF-дерева

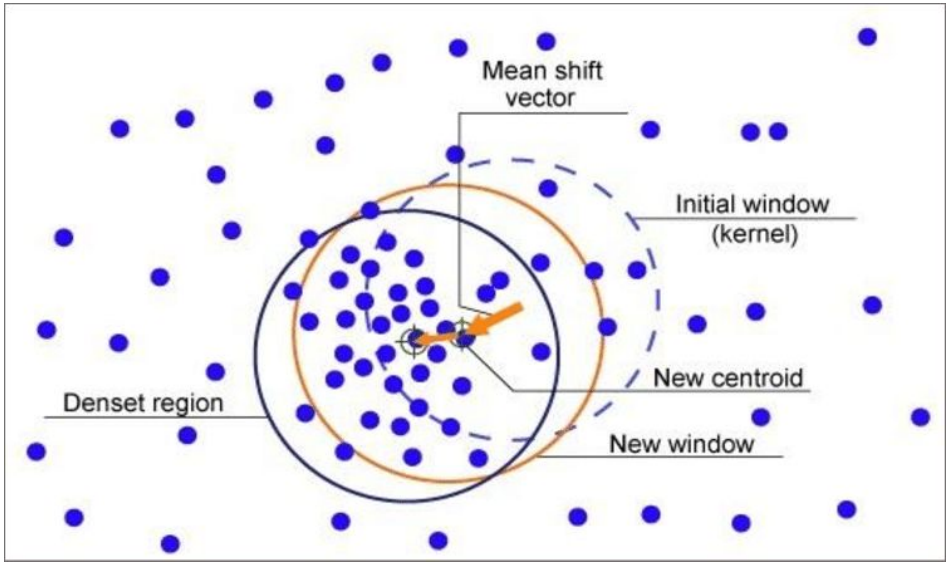
Sno	x	y	x^2	y^2
1	3	4	9	16
2	2	6	4	36
3	4	5	16	25
4	4	7	16	49
5	3	8	9	64
6	6	2	36	4
7	7	2	49	4
8	7	4	49	16
9	8	4	64	16
10	8	5	64	25



Полный пример: https://www.youtube.com/watch?v=l0Y2mWB1dBY

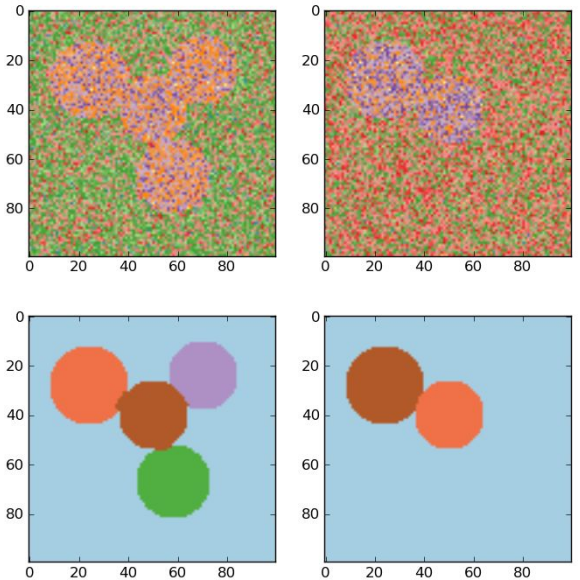
Mean-shift кластеризация

- Центроидный алгоритм с автоматическим определением количества кластеров.
- В начале вокруг каждой точки образуется ядро с заданным радиусом. Центр каждого ядра считается кандидатом для центроида.
- Рассчитывается новый центр ядра. Согласно оценке плотности точек в радиусе. Ядро смещается в сторону большей плотности точек.
- В конце, ядра, которые лежат в одной области принимаются как одно ядро. Центр оставшихся ядер определяют центроиды кластеров.



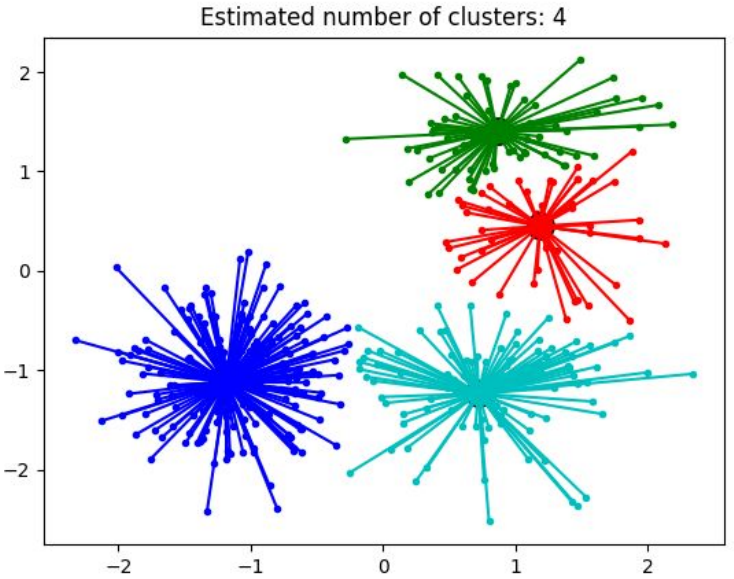
Спектральная кластеризация

- В спектральной кластеризации использует матрицы похожести между точками, представленной матрицей Кирхгофа. То есть связи между данными представляются в виде графа.
- Далее вычисляется спектр матрицы Кирхгофа, то есть его собственный вектора.
  Таким образом, происходит преобразование в пространство меньшей размерности.
- После понижения размерности используется другой алгоритм кластеризации для собственных векторов.
- Алгоритм эффективно работает для сегментации изображения с применением техники “Алгоритм нормализованных сечений”

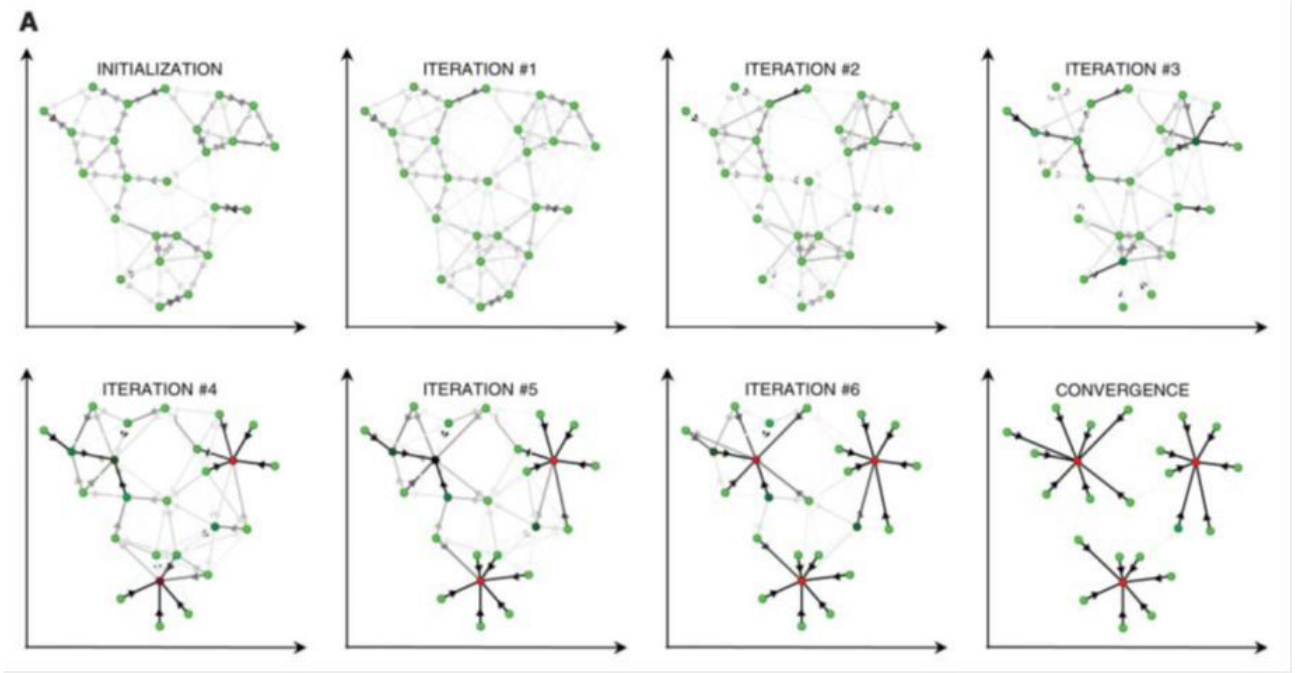


Affinity Propagation

- Алгоритм использующий матрицу схожести S.
- Основная идея заключается в том, что точки хотят “объединится” и выбрать “лидера” представляющий их всех.
- Для этого используются матрицы R - ответственности и A - доступности
- Матрица R показывает то, насколько i-я точка хочет видеть j-ю точку лидером
- Матрица A показывает то, насколько j-я точка хочет быть лидером для i-й точки
- Для выделения кластеров, считается матрица C = R + A, в которой похожие строки выделяются в один кластер

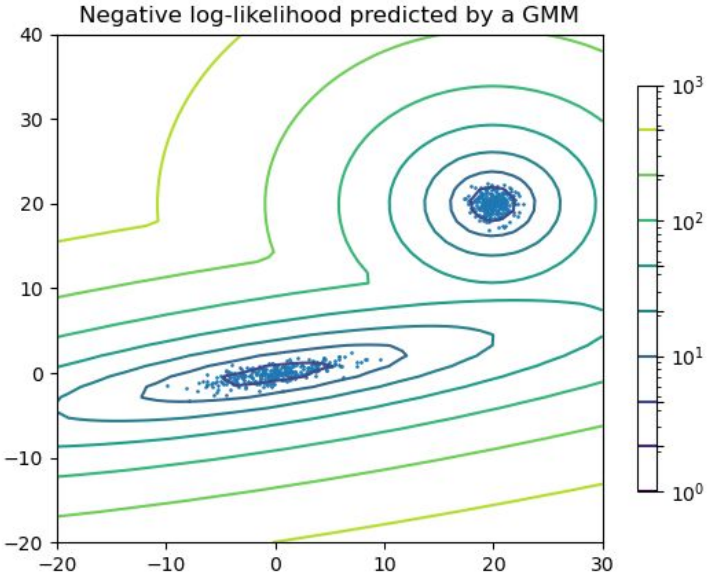


Affinity Propagation - пример итераций



Gause Mixture Model

- Модель Гауссовской смеси - алгоритм, который предполагает, что все данные сгенерированы несколькими нормальными распределениями.
- Алгоритму задается количество компонент в смеси
- Далее, алгоритм подбирает параметры каждой компоненты, используя EM-алгоритм



Обобщение

Алгоритм	**Применимость**
K-means	Равные размеры кластеров, небольшое кол-во кластеров, плоская геометрия
DBSCAN	Выделение шумов, неравные размеры кластеров, не плоская геометрия
OPTICS	Выделение шумов, неравные размеры кластеров, не плоская геометрия, разная плотность точек в кластерах
Иерархическая к.	Большое количество кластеров, не евклидовы расстояния, ограничения на расстояния
BIRCH	Большое количество наблюдений, удаление шумов, снижение размерности
Mean-shift	Много кластеров, неравные размеры кластеров, не плоская геометрия
Спектральная к.	Мало кластеров, равные размеры кластеров, не плоская геометрия
Affinity propagation	Много кластеров, неравные размеры кластеров, не плоская геометрия