Информатика. Задания на кластеризацию. Метод dbscan.

1 Разбор метода dbscan

Алгоритм ***DBSCAN*** – это метод кластеризации, который позволяет выделять кластеры различной формы и размера. Принцип работы алгоритма следующий:

- Перед началом работы алгоритма необходимо определить радиус R, в пределах которого будут рассматриваться соседние точки.
- Алгоритм начинается с выбора произвольной точки из набора данных, от которой будут искаться все остальные точки кластера.
- Для выбранной точки ищутся все точки, которые находятся на расстоянии R.
- Процесс продолжается до тех пор, пока все точки на заданном расстоянии не будут обработаны. Как только все возможные точки добавлены в кластер, алгоритм переходит к следующей непроверенной точке в наборе данных (то есть начинает формировать следующий кластер).
- Алгоритм повторяет шаги для всех непроверенных точек в наборе данных, пока не будут обработаны все точки.

Далее рассмотрим пример работы данного алгоритма на стандартном прототипе задачи. В программе мы не будем находить среднее арифметическое абсцисс и ординат центров кластеров. Наша задача – научиться распределять звезды по кластерам с помощью алгоритма ***DBSCAN***.

Условие задачи

Учёный решил провести кластеризацию некоторого множества звёзд по их расположению на карте звёздного неба. Кластер звёзд – это набор звёзд (точек) на графике, лежащий внутри окружности радиуса R. Каждая звезда обязательно принадлежит только одному из кластеров.

В файле хранятся данные о звёздах трёх кластеров. Известно, что количество звёзд не превышает 10000.

Для заданного файла требуется распределить все точки по трем кластерам с помощью алгоритма ***DBSCAN***.

Решение

```
# Импортируем функцию dist из модуля math для вычисления евклидова расстояния
from math import dist
f = open('27-6b.txt')
s = f.readline()
a = [list(map(float, i.replace(',', '.').split())) for i in f]
# Инициализируем список кластеров. Каждый кластер начинается с одной звезды,
# выбранной из 1-го, 2-го и 3-го кластера
cl = [[a.pop(7)], [a.pop(1)], [a.pop(0)]]
# Проходим по каждому кластеру (всего 3 кластера)
for k in range(3):
# Для каждой звезды в текущем кластере
    for j in cl[k]:
# Проходим по всем оставшимся звёздам в списке a
        for i in range(len(a)):
        # Проверяем, что звезда не была добавлена в кластер (не равна '*')
# и её расстояние до текущей звезды в кластере меньше 0.5
            if a[i] != '*' and dist(a[i], j) < 0.5:
# Если условия выполнены, добавляем звезду в текущий кластер
                cl[k].append(a[i])
# Помечаем звезду как обработанную, заменяя её на '*'
                a[i] = '*'
```