Информатика. Задания на кластеризацию. Повторение материала.

Учёный решил провести кластеризацию некоторого множества звёзд по их расположению на карте звёздного неба. Кластер звёзд – это набор звёзд (точек) на графике, каждая из которых находится от хотя бы одной другой звезды на расстоянии не более R условных единиц. Каждая звезда обязательно принадлежит только одному из кластеров. Истинный центр кластера, или центроид, – это одна из звёзд на графике, сумма расстояний от которой до всех остальных звёзд кластера минимальна. Под расстоянием понимается расстояние Евклида между двумя точками $A(x1, y1)$ и $B(x2, y2)$ на плоскости, которое вычисляется по формуле:

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Аномалиями назовём точки, находящиеся на расстоянии более одной условной единицы от точек кластеров. При расчётах аномалии учитывать не нужно.

Двойная звездная система – это две звезды на расстоянии менее t. При этом других звезд на расстоянии менее t у этих двух звезд быть не должно

В файле Б хранятся данные о звёздах четырех кластеров, где $R = 0.2$, а $t = 0.05$ для каждого кластера. Известно, что количество звёзд не превышает 10 000.

Необходимо в каждом кластере найти пары звезд с максимальным расстоянием между ними.

Решение

Для удобства восприятия разобъем наш код по частям:

1. Создадим функцию *dbscan*(), так как в задаче она нам понадобится несколько раз:

```
def dbscan(a, r):
    cl = [] # Инициализируем список для хранения кластеров
    while a: # Пока есть элементы в входном массиве 'a'
# Создаем новый кластер и добавляем в него первый элемент из 'a'
        cl.append([a.pop(0)])
        for i in cl[-1]: # Проходим по элементам последнего кластера
# Проверяем каждый элемент 'j' в оставшихся элементах 'a'
            for j in a:
# Если расстояние между 'i' и 'j' меньше радиуса 'r'
                if dist(i, j) < r:
                    cl[-1].append(j) # Добавляем 'j' в текущий кластер
# Удаляем 'j' из списка 'a', чтобы не проверять его снова
                    a.remove(j)
    return cl # Возвращаем список кластеров
```

2. Откроем наш файл, считаем его в список a, а затем воспользуемся созданной функцией *dbscan*(), чтобы разбить звезды на кластеры, причем в каждом кластере укажем количество точек, большее 10, чтобы убрать кластеры из аномальных точек:

```
f = open("1_B.txt")
s = f.readline()
a = [list(map(float, i.replace(',', '.').split())) for i in f]
cl = dbscan(a, 0.4)
cl_total = []
for i in cl:
    if len(i) > 10: cl_total.append(i)
```

3. Теперь переходим к поиску двойной звездной системы.

Основная мысль заключается в том, что нам нужно пройти по каждой точке в четырех найденных кластерах и с помощью уже созданной функции *dbscan*() для каждого кластера найти списки звезд, расстояние между которыми менее 0.05.

Далее в каждом кластере нужно оставить только те списки, в которых количество звезд равно двум – то есть только двойные звездные системы.

В конце остается дело за малым: для каждой звездной системы в каждом кластере найти максимальное расстояние между звездами и вывести пару звезд с найденным максимальным расстоянием:

```
t = 0.05  # Устанавливаем радиус для алгоритма DBSCAN
for i in cl_total: # Проходим по каждому элементу в списке cl_total
    found_star = dbscan(i, t) # Применяем алгоритм DBSCAN
    bin_stars = []  # Список для бинарных звездных систем
# Проходим по каждому кластеру, найденному алгоритмом DBSCAN
    for j in found_star:
        if len(j) == 2: # Проверяем, состоит ли кластер из двух звезд
            bin_stars.append(j)  # Добавляем бинарную систему в список
    mx_dist = 0  # Переменная для хранения максимального расстояния
# Список для хранения звездной системы с максимальным расстоянием
    mx_starsys = []
    for j in bin_stars: # Проходим по всем найденным бинарным системам
# Вычисляем расстояние между звездами в бинарной системе
        if dist(j[0], j[1]) > mx_dist:
            mx_dist = dist(j[0], j[1])  # Обновляем максимальное расстояние
# Сохраняем текущую звездную систему как систему с максимальным расстоянием
            mx_starsys = j
# Выводим звездную систему с максимальным расстоянием
    print(mx_starsys)
```

В результате совмещения всех трех частей в полноценный код получаем решение задачи:

```
from math import *
def dbscan(a, r):
    cl = []
    while a:
        cl.append([a.pop(0)])
        for i in cl[-1]:
            for j in a:
                if dist(i, j) < r:
                    cl[-1].append(j)
                    a.remove(j)
    return cl
f = open("1_B.txt"), s = f.readline()
a = [list(map(float, i.replace(',', '.').split())) for i in f]
cl = dbscan(a, 0.4)
cl_total = []
for i in cl:
    if len(i) > 10: cl_total.append(i)
t = 0.05
for i in cl_total:
    found_star = dbscan(i, t)
    bin_stars = [], mx_starsys = []
    for j in found_star:
        if len(j) == 2: bin_stars.append(j)
    mx_dist = 0
    for j in bin_stars:
        if dist(j[0], j[1]) > mx_dist:
            mx_dist = dist(j[0], j[1])
            mx_starsys = j
    print(mx_starsys)
```