Data Science

Assignment3 : Clustering (DB Scan)

2016025532 컴퓨터전공 심수정

1. Summary of Algorithm

clustering 은 주어진 data 들의 특성을 보고 어느 object 들끼리 가장 유사한것인지 알아내는 algorithm 입니다. 이는 크게 distance 기반, density 기반이 있는데, 이번에 과제로 수행할 algorithm 인 db scan 은 density 기반입니다.

Density based clustering 은 기본적으로 2 개의 parameter 가 존재합니다. Eps 와 min points 입니다. Eps 는 maximum radius of their neighborhood 이고, min points 는 minimum number of points in an Eps-neighborhood 입니다. 여기서 eps-neighborhood 는 한 object 로부터 eps 반경 내에 존재하는 object 들을 말합니다.

여기에는 directly density reachable 이라는 개념이 나옵니다. 이는 일정 min points 를 만족하는 한 object 로부터 eps 내에 도달할 수 있는 경우를 말합니다. 또한 density connected 라는 개념도 존재합니다. 이는 directly density reachable 은 아니지만, 다른 density reachable 한 object 들을 통하여 연결될 수 있는 경우를 말합니다.

Db scan 에서 하나의 cluster 는 maximal set of density connected points 를 말합니다.

2. Detailed Description of Code

```
import sys

input_filename = sys.argv[1]
num_of_cluster = int(sys.argv[2])
eps = float(sys.argv[3])
min_points = int(sys.argv[4])

with open(input_filename) as file_:
    objects = file.read().split()
objects = [objects[i_: i + 3] for i in range(0, len(objects), 3)] #objects = [object_id, x_coordinate, y_coordinate]
for i in objects_:
    i[0] = int(i[0])
    i[1] = float(i[1])
    i[2] = float(i[2])
    i.append(None)
```

실행시 넘겨받은 argument 로부터 input filename, number of cluster, maximum radius, minimum number of point in an eps of a point 를 받아와서 각각 input_filename, num_of_cluster, eps, min_points 에 string, int, float, int 의

형태로 넣어줍니다.

그리고 input file 을 열어 파일을 읽어 objects 에 [[object id1, x1, y1], [object id2, x2, y2], ...]의 형태로 저장한 후 이 list item 에서 각각은 int, float, float 의 형태로 바꿉니다. 그리고 cluster label 이 아직 전부 지정되지 않았기 때문에 마지막으로 None 을 추가해줍니다.

최종적인 obejcts 의 형태는[[object id1, x1, y1, label1], [object id2, x2, y2, label2], ...]의 형태가 됩니다.

```
def db_scan(objects):
    count = 0
    for i in objects:
        if i[3] != None:
            continue

        neighbor = range_query(objects, i)

        if len(neighbor) <= min_points:
            i[3] = -1  # noise
            continue


        i[3] = count

        # Seed set S = N \ {p} neighbors to extend
        neighbor.remove(i)

        for j in neighbor:
            if j[3] == -1:
                j[3] = count

            if j[3] != None:
                continue

            j[3] = count
            neighbor2 = range_query(objects, j)

            if len(neighbor2) >= min_points:
                neighbor.extend(neighbor2)

        count = count + 1
        if count > num_of_cluster:
            break
```

db scan 을 통해 clustering 하는 algorithm 입니다. 각각의 DB obejct 을 모두 확인하되 만약 label 이 이미 확정된 object 라면 바로 건너뜁니다.

만약 label 이 확정되지 않은 object 라면 eps 반경의 object 들의 density 가 일정 수를 넘는지 확인한 후 넘지 않는다면 noise 에 해당하는 -1 을 label 에 넣어줍니다.

만약 넘는다면 이 반경에 해당하는 object 들에서 자기 자신을 제거한 후 이웃 object 들에 대해 iteration 을 합니다. 원래 해당 object 가 noise 였다면 이를 현재 label 로 바꿔줍니다. 밑에서 label 이 이미 존재하는지 확인한 후 존재한다면 바로 다음 obejct 를 확인합니다.

이웃 object 에 label 이 존재하지 않는다면 다시 이가 일정 이상의 density 를 가지고 있는지 확인합니다. 일정 이상의 density 를 가지고 있다면 이를 원래 이웃 obejct 에 더 추가하여, 이에 대해서도 이 일을 반복하도록 합니다.

더 이상 추가된 object 가 없고, 모든 이웃 object 에 대해 이 일을 반복했다면 다음 label 에 대해 이를 하도록 count 의 수를 올리고, 만약 이가 정해진 number of cluster 를 벗어난다면 algorithm 을 종료합니다.

```
def range_query (objects, clustered_objects) :
  neighbor = []
  for i in objects :
    if get_distance(i, clustered_objects) <= eps :
      neighbor.append(i)
  return neighbor
```

나와 object DB 의 다른 object 간의 거리를 계산하여 eps 안이라면 나의 이웃 object 목록에 이를 추가하고, 최종적으로 이를 return 해줍니다.

```
def get_distance (new_object, clustered_object) :
  return (((clustered_object[1]-new_object[1])**2+(clustered_object[2]-new_object[2])**2)**0.5)
```

clustering 되고 있는 object 와 DB 에 들어있는 다른 obejct 와의 거리를 계산하여 return 합니다.

```
db_scan(objects)

for i in range(num_of_cluster) :
  with open("input"+input_filename[-5]+"_cluster_"+str(i)+".txt", "w") as fo:
    for j in objects :
      if j[3] == i :
        fo.write(str(j[0])+"\n")

for i in objects :
    print("id : {} label : {}".format(i[0], i[3]))
```

input file 에서 받은 object 들에 대해 db scan 을 수행하고, 이를 각각의 cluster label file 에 써줍니다. 써줄 때는 object id 를 쓰고 줄을 바꾸는 format 으로 써줍니다.

후에는 각각의 object id 와 label 을 출력합니다. 이는 과제 명세와는 관련이 없는 cluster 의 결과를 바로 확인하기 위한 code 입니다.

3. Instruction for Compiling Source Code

이 코드는 python3 을 기반으로 작성되었습니다. 따라서 python3 가 설치되어 있어야 합니다.

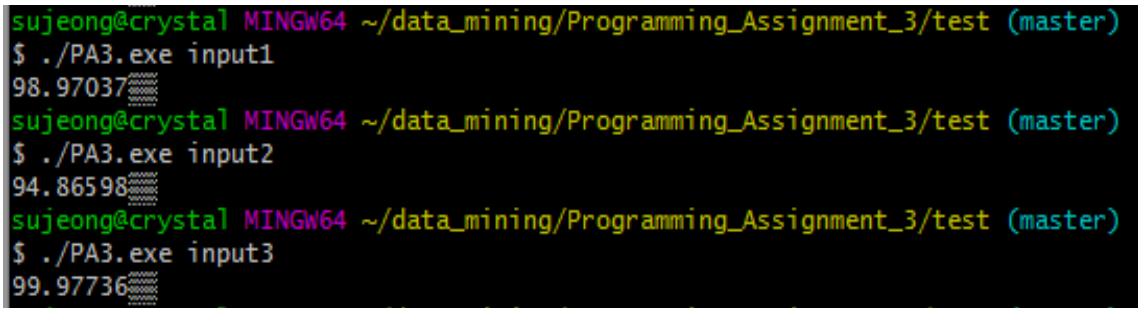
"**python3 clustering.py [input file name] [number of cluster] [maximum radius] [min points]**"의 형태로 complie, 실행하면 됩니다.

4. Other Sepcification of Implementation and Testing

각각의 object 는 [object id, x position, y position, cluster label] 형태의 list 를 가지고 있습니다.

또한, 데이터가 많아질수록 program 의 실행시간이 다소 길어집니다. 무한히 도는 loop 는 아니기에 program 이 정상적으로 끝날 때까지 기다리면 clustering 의 결과를 이상 없이 볼 수 있습니다.

test 결과는 아래와 같이 input1 에 대해서는 약 99%, input 2 에 대해서는 약 95%, input3 에 대해서는 약 100%에 달하는 성능을 내는 것을 확인할 수 있습니다.


sujeong@crystal MINGW64 ~/data_mining/Programming_Assignment_3/test (master)
$ ./PA3.exe input1
98.97037
sujeong@crystal MINGW64 ~/data_mining/Programming_Assignment_3/test (master)
$ ./PA3.exe input2
94.86598
sujeong@crystal MINGW64 ~/data_mining/Programming_Assignment_3/test (master)
$ ./PA3.exe input3
99.97736