Assignment 3: Clustering

2019094511 김준표

1. Summary of algorithm

해당 과제에서는 DBSCAN 방식을 이용하여 clustering을 수행한다.

DBSCAN:

1. Arbitrary select a point p
2. Retrieve all points which are density-reachable from p w.r.t. ε and *MinPts*
3. If p is a core point, a cluster is formed
4. If p is a border point, no points are density-reachable from p, so DBSCAN visits the next point of the database
5. Continue the process until all of the points have been processed

이는 위와 같이 Database 안의 임의의 점에서 인자로 받는 eps와 min pts를 기준으로 density reachable한 점을 찾아 cluster를 형성하는 과정을 통해 구현한다. 타겟 점이 core point라면 cluster를 형성하고, 타겟 점이 border point라면 density reachable한 점이 없으므로 다음 타겟에 대해서 cluster 생성 작업을 수행한다. 이 과정을 모든 점들에 대해서 확인할 때까지 수행한다.

2. Code description

Point class는 각 점마다 id와 x좌표, y좌표를 저장하기 위한 구조체이다.

```
6   class Point:
7       def __init__(self, id, x, y):
8           self.id = int(id)
9           self.x = float(x)
10          self.y = float(y)
11          self.label = None
12          ### label ###
13          # None: not classified
14          # 0: Noise
15          # 1 ~ n : cluster id
```

main 함수에서는 input으로 입력 받는 데이터를 Point class에 맞게 저장하고 이를 데이터에 추가

하여 clustering 함수를 호출한다.

Clustering 함수는 data에 저장된 각 object를 확인하며 classified 되어 있지 않은 object에 대해 해당 object의 neighbor를 찾고 neighbor의 수가 min_pts 이상이라면 cluster를 expand 해준다. 이후 각 cluster에 속한 object를 정리하여 cluster list를 생성한 뒤 리턴해준다.

```
34  def clustering(data, n, eps, min_pts):
35      cluster_id = 1
36
37      for obj in data:
38          # Skip classified object
39          if (obj.label is not None):
40              continue
41
42          # For non-classified object
43          # Get neighbors of the object
44          neighbors = get_neighbors(data, eps, obj)
45          # Cannot make cluster
46          if (len(neighbors) < min_pts):
47              obj.label = 0
48              continue
49
50          # Expand Cluster
51          obj.label = cluster_id
52          expand(data, neighbors, cluster_id, eps, min_pts)
53          cluster_id += 1
54
55      cluster_list = [[] for _ in range(0, cluster_id)]
56      for obj in data:
57          if (obj.label == 0):
58              continue
59          cluster_list[obj.label - 1].append(obj.id)
60      cluster_list = cluster_list[:n]
61      return cluster_list
```

Get Neighbor함수는 전체 data의 모든 object에 대해서 타겟 object와의 거리가 eps 이내인 애들을 찾아준다.

```
20  def get_neighbors(data, eps, obj):
21      return [o for o in data if (o != obj and get_distance(o,obj) <= eps)]
```

Get Distance 함수는 두 좌표 사이의 거리를 계산해준다.

```
17  def get_distance(a: Point, b: Point):
18      return float(math.sqrt(math.pow(a.x-b.x, 2) + math.pow(a.y-b.y, 2)))
```

Expand 함수는 특정 object의 neighbor들에 대해서 get neighbor를 실시하여 density reachable한 점들을 찾아준다.

```python
def expand(data, neighbors, cluster_id, eps, min_pts):
    for obj in neighbors:
        if (obj.label == 0):
            obj.label = cluster_id

        if (obj.label == None):
            obj.label = cluster_id
            next_neighbors = get_neighbors(data, eps, obj)
            if (len(next_neighbors) >= min_pts):
                neighbors.extend(next_neighbors)
```

3. Compiling method & Result Capture

실행환경:

OS: Windows 10

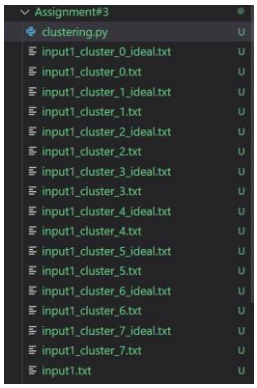
Python version: Python 3.8.3

실행방법:

```
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#3> python clustering.py input1.txt 8 15 22
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#3> ./PA3.exe input1
98.90826점
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#3> python clustering.py input2.txt 5 2 7
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#3> ./PA3.exe input2
94.60035점
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#3> python clustering.py input3.txt 4 5 5
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#3> ./PA3.exe input3
99.97736점
PS C:\Users\pyo99\CSE\DS\2022_ite4005_2019094511\Assignment#3>
```

python clustering.py [input_name.txt] [n] [eps] [min_pts]

위의 실행 명령어를 입력하면 아래와 같이 output file이 생성되는 것을 확인할 수 있다.



이후 test program인 PA3.exe를 통해 test하면 위의 예시처럼 테스트 결과값을 확인할 수 있다.