DBSCAN - 2016025950 정재용

알고리즘에 사용된 Structure

Label -Enum

```
6   class Label(Enum):
7       UNASSIGN = auto()
8       ASSIGN = auto()
9       NOISE = auto()
10
```

- 포인트의 상태를 나타내는 열거형
- UNASSIGN - 아직 군집이 정해지지 않은 포인트
- ASSIGN - 군집이 정해진 포인트
- NOISE - 노이즈 포인트

PointObj 클래스

```
12  class PointObj():
13      def __init__(self, p):
14          self.oid = p[1]
15          self.x = p[2]
16          self.y = p[3]
17          self.label = Label.UNASSIGN
18
19      def _dist(self, q): # euclidean distance with point p
20          return ((self.x - q.x) ** 2 + (self.y - q.y) ** 2) ** 0.5
21
22      def scanNeighbors(self, X, eps):
23          neighbors = []
24          for q in X:
25              if q != self and self._dist(q) <= eps:
26                  neighbors.append(q)
27
28          return neighbors
29
30      def pushCluster(self, clusterSet, k):
31          self.label = Label.ASSIGN
32          if k in clusterSet:
33              clusterSet[k].append(self.oid)
34          else:
35              clusterSet[k] = [self.oid]
```

- 포인트를 나타내는 객체

Variables

- oid - 각 포인트를 구별하는 object id
- x, y - 포인트의 x, y 좌표
- label - 포인트의 현재 상태를 나타내는 라벨

Methods

- def _dist (self, q) - 인자로 받은 포인트 q와의 유클리드 거리를 계산해준다.
- def scanNeighbors (self, X, eps) - 전체 포인트 집합 X 안에 있는 포인트들 중에서, eps 범위 안에 있는 포인트들을 찾아 리턴해준다.
- def pushCluster(self, clusterSet, k) - 포인트의 상태를 ASSIGN으로 설정하고 k번째 클러스터에 포인터를 추가한다.

DBSCAN 알고리즘

```
37  def dbscan(X, eps, minpts, clusterSet):
38      k = -1
39      for p in X:
40          if p.label == Label.UNASSIGN: # no visited
41              neighbors = p.scanNeighbors(X, eps) # get neighbors in eps
42
43              if len(neighbors) >= minpts: # this point is core
44                  k += 1
45                  p.pushCluster(clusterSet, k)
46
47                  queue = deque(neighbors)
48                  while queue:
49                      n = queue.popleft()
50
51                      if n.label != Label.ASSIGN:
52                          n.pushCluster(clusterSet, k)
53
54                          neighbors = n.scanNeighbors(X, eps)
55                          if len(neighbors) >= minpts: # this neighbor is also core
56                              queue.extend(neighbors)
57              else:
58                  p.label = Label.NOISE
59
```

- 포인트 배열 X를 순회하면서 아직 클러스터가 부여되지 않은 점을 찾는다.
- 포인트 p에 대해서 eps 범위 내의 이웃 점들을 찾는다.
  - 범위 안의 이웃 수가 minpts 보다 작으면 일단 노이즈로 분류해놓는다.

    - 범위 안의 이웃 수가 minpts 를 만족하면 포인트 p는 core point이므로 새로운 클러스터를 생성하고 p를 추가한다.
- core point p을 비롯한 이웃 점들을 모두 큐에 넣고 연쇄적으로 클러스터에 들어갈 포인트를 찾는다.
    - 이웃 포인트 중에서 클러스터가 이미 부여됐다면 다른 클러스터임을 나타내는 것이고, 클러스터가 부여되지 않았다면 클러스터 k에 합류시킨다.
    - 해당 포인트에 대해 이웃 포인트 수를 구하고, core point라면 그 이웃들도 다시 큐에 넣는다.
        - core point가 아니라면 border point임을 알 수 있고, 추가 연산을 하지 않는다.

Program Main

```
63  if __name__ == '__main__':
64      # command line arguments
65      input_file_name = sys.argv[1]
66      n = int(sys.argv[2])
67      Eps = int(sys.argv[3])
68      MinPts = int(sys.argv[4])
69
70      # open files
71      input_file = pd.read_csv('./data-3/' + input_file_name, '\t', header=None, names=['object_id', 'x', 'y'])
72
73      X = [PointObj(item) for item in input_file.itertuples()]
74      clusterSet = {}
75      dbscan(X, Eps, MinPts, clusterSet)
76
77      clusters = sorted(clusterSet.values(), key=lambda x:len(x))
78      while len(clusters) > n:
79          del clusters[0]
80
81      for i in range(n):
82          fn = input_file_name.rstrip('.txt')
83          pd.DataFrame(sorted(clusters[i])).to_csv(f'{fn}_cluster_{i}.txt', header=False, index=False)
84
```

- 프로그램을 실행하면, data 파일을 열고 PointObj 객체 타입으로 data들을 저장한다.
- 커맨드 라인에서 받아온 값들을 인자로 넘겨주어 dbscan 알고리즘을 실행한다.
- 만들어진 클러스터들을 크기 순으로 정렬하고, 만약 입력 받은 클러스터의 수 n보다 만들어진 클러스터가 많다면, 크기가 작은 것 순으로 삭제한다.

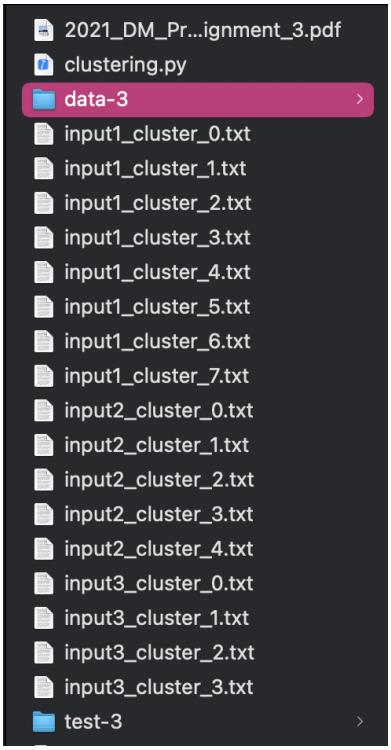
Run Program

```
python3 clustering.py input_file.txt n eps minpts
```

```
(base)  simon  ~/Desktop/data_science/assignment/project3_DBscan  python3 clustering.py input1.txt 8 15 22
```

```
(base)  ✘ simon  ~/Desktop/data_science/assignment/project3_DBscan  python3 clustering.py input2.txt 5 2 7
(base)  simon  ~/Desktop/data_science/assignment/project3_DBscan  python3 clustering.py input3.txt 4 5 5
(base)  simon  ~/Desktop/data_science/assignment/project3_DBscan
```

Result



Test Result