Clustering using DBSCAN Algorithm

2018008331 박민경

summary

DBSCAN algorithm을 구현하여 주어진 data set에 대해 clustering을 수행한다.

Compilation method and environment

Windows(cmd): Python 3.7

execute

```
$ python DBSCAN.py input1.txt 8 15 22
```

argv[1] = input file name

argv[2] = n

argv[3] = Eps

argv[4] = MinPts

check score

```
$ PA3.exe input1
```

argv[1] = input file name

DBSCAN.py

class Point

`__init__`

```
def __init__(self, id, x, y):
    self.id = id
    self.x = x
    self.y = y
    self.label = None
```

init 함수는 Point 객체 생성 시 초기화하는 함수이다. 각 노드는 id, x좌표, y좌표, cluster label을 가진다. 처음에는 cluster가 생성되지 않았음을 가정하여 `None`으로 할당해준다.

class DBSCAN

`__init__`

```
def __init__(self, data, Eps, MinPts):
    self.data = data
    self.Eps = Eps
    self.MinPts = MinPts
```

init 함수는 DBSCAN 모델 객체 생성 시 초기화하는 함수이다. 전체 Point 객체들을 담고 있는 `data`, neighborhood의 maximum radius인 `Eps`, 한 point가 cluster을 형성하기 위한 최소한의 Eps-neighborhood 수인 `MinPts` 을 가진다.

`dist`

```
def dist(self, p1, p2):
    return sqrt((p1.x - p2.x) ** 2 + (p1.y - p2.y) ** 2)
```

dist 함수는 두 point를 인자로 받아 point 사이의 거리를 return한다.

`get_neighbors`

```
def get_neighbors(self, x):
    neighbors = [p for p in self.data if self.dist(x, p) <= self.Eps]
    return neighbors
```

get_neighbors 함수는 한 point를 인자로 받아 Eps-neighborhood를 모두 구해 list로 return한다.

`clustering`

```
def clustering(self):
    cluster_num = -1
    clusters = []
    for point in self.data:
        if point.label is None:
            neighbors = self.get_neighbors(point)
            if (len(neighbors) >= self.MinPts):
                cluster_num += 1
                point.label = cluster_num
                clusters.append([])
                clusters[-1].append(point)
                pts = set(neighbors)
                while pts:
                    pt = pts.pop()
                    if pt.label is None:
                        pt.label = cluster_num
                        clusters[-1].append(pt)
                        neighbors = self.get_neighbors(pt)
```

```
                        if len(neighbors) >= self.MinPts:
                            pts.update(neighbors)

                print("cluster_num %d, points = %d" % (cluster_num, len(clusters[-1])))
    return clusters
```

clustering 함수는 DBSCAN 객체가 가지고 있는 `data`에 대해 clustering을 수행한다. `data`에 있는 모든 점 `point`들에 대해 반복문을 돌며 `point`의 label이 정해져 있지 않을 경우, 즉 cluster에 속해있지 않을 경우 `point`의 Eps-neighborhood를 모두 구해 `neighbors`에 저장한다. 그 수가 `MinPts` 이상일 경우, `point`를 core point로 하여 새로운 cluster을 형성해주기 위해 현재 `point`를 우선적으로 cluster에 넣어준다.

`neighbors`의 모든 point들에 대해 반복문을 돌며 label이 지정되지 않았다면 새로 형성해준 cluster에 넣어주고, 그 point의 `neighbors`를 다시 구하여, 그 `neighbors`도 하나의 cluster를 형성할 수 있다면 (core point라면) `neighbors`를 현재 cluster에 포함될 수 있는 후보에 포함시키는 과정을 반복한다.

`main`

```
if __name__ == "__main__":
    input_file = sys.argv[1]
    n = int(sys.argv[2])
    Eps = int(sys.argv[3])
    MinPts = int(sys.argv[4])

    data = list(map(lambda i: Point(i[0], i[1], i[2]), (np.loadtxt(input_file).tolist())))
    clusters = DBSCAN(data, Eps, MinPts).clustering()
    clusters.sort(key = len, reverse = True)
    if len(clusters) > n:
        print("remove %d clusters" % (len(clusters) - n))
        clusters = clusters[:n]

    cnt = 0
    for cluster in clusters:
        cluster.sort(key=lambda p: p.id)
        output_file = open("input%d_cluster_%d.txt" % (int(input_file[-5]), cnt), 'w')
        for point in cluster:
            output_file.write(str(int(point.id)) + '\n')
        cnt += 1
        output_file.close()
```

command line으로 input file name, n, Eps, MinPts를 입력받고, input file로부터 `data`를 읽어들인다. input file의 각 행은 id, x좌표, y좌표를 가지는 하나의 Point이며, 모든 Point 객체들이 `data`에 list 형태로 저장된다.

DBSCAN 객체를 하나 생성하여 clustering을 수행하고 `clusters`에 저장한 후, cluster의 개수가 n을 초과할 경우 remove하기 위해 `clusters`를 sort하여 가장 members가 적은 cluster들을 순서대로 삭제한다.

마지막으로 출력 형식에 맞춰 n개의 cluster를 각각의 output_file에 쓴다.

result (ex)

```
C:₩Users₩박민경님₩Desktop₩한양대학교₩4-1₩데이터사이언스₩과제₩과제3(DBscan_Algorithm)>python DBSCAN.py input1.txt 8 15 22
cluster_num 0, points = 1481
cluster_num 1, points = 1593
cluster_num 2, points = 1458
cluster_num 3, points = 1124
cluster_num 4, points = 1596
cluster_num 5, points = 177
cluster_num 6, points = 22
cluster_num 7, points = 34
cluster_num 8, points = 34
cluster_num 9, points = 15
cluster_num 10, points = 22
remove 3 clusters

C:₩Users₩박민경님₩Desktop₩한양대학교₩4-1₩데이터사이언스₩과제₩과제3(DBscan_Algorithm)>PA3.exe input1
98.97037점
```

```
C:₩Users₩박민경님₩Desktop₩한양대학교₩4-1₩데이터사이언스₩과제₩과제3(DBscan_Algorithm)>python DBSCAN.py input2.txt 5 2 7
cluster_num 0, points = 640
cluster_num 1, points = 386
cluster_num 2, points = 192
cluster_num 3, points = 489
cluster_num 4, points = 235
cluster_num 5, points = 4
remove 1 clusters

C:₩Users₩박민경님₩Desktop₩한양대학교₩4-1₩데이터사이언스₩과제₩과제3(DBscan_Algorithm)>PA3.exe input2
94.86598점
```

```
C:₩Users₩박민경님₩Desktop₩한양대학교₩4-1₩데이터사이언스₩과제₩과제3(DBscan_Algorithm)>python DBSCAN.py input3.txt 4 5 5
cluster_num 0, points = 500
cluster_num 1, points = 500
cluster_num 2, points = 600
cluster_num 3, points = 499

C:₩Users₩박민경님₩Desktop₩한양대학교₩4-1₩데이터사이언스₩과제₩과제3(DBscan_Algorithm)>PA3.exe input3
99.97736점
```