< Data Science (ITE4005) >

Programming Assignment#3 – DBscan Algorithm

소프트웨어 전공

2013010926 강은석

1. Introduction

: This is an assignment using 'DBScan Algorithm' that learned in clustering class.

DBScan is an algorithm that using Density. If point is have the number of Min points in range of EPS, the point is core point. And if each core point are ranged in Eps, each core point incorporate to one cluster.

2. Summary of My algorithm

:

1. find any core point that satisfies 'Density-based spatial clustering of applications with noise' that is called 'DBSCAN' condition.

2. Using DBSCAN condition, check all point that in core point range.

3. if point(find in 2.) satisfies condition, repeat 2.

4. if there are no core point, end

5. find remained point and incorporate in the nearest cluster.

3. Description of codes

```
def read_input_file(input_file):    # read input#.txt
    input_dict = dict()
    with open(input_file) as f:
        lines = f.readlines()
        for line in lines:
            line = list(map(float, line.strip().split('₩t')))
            input_dict[int(line[0])] = line[1:]
        f.close()
    return input_dict
```

Figure 1 - read file

: This function is read 'input#.txt' file and make dictionary that is consist of key and value. Dictionary key is point id and value is x coordinate, y coordinate.

```
def main(input_file, n, eps, minpts):
    n = int(n)
    eps = int(eps)
    minpts = int(minpts)
    objects = read_input_file('data₩₩'+input_file)
    original_objects = copy.deepcopy(objects)
    cluster_dict = dict()
    core_dict = dict()

    for i in range(n):
        cluster_dict[i] = []
        core_dict[i] = []
    for i in range(n):
        cluster_pt = find_cluster_pt(eps, minpts, objects)  # find any core point
        checked_pts = []
        cluster_elem, core_elem = dbscan(eps, minpts, cluster_pt, checked_pts, objects)  # run dbscan algorithm
        cluster_dict[i] = cluster_elem  # make one cluster
        core_dict[i] = core_elem
        for id in cluster_elem:  # remove core point from all point objects
            objects.pop(id, None)

    cluster_dict = remained_pt_clustering(objects, original_objects, cluster_dict, core_dict)
    write_out_file(n, cluster_dict, input_file)
```

Figure 2 - main function

: main function processed by find any core point and run DBscan algorithm.

And incorporate all remained point to nearest exist cluster.

'cluster_dict' is cluster dictionary. It contain cluster point id.

'core_dict' is used for remove core points that are already checked and included in cluster.

```
def find_cluster_pt(eps, minpts, objects):  # find any core point
    for key1 in objects.keys():
        pt1 = objects[key1]
        in_dist = []
        for key2 in objects.keys():  # find  point that satisfies core point condition
            pt2 = objects[key2]
            dist = math.sqrt((pt1[0]-pt2[0])**2 + (pt1[1]-pt2[1])**2)
            if dist <= eps and dist > 0:
                in_dist.append(pt2)
        if len(in_dist) >= minpts:
            return key1
```

Figure 3 - find core point function

: this function are find any core points that are not included in cluster.

If find any core point return core point information

```
def dbscan(eps, minpts, cluster_pt, checked_pts, objects):  # DBscan algorithm
    cluster_elem = [cluster_pt]
    core_elem = [cluster_pt]
    while len(checked_pts) != len(cluster_elem):  # if check all core point, end algorithm
        in_dist = []
        for key in objects.keys():
            pt = objects[key]
            dist = math.sqrt((pt[0]-objects[cluster_pt][0])**2 + (pt[1]-objects[cluster_pt][1])**2)
            if dist <= eps and dist > 0:
                in_dist.append(key)

        if len(in_dist) >= minpts:
            core_elem.append(cluster_pt)
            no_dup_elem = list(set(in_dist) - set(cluster_elem))
            for p in no_dup_elem:
                cluster_elem.append(p)

        tmp = list(set(cluster_elem)-set(checked_pts))
        cluster_pt = tmp[0]
        checked_pts.append(cluster_pt)
    return cluster_elem, core_elem
```

Figure 4 - DBScan finction

: this function runs DBScan algorithm. It takes cluster point and find points that distance is in eps range. Then, Include points to 'cluster_elem' and Check points satisfy core point condition.

If point is checked, it belonged into 'checked_pts'. End condition is that the number of 'cluster_elem' are equal to the number of 'checked_pts'

```python
# incoperate remain point to nearest exist cluster
def remained_pt_clustering(objects, original_objects, cluster_dict, core_dict):
    clustered_elem = []
    for key in core_dict.keys():
        clustered_elem.extend(cluster_dict[key])

    for key in objects.keys():
        min_dist = 100000000
        min_dist_pt = 0
        pt1 = objects[key]
        for key1 in clustered_elem:
            pt2 = original_objects[key1]
            dist = math.sqrt((pt1[0]-pt2[0])**2 + (pt1[1]-pt2[1])**2)
            if dist > 0 and dist < min_dist:
                min_dist = dist
                min_dist_pt = key1
        for key1 in cluster_dict.keys():
            if min_dist_pt in cluster_dict[key1]:
                cluster_dict[key1].append(key)

    return cluster_dict
```

Figure 5 - remained_point_function

: Incorporate remained points that are not included in any cluster.

So calculate distance to all core point and That remined points are incorporated in the nearest core point in cluster.

```python
def write_out_file(n, cluster_dict, input_file):
    for i in range(n):
        f = open('test₩₩'+input_file[:-4]+'_cluster_'+str(i)+'.txt', "w")
        object_ids = cluster_dict[i]
        for id in object_ids:
            f.write(str(id))
            f.write('₩n')
        f.close()
```

Figure 6 - write function

: write the clustering result in ‘input#_cluster_#.txt’.

4. How to compile code

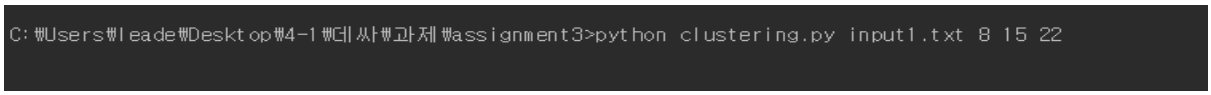

Figure 7 – how to run clustring.py

[python clustering.py inputfile_name the_number_of_clustering eps, minpt]

Input file is in 'data' folder, output file is saved in 'test' folder.

5. Result of test

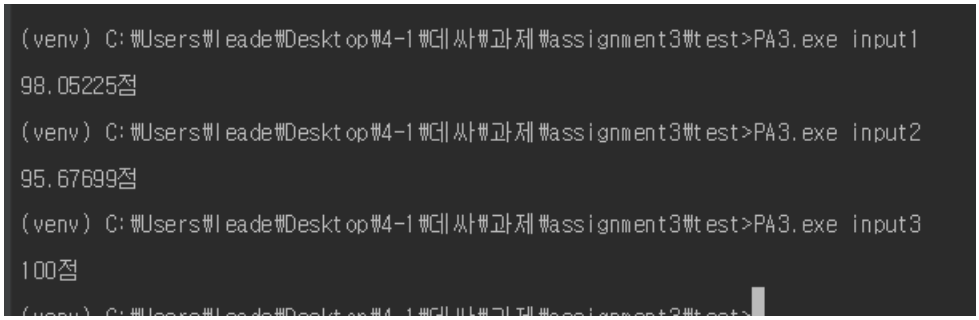

Figure 8 - result of PA3.exe