Data Science – Report #3

I. Environment

i. OS: Mac

ii. Python 3.8.2

II. Compiling the Source Code

i. Example of compiling in Mac

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

III. Summary of an Algorithm

i. My algorithm looks every unvisited point and check them whether the point is a core point or not. If it is a core point, it adds its neighbor into a same cluster and check whether the neighbor is a core point or not. If so, it keeps checking every neighbor point according to the condition of the core point until there is no neighbor. If it is a border point, it checks a next unvisited point.

ii. I referenced the pseudo-code of DBSCAN in the lecture PPT.

IV. Detailed Description of Codes

```
# Open File
def parse():
    obj = []
    with open(sys.argv[1], 'r') as f:
        lines = f.readlines()
        obj = [ list(line.strip().split('\t')) for line in lines
    return obj
```

i. This function is to open the given input file. It parses the line into the list: [id, x coordinate, y coordinate]

```
# Check neighbor
def check_core(obj, objects):
    Eps = int(sys.argv[3])
    x = float(obj[1])
    y = float(obj[2])
    neighbor = []
    for i in range(len(objects)):
        nx = float(objects[i][1])
        ny = float(objects[i][2])
        # If the distance is less than eps, add this point as a neighbor
        if np.sqrt(np.square(nx-x)+np.square(ny-y)) <= Eps:
            neighbor.append(objects[i])
    return neighbor
```

ii. This function is to check that a given object is a core point or not. It calculates the distance with all other points, and if the distance is less that eps, it adds the point as a neighbor.

```
# Create Cluster
def clustering(objects):
    MinPts = int(sys.argv[4])
    visited = [False]*(len(objects))
    cluster = []
    cid = 0
    for obj in objects:
        pid = int(obj[0])
        # If visited, Check a next point
        if visited[pid] == True:
            continue
        # If not visitied, Check whether it is a core point or not
        neighbor = check_core(obj, objects)

        if len(neighbor) >= MinPts:
            # Core point
            # Create a new cluster
            cluster.append([])
            while True:
                # If no more points, stop increasing the cluster
                if len(neighbor) == 0:
                    break
                n_pts = neighbor.pop()
                #print(n_pts)
                n_pid = int(n_pts[0])

                # If visited, check a next point
                if visited[n_pid] == True:
                    continue
                visited[n_pid] = True

                cluster[cid].append(n_pid)

                tmp = check_core(n_pts, objects)
                if len(tmp) >= MinPts:
                    neighbor += tmp

            cid += 1
        else:
            # Border Point
            visited[pid] = True
    # Sort according to cluster's size
    # To easily get n clusters as a result
    cluster.sort(key=len, reverse=True)
    return cluster
```

iii. This function is to create a cluster. It visits every point, and when it visits, it checks whether the point is a core point or not. By using check_core function, it gets all neighbors, and if the number of neighbors is less than MinPts, it means that this point is a border point, so it visits a next point. However, if the number of neighbors is larger than MinPts, it means that this point is a core point so it creates a new cluster. When a new cluster is created, it checks whether the core point's neighbor is a core or not. If so, it adds a new neighbor. It keeps this process until there is not a new point to be added. Finally, it sorted the cluster in a descending order to save the result easily.

```
# Save result
def write(cluster):
    n = int(sys.argv[2])
    idx = sys.argv[1].find(".txt")
    input_file = sys.argv[1][:idx]
    # Get only N elements
    for i in range(n):
        with open(input_file+'_cluster_'+str(i)+'.txt', 'w') as f:
            for obj in cluster[i]:
                f.write(str(obj)+'\n')
```

iv. This function is to save the result. It saves only 'n' clusters which is sorted according to the cluster's size in a descending order.

V. Test Result

i.

```
C:₩Users₩hsy16₩Downloads₩test (1)>PA3.exe input1
98.97277점
C:₩Users₩hsy16₩Downloads₩test (1)>PA3.exe input2
94.86598점
C:₩Users₩hsy16₩Downloads₩test (1)>PA3.exe input3
99.97736점
```