Assignment #3

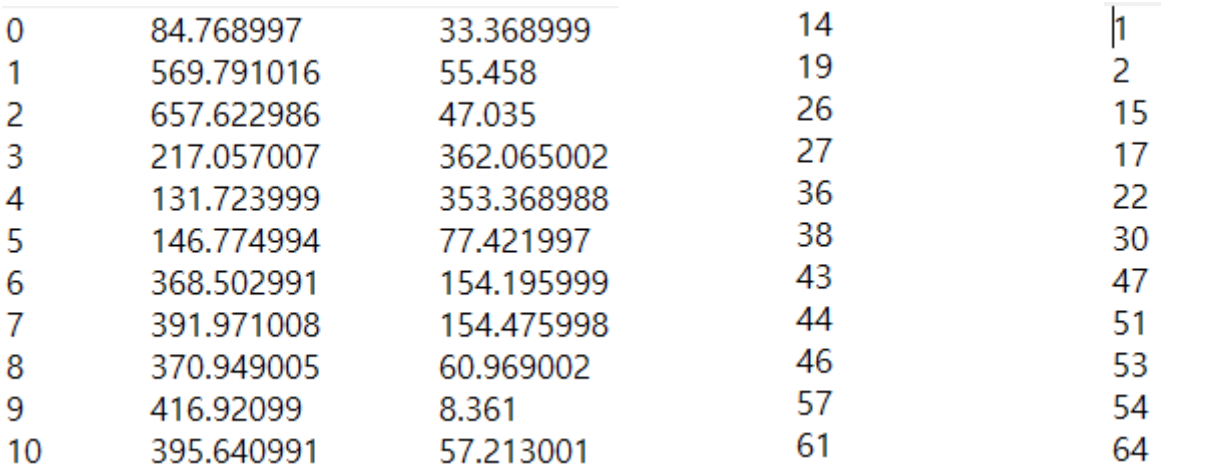
과목명.	데이터사이언스
담 당.	김 상 욱 교수님
제출일.	2021년 06월 05일
공과대학	컴퓨터소프트웨어학부
학 번	2016025969
이 름.	정지훈

HANYANG UNIVERSITY

목차.

1. Summary of algorithm

- Explain Algorithm



0	84.768997	33.368999
1	569.791016	55.458
2	657.622986	47.035
3	217.057007	362.065002
4	131.723999	353.368988
5	146.774994	77.421997
6	368.502991	154.195999
7	391.971008	154.475998
8	370.949005	60.969002
9	416.92099	8.361
10	395.640991	57.213001

Input1.txt

14
19
26
27
36
38
43
44
46
57
61

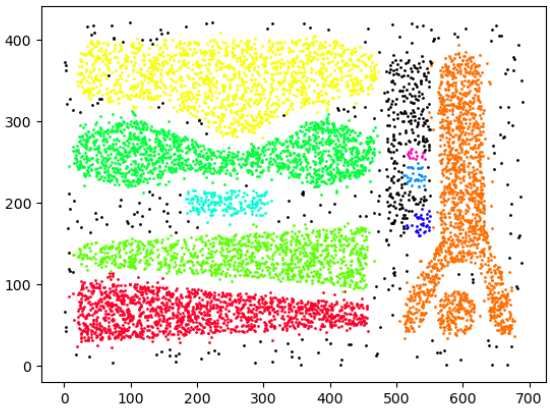
Cluster 1

1
2
15
17
22
30
47
51
53
54
64

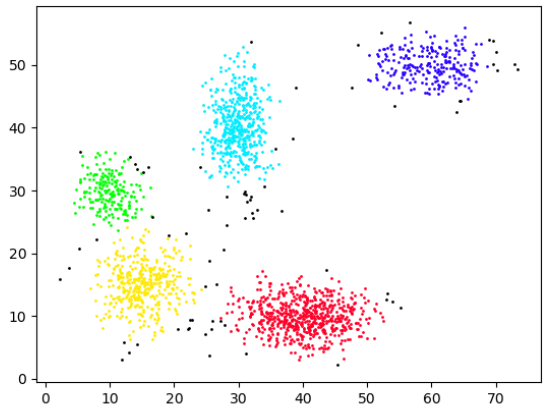
Cluster 2

Input을 통해 Point의 좌표와 Id가 들어오면 DBSCAN 알고리즘을 통해 clustering을 진행하고 cluster에 포함된 점들의 id를 이용해 최종 output을 만드는 과정입니다. DBSCAN은 처음에 임의의 점 P1을 고른 뒤 거기서 주어진 반경 Eps와 반경 안의 최소 점 개수 MinPts를 이용해 모든 가능한 density-reachable인 점을 찾습니다. Density-reachable이란 주어진 P1을 이용해서 해당 점까지 directly density-reachable한 경로를 가지는 점들을 이용해 이어질 때입니다. Directly density-reachable는 임의의 점 P2가 P1로부터 반경 안에 있고 P1의 Eps 안에 MinPts보다 많은 점이 있는 Core Point일 때 성립합니다. 이를 통해 P1이 Core Point이라면 얻은 점들을 Cluster로 형성하고 border point라면 database에서 Cluster 되지 않은 다른 점으로 넘어 갑니다. 이를 반복해 모든 점을 계산한 뒤 종료합니다.

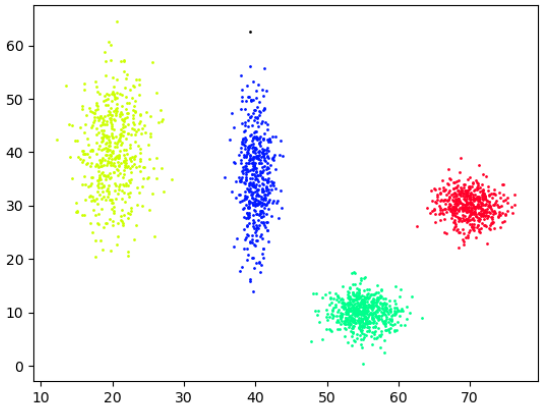
- Result Figure



Input



Input



Input

- Final Score

```
(project) D:\GitHub\ITE4005_HYU_2021\Assignment3\test-3>PA3.exe input1
98.97037점
(project) D:\GitHub\ITE4005_HYU_2021\Assignment3\test-3>PA3.exe input2
94.86598점
(project) D:\GitHub\ITE4005_HYU_2021\Assignment3\test-3>PA3.exe input3
99.97736점
```

2. Detailed description of codes

- Data Read

```
input_file = sys.argv[1]
number = input_file[5:-4]
n = int(sys.argv[2])
eps = int(sys.argv[3])
minPts = int(sys.argv[4])

input_df = pd.read_csv('./data-3/'+input_file,
input = input_df.to_numpy()

n_points = len(input)
classify = [-1] * n_points
```

compile에서 주어지는 최대 Cluster 개수와 Eps, MinPts값을 저장한 후 input file을 읽습니다. 이때 input file을 읽을 때 주어진 object_id를 index로 사용하여 Dataframe을 만듭니다. 만들어진 Dataframe을 이용해 총 점의 개수와 그를 이용해 각 점이 무슨 Cluster에 속하는지 정보를 담는 classify 배열을 만듭니다.

- Distance

```
def dist(x, y):
    return np.linalg.norm(np.array(x)-np.array(y))
```

두 점이 주어지면 Euclidian Distance를 구하는 함수입니다.

- Get Neighbor

```
print("Calculate Neighbor ... ", end="", flush
neighbor_list = defaultdict(list)
for i in range(len(input)):
    for j in range(i+1, len(input)):
        if dist(input[i], input[j]) <= eps:
            neighbor_list[i].append(j)
            neighbor_list[j].append(i)
print("Done")
```

기존에 새롭게 추가되어지는 점이 있을 때마다 이웃을 구해주면 중복되는 시간이 많아지기에 처음 Input을 받고 각 점에 대해 Eps 안에 있는 모든 점들을 구해 준 배열을 만듭니다.

- DBSCAN iterative

```
print("Calculate DBSCAN ... ", end="", flush=True)
for idx in range(len(input)):
    if classify[idx] != -1:
        continue
    if _assign(input, idx, classify, cur_cluster, neighbor_list):
        cur_cluster += 1
print("Done")
```

Cluster를 시작하는 점을 선택합니다. 이때 Core Point나 Outlier라고 Assign 되지 않은 점을 선택합니다.

- Assign Cluster

```
def _assign(array, cidx, classify, cur, ndict):
    global minPts

    nlist = ndict[cidx]
    if len(nlist) < minPts-1:
        classify[cidx] = None
        return False

    classify[cidx] = cur
    while nlist:
        nidx = nlist.pop()

        if classify[nidx] == None:
            classify[nidx] = cur
        elif classify[nidx] == -1:
            classify[nidx] = cur
            pnew_neighbor = ndict[nidx]

            if len(pnew_neighbor) >= minPts-1:
                nlist = list(set(nlist) | set(pnew_neighbor))

    return True
```

임의로 선택된 Point의 index를 이용해 Point의 이웃 점 집합을 구합니다. 만약 해당 점 포함 집합이 MinPts를 넘지 못한다면 Outlier 또는 Border Point이므로 모르겠다는 None을 할당한 뒤 새로운 Cluster를 만들지 않으므로 False를 반환합니다. 만약 넘었다면 해당 점은 Core Point가 되고 그

점은 Clustering 됩니다 이후 그 점을 통해 얻어지는 directly density-reachable된 모든 점들을 같은 Cluster에 속하게 정해줍니다. 이때 기존 None이었다면 Border Point이므로 Assign만 해주고 -1이었다면 새로운 점의 정보이므로 그 점을 기준으로 다시 directly density reachable을 neighbor list에 넣어 줌으로써 시작 점을 통한 모든 density-reachable 점들을 Assign합니다. 이 때 시간을 줄이기 위해 새로운 Neighbor 배열을 만들 때 중복을 제거함으로써 연산을 줄입니다.

- Plot Cluster

```
unique_labels = set(classify)
print(f"{len(unique_labels)} clusters created")
colors = [plt.cm.gist_rainbow(each) for each in np.linspace(0, 1, len(unique_labels))]

for cluster_idx, col in zip(unique_labels, colors):
    if cluster_idx == None:
        col = [0, 0, 0, 1]
    class_mask = list(filter(lambda x : classify[x] == cluster_idx, range(n_points)))
    plt.plot(input_df.values[class_mask][:, 0], input_df.values[class_mask][:, 1],
        'o', markerfacecolor=tuple(col), markeredgecolor=tuple(col),
        markersize=1)
plt.show()
```

DBSCAN을 통해 구한 Cluster의 결과를 확인하기 위해 해당 점들의 좌표와 classify에 담긴 cluster 정보를 합쳐 Cluster Figure를 얻습니다.

- Data Write

```
print("Make Result File ... ", end="", flush=True)
path = "./result"
if not os.path.isdir(path):
    os.mkdir(path)
cnt = 0
for cls, num in counter:
    if cls == None:
        continue
    if cnt >= n:
        break
    li = list(filter(lambda x : classify[x] == cls, range(n_points)))
    li_df = pd.DataFrame(li)
    li_df.to_csv(path+f'/input{number}_cluster_{cnt}.txt', index=False,header=False)
    cnt += 1
print("Done")
```

Classify 배열의 결과를 통해 Cluster에 포함된 점의 개수가 많은 순으로 Output file을 만듭니다.

3. Instructions for compiling source codes at TA's computer

```
(project) D:\GitHub\ITE4005_HYU_2021\Assignment3>python clustering.py input2.txt 5 2 7
Calculate Neighbor ... Done
Calculate DBSCAN ... Done
7 clusters created
[(0, 640), (3, 489), (1, 386), (4, 235), (2, 192), (None, 54), (5, 4)]
Make Result File ... Done
```

주어진 clustering.py 파일을 python을 통해 compile 하면서 이때 Input 파일과 cluster 개수, Eps, MinPts를 차례대로 입력해야 합니다. 파일은 상대경로를 통해 받으므로 해당 python파일이 위치한 곳에 있는 data-3 folder 안에 input.txt가 있어야 합니다. 또한 Result File은 python 파일이 위치한 경로에 result folder가 없다면 생기는 folder 안에 생기게 됩니다.

4. Any other specification of implementation and testing

Python version은 3.8.10이며 numpy의 버전은 1.19.2 pandas의 버전은 1.2.4, 그리고 matplotlib의 버전은 3.3.4 입니다.