DBSCAN 알고리즘 구현

컴퓨터소프트웨어학부

2016025996

조성진

Assignment3 는 DBSCAN 알고리즘을 구현하는 과제로, 기존 과제까지 C++에서 진행하였지만 이번에는 python 으로 구현하였다.

우선 사용한 모듈은 다음과 같다.

```
import numpy as np
from matplotlib import pyplot as plt
import sys
```

그리고 파일을 터미널에서 입력을 넘기는 방법을

```
input_path = sys.argv[1]
input_number = int(input_path[5:-4])
with open(input_path, "r") as f:
    total = f.readlines()
    for line in total:
        idx, x, y = line.strip('\n').split('\t')
        data.append([idx, x, y])
```

다음과 같은 방식을 사용해서 데이터를 생성하였다. 여기서 input_number 는 input_path[5:-4] 의 형태로 표현하였는데, 인풋파일의 이름이 input1.txt 와 같은 형태이므로 숫자를 가져오게 했다.

알고리즘 구현에 관련하여 visited 라는 int 배열은 0 의 값으로 미방문 상태, 1 은 방문했음, 2 는 방문했고 노이즈임을 표시하기 위하여 사용했다.

DBSCAN 알고리즘은 다음과 같다.

```
clusters = np.zeros(data.shape[0], dtype=np.int32)
noises = np.zeros(data.shape[0], dtype=bool)
end_condition = np.ones(data.shape[0], dtype=bool)
cluster_idx = 0
while not np.array_equal(end_condition, visited != 0):# and cluster_idx < n_clusters:
    next = getUnvisitedObject(visited)
    distances = getDistances(next)
    adj_idx = getAdjacentIndex(distances)
    mask = distances[adj_idx]
#     print(mask.shape[0]) # adj_idx -> distance가 radius보다 작은 집합, .shape -> 갯수
#     print("next:",next," and adj count:", mask.shape[0])
    if mask.shape[0] >= minPts:
        cluster_idx += 1
        clusters[next] = cluster_idx
        neighbors = list(adj_idx)
        while neighbors:
            idx = neighbors.pop()
            if visited[idx] == 0:
                visited[idx] = 1
                check_distances = getDistances(idx)
                check_adj_idx = getAdjacentIndex(check_distances)
                check_mask = check_distances[check_adj_idx]
                if check_mask.shape[0] >= minPts:
                    for check_idx in check_adj_idx:
                        if visited[check_idx] == 0:
                            neighbors.append(check_idx)
                if clusters[idx] == 0:
                    clusters[idx] = cluster_idx
    else:
        visited[next] = 2
```

우선 랜덤하게 하나의 노드를 구해 거리를 계산하고 인접한 노드의 갯수를 파악해 그 갯수가 minPts 이상인 경우 neighbor 에 추가하고 그 이하이면 noise 로 처리한다. 그리고 neighbor 이 empty 가 되기 전까지 core Point 들은 eps 이내의 노드들을 neighbor 에 추가하고, neighbor 들의 클러스터가 정해져있지 않으면 클러스터에 추가한다.

여기서 cluster 의 넘버가 0 인경우 아직 클러스터가 정해져 있지 않음을 의미한다.

```
def getUnvisitedObject(visited):
    while True:
        ret = np.random.randint(0, visited.shape[0])
        if visited[ret] == 0:
            visited[ret] = 1
            return ret
```

이 함수는 DBSCAN 알고리즘에서 랜덤하게 하나의 인덱스를 추출할 때 사용하는 함수이다.

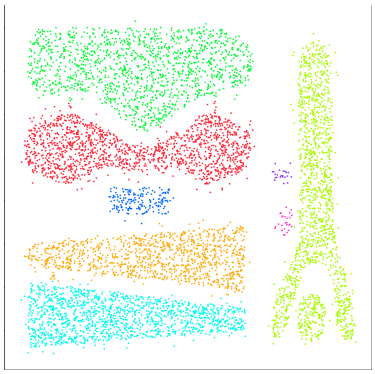
그 이후의 코드 부분들은 Descending order 로 정렬해 가장 노드를 많이 가지는 클러스터 순서대로 적어 정답 파일을 생성하는 코드이다.

```
total_labels = set(clusters)
cluster_size = np.zeros(len(total_labels), dtype=np.int32)
for k in range(len(total_labels)):
    if k == 0:
        continue
    cluster_size[k] = np.where(clusters==k)[0].shape[0]

print(cluster_size)
size = n_clusters
chosen_clusters_idx = np.where(cluster_size > np.sort(cluster_size)[::-1][size])[0]
hash_table = dict()
for index, number in enumerate(chosen_clusters_idx):
    hash_table[number] = index

print(chosen_clusters_idx)
for idx in range(len(clusters)):
    if clusters[idx] not in chosen_clusters_idx:
        clusters[idx] = -1

unique_labels = set(clusters[clusters >= 0])
colors = [plt.cm.gist_rainbow(each)
        for each in np.linspace(0, 1, len(unique_labels))]

print(cluster_size)
```

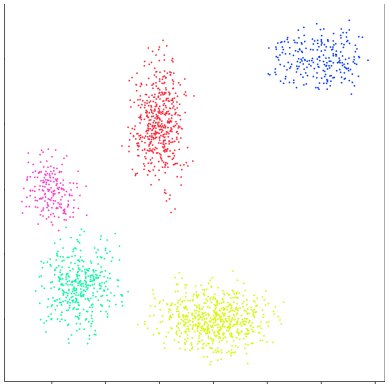
```
plt.figure(figsize=[8, 8])
for cluster_index, col in zip(unique_labels, colors):
    print("cl idx:", cluster_index)
    if cluster_index == -1:
        col = [0, 0, 0, 1]
    class_mask = np.where(clusters == cluster_index)
    sorted_idx = sorted(class_mask[0])
    # print(sorted_idx)
    with open("input{}_cluster_{}.txt".format(input_number, hash_table[cluster_index]), "w") as f:
        for index in sorted_idx:
            f.write("{}\n".format(index))
        plt.plot(data[class_mask][:, 1],
             data[class_mask][:, 2],
            'o', markerfacecolor=tuple(col), markeredgecolor=tuple(col),
            markersize=1)
plt.show()
```

실행결과는 다음과 같다.

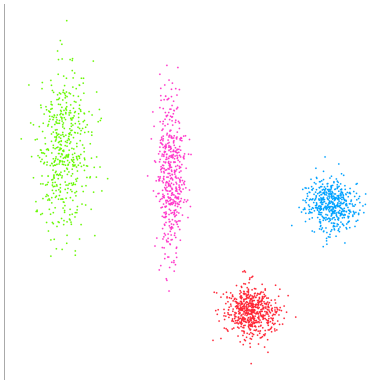
<input1.txt>



<input2.txt>



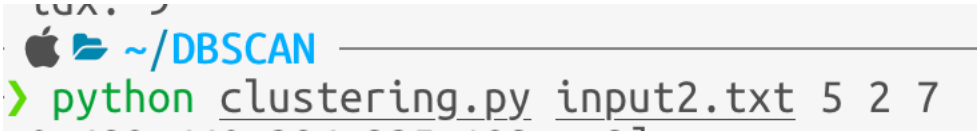
<input3.txt>



그리고 test program 의 score 은 다음과 같다.

```
root@fa0a3e3773b9:/test/DBSCAN/test-3# mono PA3.exe input1
98.98281?root@fa0a3e3773b9:/test/DBSCAN/test-3# mono PA3.exe input2
94.86598?root@fa0a3e3773b9:/test/DBSCAN/test-3# mono PA3.exe input3
99.97736?root@fa0a3e3773b9:/test/DBSCAN/test-3#
```

프로그램의 실행 방법은 파이썬으로 구현했기 때문에



다음과 같은 방법으로 실행할 수 있다.

사용한 python 버전은 3.7.10 이고 os 는 Big Sur 11.4 이다.