assignment3

Summary Of Algorithm

1. 어떠한 cluster에도 할당되지 않은 core point 한 개를 무작위로 선택
2. 선택된 core point에 cluster를 부여
3. 선택된 core point의 neighbor points를 구한다.
4. neighbor points 중 아직 cluster에 포함되지 않은 neighbor point를 선택
    - 4-a : 해당 neighbor가 core point라면 다시 해당 neighbor의 neighbor 중 cluster에 포함되지 않은 neighbor point를 선택 후 2번부터 반복
    - 4-b : 해당 neighbor가 border point라면 cluster만 할당한 뒤 다음 neighbor로 이동
5. 4번에서 더 이상 이동할 neighbor point가 없다면 1번부터 다시 반복
6. 이렇게 구한 cluster 중에서 포함한 point의 수가 많은 순으로 n개의 cluster를 선택

Detailed Description

dbscan

실제 clustering을 수행하는 함수

input

- **objs**(np.array) : 전체 object list
- **labels**(np.array) : 각 object가 속한 cluster를 저장한 list(초깃값은 모두 outlier로 설정)
- **core_points**(np.array) : 각 object가 core points인지를 나타낸 list
- **neighbor_sparse_matrix**(np.array) : 각 object가 서로 neighbor인지를 one-hot vector로 표현한 matrix

output

labels에 cluster 정보를 update

code

```
def dbscan(objs:np.array, labels:np.array, core_points:np.array, neighbor_sparse_matrix:np.array):
    cluster_num=0
    q = deque([])

    # object를 순서대로 탐색
    for i in range(len(objs)):
        obj = objs[i]
        # 1. 어떠한 cluster에도 할당되지 않은 core point 한 개를 무작위로 선택
        if labels[obj] !=-1 or not core_points[obj]:
            continue

        while True:
            if labels[obj] == -1:
                # 2. 선택된 core point에 cluster를 부여
                labels[obj] = cluster_num
                # 3. 선택된 core point의 neighbor points를 구한다.
                # 4-a : 해당 neighbor가 core point라면 다시 해당 neighbor의 neighbor 중 cluster에 포함되지 않은 neighbor point를 선택
                # 4-b : 해당 neighbor가 border point라면 cluster만 할당한 뒤 다음 neighbor로 이동
                if core_points[obj]:
                    neighbors = objs[neighbor_sparse_matrix[obj]]
                    # 4. neighbor points 중 아직 cluster에 포함되지 않은 neighbor point를 선택
                    q.extend(neighbors[labels[neighbors]==-1].tolist())

            # 5. 더 이상 이동할 neighbor가 없으면 다시 1번으로 이동해 cluster에 할당되지 않은 core point를 무작위로 선택
```

```
            if len(q)==0:
                break

            obj = q.popleft()

        cluster_num+=1

    return
```

get_neighbor_sparse_matrix

데이터를 입력받아 neighbor sparse matrix를 반환하는 함수

input

- **data**(np.array) : (n_data, n_feature) 형태로 이루어진 numpy array matrix
- **eps**(int,float) : 각 data pair가 서로 neighbor인지를 판단하기 위한 거리 기준

output

- **neighbor_sparse_matrix**(np.array) : (n_data,n_data) 형태로 이루어져, 각 point끼리 neighbor인지 여부를 one-hot vector로 표현한 matrix

code

```
def get_neighbor_sparse_matrix(data:np.array,eps:Union[int,float])->np.array:
    return pairwise_distances(data,data)<eps
```

selecting cluster & save

dbscan으로 얻은 cluster 중 포함한 **point의 수가 많은 순**으로 n개의 cluster를 선택 후 저장

selecting cluster

```
selected_cluster = list(Counter(labels[labels!=-1]).keys())[:N]
```

save

```
save_num=0
  for cluster_num in selected_cluster:
      OUTPUT_NAME = '_'.join([INPUT_NAME.split('.')[0],"cluster",str(save_num)]) +'.txt'
      OUTPUT_PATH = os.path.join(OUTPUT_ROOT,OUTPUT_NAME)

      cluster_objs = objs[(labels==cluster_num)]

      with open(OUTPUT_PATH,'w') as f:
          f.write('\n'.join(list(map(str,cluster_objs.tolist()))))

      save_num+=1
```

main

```
def main(args:argparse.Namespace):
    INPUT_ROOT = "./data-2"
    INPUT_NAME = args.input_name
    INPUT_PATH = os.path.join(INPUT_ROOT,INPUT_NAME)
    OUTPUT_ROOT = "./test-2"

    N = args.n
    EPS = args.eps
    MinPtr = args.minptr

    objs,points = data_read(INPUT_PATH) # 데이터 읽기
    neighbor_sparse_matrix = get_neighbor_sparse_matrix(points,EPS) # neighbor_sparse_matrix 생성

    n_neighbors = neighbor_sparse_matrix.sum(axis=1) # 각 point의 neighbor 수 계산
    core_points = n_neighbors >= MinPtr # 각 point가 core point인지 아닌지를 계산
    labels = np.full_like(objs,-1) # 초기 모든 point를 outlier로 표시

    dbscan(objs,labels,core_points,neighbor_sparse_matrix) # dbscan이 완료되면 labels에 각 point가 속한 cluster number가 저장됨

    selected_cluster = list(Counter(labels[labels!=-1]).keys())[:N] # 전체 labels에 나타난 cluster 중 포함한 point가 많은 상위 N개 cluster만 저장

    save_num=0
```

```
    for cluster_num in selected_cluster:
        OUTPUT_NAME = '_'.join([INPUT_NAME.split('.')[0],"cluster",str(save_num)]) +'.txt'
        OUTPUT_PATH = os.path.join(OUTPUT_ROOT,OUTPUT_NAME)

        cluster_objs = objs[(labels==cluster_num)]

        with open(OUTPUT_PATH,'w') as f:
            f.write('\n'.join(list(map(str,cluster_objs.tolist()))))

        save_num+=1
```

Execute guide

1. `clustering.py`의 첫 줄에 python 실행 위치를 적어줍니다.

```
1   #!/usr/bin/python3
```

in clustering.py

2. 현재 유저의 `clustering.py` 실행권한을 허용합니다.

```
chmod 755 clustering.py
```

Results

input1 : 98.98035점

input2 : 94.86598점

input3 : 99.97736점