ITE4005 – Assignment3

박준영

1. Summary

본 프로젝트의 목표는 density 기반의 clustering 알고리즘인 DBSCAN 을 이용해 주어진 데이터를 clustering 하는 프로그램의 개발이다. 해당 알고리즘은 어떤 점으로부터 특정 Euclidean distance 이내에 점이 특정 개수만큼 있으면 해당 점들을 하나의 cluster 로 지정하는 방식으로 동작한다. 프로그램 내에서 가장 병목지점이 될 한 점과 다른 점들 과의 거리는 SIMD 등의 계산 가속 지원을 받을 수 있도록 vectorize 하여 연산하였다.

2. Detailed description

main entry

```python
def main(filename: str, n: int, eps: float, min_pts: int):
    data_index, data = load_data(filename)
    clusters = dbscan(data, eps, min_pts)

    clusters = sorted(clusters, key=lambda x: len(x), reverse=True)[:n]

    basename = os.path.basename(filename)
    basename = os.path.splitext(basename)[0]

    for cluster_id, cluster in enumerate(clusters):
        with open(f'{basename}_cluster_{cluster_id}.txt', 'wt') as f:
            f.write('\n'.join([data_index[idx] for idx in cluster]))
```

위 코드는 프로그램의 main entry 이다. 이곳에서 데이터 로딩, dbscan 알고리즘을 수행하는 subroutine 을 각각 부른 뒤, 클러스터를 클러스터의 크기로 정렬하고 결과를 파일에 쓰는 역할을 수행한다.

본 프로그램 내부적으로 각 data point 를 표현하는 것은 zero-base 의 index 이다. 하지만 입력 파일에 적힌 index 를 적기 위하여 입력 파일에 있는 data point 의 정확한 index 를 적고 프로그램 내부의 index 와 연동하여 출력 파일엔 올바른 index 를 적을 수 있도록 하였다.

data loader

```python
def load_data(filename: str):
    data_index = []
    data = []
    with open(filename, 'rt') as f:
        for line in f.readlines():
            idx, x_coord, y_coord = line.split()
            x_coord, y_coord = float(x_coord), float(y_coord)

            data_index.append(idx)
            data.append(np.array([x_coord, y_coord]))

    return data_index, np.array(data)
```

Clustering 을 수행할 데이터를 불러오는 함수로, 파일을 라인별로 읽어와 index 와 x, y 좌표를 구해 각각을 반환한다.

dbscan (part1)

```python
def dbscan(data: np.ndarray, eps: float, min_pts: int):
    status = np.zeros(data.shape[0])

    def _get_neighbor_idx(p_idx):
        distances = np.sqrt(np.sum(np.square(data - data[p_idx]), axis=-1))
        return np.where(distances <= eps)[0].tolist()
```

status 는 각 data point 가 outlier 인지, 아니면 특정 cluster 로 지정이 되었는지를 나타낸다. status 가 0 은 아직 처리가 되지 않음을, -1 은 outlier 임을, 1 이상의 수는 해당 cluster 로 분류됨을 의미한다.

_get_neighbor_idx 는 한 점 p와 모든 data point 간의 Euclidean distance 를 계산하여 기준 거리에 들어오는 점의 목록을 구하는 함수다.

dbscan (part2)

```python
clusters = []
for idx in range(data.shape[0]):
    if status[idx] != 0:
        continue

    neighbors_idx = _get_neighbor_idx(idx)
    if len(neighbors_idx) < min_pts:
        status[idx] = -1 # mark as outlier
```

위 코드는 dbscan 알고리즘의 main loop 로 모든 data point 를 순회하며 처리되지 않은 data point 를 처리하도록 한다. 우선적으로 한 data point 와 가까운 data point 의 수를 구해, 기준보다 낮으면 해당 data point 를 outlier 로 일단은 mark 한다.

dbscan (part3)

```
else:
    new_cluster_id = len(clusters) + 1 # to make 0 as not processed
    new_cluster = [idx]
    status[idx] = new_cluster_id

    neighbors = deque(neighbors_idx)
    while len(neighbors) > 0:
        nb = neighbors.popleft()
        if status[nb] <= 0:
            if status[nb] == 0:
                nb_neighbors_idx = _get_neighbor_idx(nb)
                if len(nb_neighbors_idx) >= min_pts:
                    neighbors += nb_neighbors_idx

            status[nb] = new_cluster_id
            new_cluster.append(nb)

    clusters.append(new_cluster)
```

위에서 이어지는 코드로, 특정 거리 안에 있는 data point 수가 기준을 넘은 경우 새로운 cluster id 를 부여하고, 이웃한 data point 를 큐에 넣고 큐가 빌 때까지 해당 cluster 를 키우는 방식으로 동작한다. cluster 의 생성이 끝났다면 새 cluster 에 속한 점들의 목록으로 표현된 cluster 를 cluster 리스트에 추가한다.

3. Instructions for executing the program

우선 다음 명령을 통해 본 프로젝트를 받는다.

```
git clone https://hconnect.hanyang.ac.kr/2022_ite4005_13026/2022_ite4005_2019064811
```

이후 다음 명령을 통해 본 프로그램의 위치로 이동한다.

```
cd ITE4005/assignment3
```

마지막으로, 다음의 명령을 통해 프로그램을 실행시킨다.

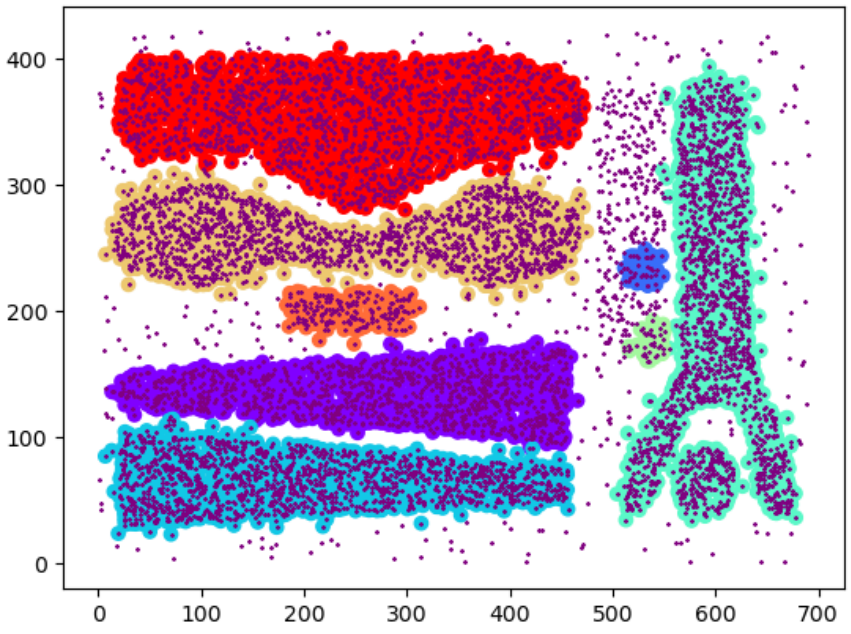
```
python clustering.py <input filename> <n> <eps> <min_pts>
ex) python clustering.py input1.txt 8 15 22
```

4. Additional information

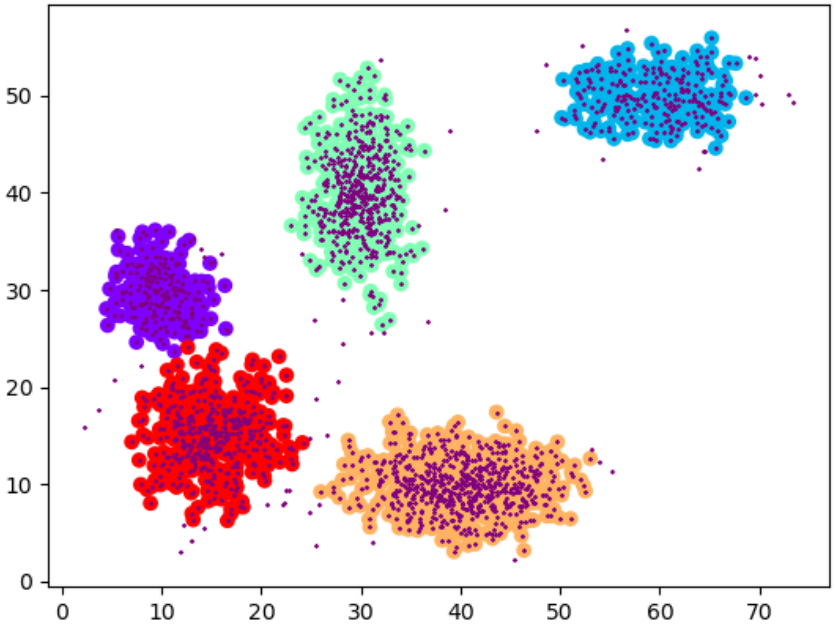
본 프로젝트는 다음의 환경에서 테스트되었다.

- CPU: Intel® Core™ i9-10940X
- RAN: DDR4 32GB×4 (128GB)
- OS: Ubuntu Server 20.04 LTS
- Python 3.8.8
- Numpy 1.20.1

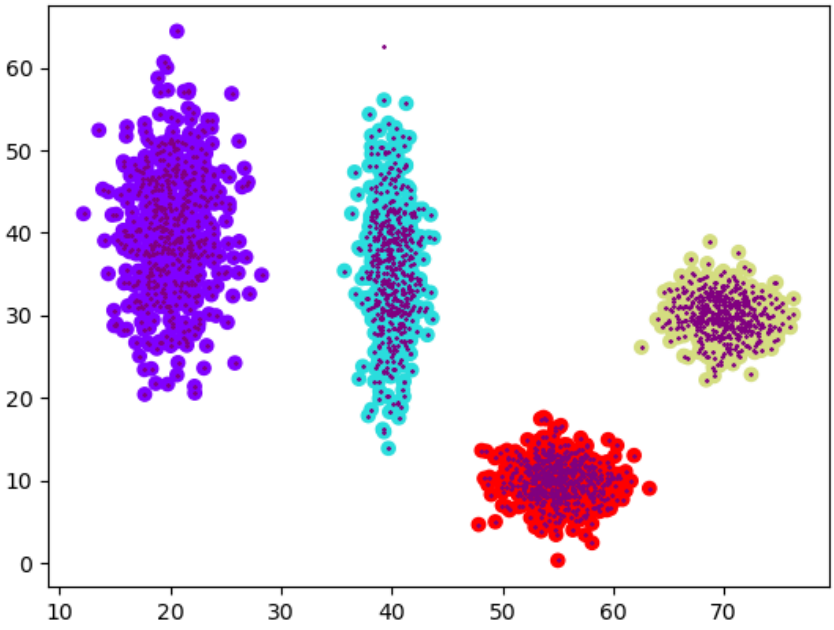
아래는 본 프로그램의 예시 데이터에 대한 clustering 결과이다.



<result of input1>



<result of input2>



<result of input3>