# Report

2016025196

김동규

## 1. 실행 환경 및 실행 방법

환경 : Windows 10, python 3.9.2, numpy1.19.5

실행 방법 : assignment3.py (input.txt)

## 2. 알고리즘 요약

임의의 한 점에서 시작하여 eps조건과 minpts조건을 확인한다. 이때, minpts가 넘으면 해당 점에서 클러스터 구성을 시작하며, 연쇄적으로 퍼져나가면서 클러스터를 구성한다. minpts조건이 안 될 경우 추후 다른 점에 의해 density connected가 될 때까지 outlier로 처리한다. Remain points가 남지 않을 때까지 반복한다.

## 3. 코드설명

Cluster_id는 0에서 시작하여 클러스터 집합이 구성될때마다 1씩 증가한다. Remain_DB는 원래 DB상태에서 시작하여 outlier를 포함한 클러스터에 속하지 않는 점들을 제외하며 줄어든다.

크게 2단계의 과정으로 클러스터링한다.

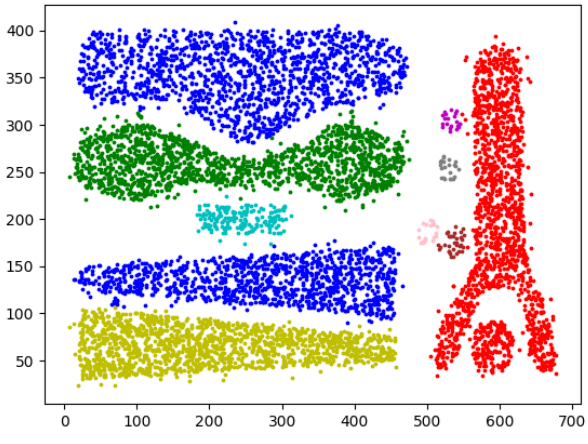
1. Remain_DB에서 임의의 한 점을 선택한다. 해당 점에서 eps와 minpts를 만족할 경우 클러스터를 구성하고 주변 점을 큐에 넣는다.

2. 큐가 비어있을 때까지 한 점을 꺼내 클러스터 집합을 구성한다. 주변 점도 큐에 다시 넣는다.

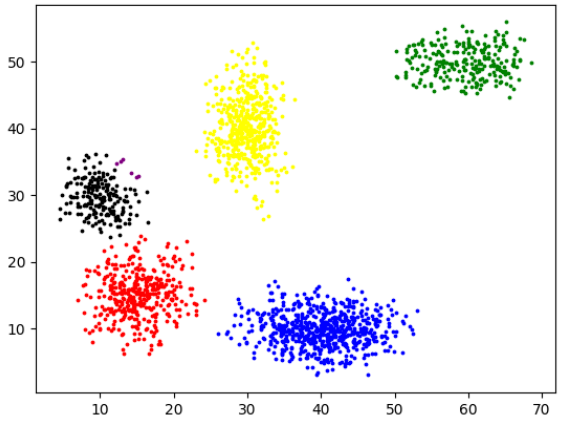
이후 출력결과를 쓸 때는 DB에 저장한 cluster id 값에 따라 별도의 파일로 저장한다.

```
def DBSCAN(DB,n,eps,minpts):
    remain_DB=DB
    cluster_id=0
    idx_list=[i for i in range(0,len(DB))]
    idx_list=np.array(idx_list)
    id=idx_list

    while len(remain_DB)>0:
       # print('-----')
        print(cluster_id)
        print(len(remain_DB))
        next_list=np.empty((0,3))
        rand_idx=random.randint(0,len(remain_DB))
      #  rand_idx=368
        core_point=remain_DB[rand_idx]
        DB_idx=id[rand_idx]
        #print(core_point)
      #  print(rand_idx,DB_idx)
        count,neighbor=count_neighbor(DB,core_point,eps)
        remain_DB=np.delete(remain_DB,rand_idx,axis=0)

        DB[DB_idx,2]=-2
      # print(DB_idx,DB[DB_idx])
       # print('=========')

        if count>=minpts:
            #print('core')
            core_point[2]=cluster_id


            n_data=DB[neighbor,:]
            n_bool=n_data[:,2]<-0.5
            DB[neighbor,2]=cluster_id

            next_list=np.append(next_list,n_data[n_bool],axis=0)

            if len(next_list)==0:
               # print("0state")
                continue
            while len(next_list)>0:
                next_core=next_list[0,:]
                next_list=np.delete(next_list,0,axis=0)
                count,neighbor=count_neighbor(DB,next_core,eps)

                if count>=minpts:
                    n_data=DB[neighbor,:]
                    n_bool=n_data[:,2]<-0.5
                    DB[neighbor,2]=cluster_id
                    next_list=np.append(next_list,n_data[n_bool],axis=0)

            cluster_id+=1


        DB_bool=DB[:,2]==-1
        remain_DB=DB[DB_bool]
        id=idx_list[DB_bool]
```

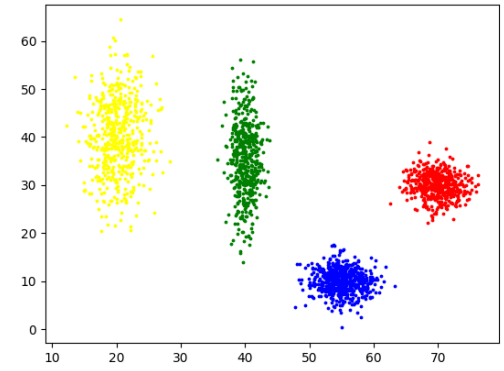
## 4. 실험 결과




input1



input2



input3

결과값의 측정을 위해 matplot으로 클러스터 집합을 출력하는 코드를 작성했다.

```
    DB=file_open(file_name)
    c=[]
    number=11
    i=1
    for n in range(0,number):
        c.append(ideal_open(i,n))
    x_values=DB[:,0]
    y_values=DB[:,1]
    color=['r','b','y','g','b','c','m','w']
    for i in range(0,number):
        plt.scatter(x_values[c[i]],y_values[c[i]],color=color[i%number],s=3)
```

PA3.exe의 값은 25~40점인데 반해 실제적인 클러스터링 결과는 잘 되어있음을 알 수 있었다.