Programming Assignment #3

2018009870 이찬호

1. Summary of your algorithm

DBSCAN class의 method는 아래와 같다.

- countdistance : 한 점이 주어졌을 때, 그 점과 데이터 간의 거리를 계산하고, 주어진 점을 기준으로 반경 안에 얼마나 많은 점이 있는지 판단하는 메소드.
- unionCluster : 새로 생긴 클러스터가, 기존에 존재하는 클러스터와 연결이 될 경우, 그 클러스터를 기존의 클러스터와 합치는 메소드
- plot : 결과를 plot 해주는 메소드
- expanding : countdistance를 통해서, cluter가 확장이 될 수 있는지 판단을 한 다음, 확장이 되면 새로 들어온 점들을 기준으로 다시 cluster가 확장을 할 수 있는지 판단하는 메소드
- checking : 특정 클러스터의 모든 점을 조사하여 더 이상 확장이 불가능하다고 판단하면, 빈 배열을 반환, 확장이 가능하면, 고려해야 할 점들을 반환
- training : expanding 메소드와 checking 메소드를 활용하여 clustering 진행하는 메소드

2. Detailed description of your codes

```
def __init__(self,data,n,eps,minPts):
        self.data = data # cluster 할 데이터,
        self.NumberOfCluster = n # cluster의 개수
        self.eps = eps # cluster를 확인할 반경
        self.minPts = minPts # 반경 안에 있어야 하는 최소한의 점의 개수
        self.cluster = np.zeros(self.data.shape[0]).reshape(self.data.shape[0]) # label 저장 preallocation
        self.checked = np.zeros(self.data.shape[0]).reshape(self.data.shape[0]) #cluster 인지 아닌지 확인한 정보를 저장할 공간 preallocation

def countdistance(self,point)
        # xi - xj, yi - yj를 broadcast를 통해 계산
        minus = self.data[:,1:3] - point
        # 유클리드 거리를 broadcast를 통해 계산
        distance = (minus[:,0]**2 + minus[:,1]**2)**0.5
        # class 생성 시, eps를 설정해 eps 반경 이내에 있는 점들의 좌표 반환
        count = np.less_equal(distance,self.eps)
        # 좌표의 개수와 그 좌표를 반환
        return np.sum(count), count
def unionCluster(self,clusterNumber):
        # 현재 진행되었던 cluster의 label를 가지고 현재 분류된 데이터의 좌표 반환
```

```
        _index = self.cluster == clusterNumber
        # 현재 진행되었던 cluster의 label를 가진 데이터를 가지고 더 확장할 곳이 있는지 조사
        candidate = self.data[_index]
        for i in range(candidate.shape[0]):
            # 현재의 데이터를 가지고 판단, 만약 eps 반경 내에, 서로 다른 class label이 존재하면 1을 반환, 아니면 0을 반환
            (count,index) = self.countdistance(candidate[i,1:3])
            cluster = self.cluster[index]
            if count >= self.minPts:
                for j in range(count):
                    if cluster[j] != clusterNumber:
                        self.cluster[_index] = cluster[j]
                        return 1
            else :
                continue
        return 0
    def expanding(self,candidate,clusterNumber):
        # 특정 점들을 받아드려서, cluster인지 아닌지 확인하는 메소드
        # 들어오는 점의 개수를 row에 저장
        row = candidate.shape[0]
        for i in range(row):
            # 만약 check라는 배열을 확인하여, 1인 경우는 기존에 cluster인지 확인을 했던 점이므로 확인을 하지 않음
            if self.checked[int(candidate[i,0])] == 1:
                continue
            # 그렇지 않은 경우, countdistance 메소드를 통해 반경 안의 점의 개수를 세고 cluster가 될 수 있는지 확인함.
            (count,index) = self.countdistance(candidate[i,1:3])
            # 다음에 확인하지 않도록 check 배열을 1로 설정
            self.checked[int(candidate[i,0])] = 1
            if count >= self.minPts:
                self.cluster[index] = clusterNumber
    def checking(self,clusterNumber):
        # 특정 클러스터의 모든 점을 조사하여 더 이상 확장이 불가능하다고 판단하면, 빈 배열을 반환, 확장 가능하면 고려해야 할 점들을 반환
        # 같은 클러스터라고 분류된 점들을 뽑아냄.
        cluster = (self.cluster == clusterNumber)
        # 만약 cluster가 검사를 하지 않았으면, 검사하지 않은 점들을 반환, 전부 검사를 했으면 0을 반환
        index = np.logical_and(cluster,(np.logical_not(self.checked)))
```

```
        return self.data[index]
    def training(self):
        # expanding 메소드와 checking 메소드를 활용하여 clustering 하는 메소드
        clusterNumber = 1
        while True:
            # 아무 번호나 하나를 뽑음
            i = np.random.randint(0,self.data.shape[0])
            # 만약 cluster인지 확인을 하지 않은 점이라면 검사 시작
            while self.checked[i] != 0:
                i = np.random.randint(0,self.data.shape[0])
            # 검사를 시작했으므로, 그 점을 1이라고 놓고 시작
            self.checked[i] = 1
            # 그 점을 기준으로 주변의 점들을 조사
            (count,index) = self.countdistance(self.data[i,1:3])
            # 기준보다 많은 점들을 가지고 있으면, 확인
            if count >= self.minPts:
                # 만약 반경 안의 모든 점이 label을 가지고 있지 않다면 새로운 label 부여
                if np.sum(self.cluster[index]) == 0:
                    # 현재의 구역에 있는 clsuter는 label을 같게 함.
                    self.cluster[index] = clusterNumber
                     # 현재 구역에는 label이 있지만 cluster인지 확인을 하지 않은 점이 있으므로 그를 활용해서 영역을 넓히기 위해 expanding 메소드 활용.
                    candidate = self.checking(clusterNumber)

                    while candidate.size != 0:
                        self.expanding(candidate,clusterNumber)
                        candidate = self.checking(clusterNumber)
                    # 만약 기존의 클러스터와 현재의 클러스터가 합쳐진다면, 현재의 클러스터의 label은 다시 사용할 수 있으므로 clusterNumber을 감소 시킴.
                    if clusterNumber != 1 and self.unionCluster(clusterNumber):
                        clusterNumber -= 1
                    # 그렇지 않은 경우, cluster가 완료되었다고 출력
                    else :
                        print("cluster",clusterNumber,"is done!")
                    clusterNumber += 1
            # 만약 모든 데이터를 확인하였을 때
            if np.sum(self.checked) == self.data.shape[0]:
                # 각각의 label에 몇 개의 점들이 있는지 확인
                counts = list()
                for j in range(1,clusterNumber+1):
                    counts.append(np.sum(self.cluster == j))
```

```
            # 만약 입력한 cluster의 개수가 원하는 class의 개수보다 많으면, 작은 개수를 가지는 cluster를 제거
            while len(counts) != self.NumberOfCluster:
                minimum = min(counts)
                idx = counts.index(minimum)
                self.cluster[self.cluster == idx + 1] = 0
                counts.pop(idx)
                self.data = self.data[self.cluster != 0]
                self.cluster = self.cluster[self.cluster != 0]
            break
        return self.cluster
```

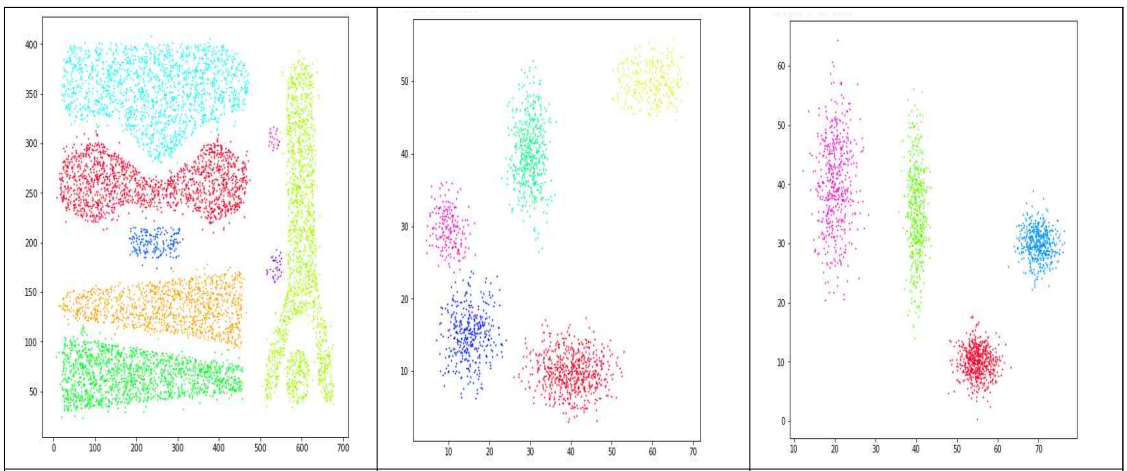
3. Instructions for compiling your source codes at TA's computer

```
C:₩Users₩이찬호₩Documents₩GitHub₩DataScienceProject₩project3>python clustering.py input1.txt 8 15 22
cluster 1 is done!
cluster 2 is done!
cluster 3 is done!
cluster 4 is done!
cluster 5 is done!
cluster 6 is done!
cluster 7 is done!
cluster 8 is done!
cluster 9 is done!
cluster 10 is done!
```

다음의 화면과 같이 python clustering.py input_file_name n eps minPts 순서대로 입력을 하면 된다.

```
C:₩Users₩이찬호₩Documents₩GitHub₩DataScienceProject₩project3>PA3.exe input1
99.00118점
C:₩Users₩이찬호₩Documents₩GitHub₩DataScienceProject₩project3>PA3.exe input2
94.86598점
C:₩Users₩이찬호₩Documents₩GitHub₩DataScienceProject₩project3>PA3.exe input3
99.97736점
C:₩Users₩이찬호₩Documents₩GitHub₩DataScienceProject₩project3>
```

다음과 같이 결과가 나왔으므로 DBSCAN이 작동한 것이라고 할 수 있다.
아래의 결과는 cluster의 결과를 시각화 한 것이다.



input1.txt	input2.txt	input3.txt