데이터 사이언스 Project3 Report

컴퓨터소프트웨어학부

2016024839 박정민

1. 코드 구성

주어진 input file을 pandas dataframe으로 받아서 저장하고, dataframe을 바탕으로 여러 함수를 통해 clustering을 수행하는 DBSCAN class를 만들었습니다.

```
class DBSCAN:

    # Initiate parameters
    def __init__(self, n, Eps, MinPts):
        self.n = n
        self.Eps = Eps
        self.MinPts = MinPts
        self.ClusterIdx = 0
        self.noiseCluster = []
        self.neighCount = []
```

n : 프로그램 실행 파라미터인 cluster 최대 개수입니다.

Eps : 프로그램 실행 파라미터인 cluster에 포함되는 거리입니다.

MinPts : 프로그램 실행 파라미터인 cluster가 되기 위한 최소 포함 점 갯수입니다.

ClusterIdx : 현재 몇 번째의 clustering을 하는 지를 나타내는 변수입니다.

noiseCluster : clustering 완료 후 cluster 개수가 n개보다 많을 시 noise처리할 cluster를 관리할 list입니다.

neighCount : cluster별로 neighbor수를 저장하는 list입니다.

2. Class 내 method

```
# Read input.txt and set parameters
def readData(self, input):
    self.df = pd.read_table(input, sep='\t', header=None, names=['index', 'x', 'y'])
    self.size = len(self.df)
    self.visit = np.full((self.size), False)
    self.noise = np.full((self.size), False)
    self.realm = np.full((self.size), 0)
    self.calDist()
    self.inputNum = input[0:6]
```

readData(self, input)

함수 실행 파라미터로 주어진 input file을 dataframe으로 받아오는 함수입니다. 데이터를 받아온 후 점의 총 개수인 size, 점 방문 여부를 나타내는 visit array, noise인지 여부를 나타내는 noise array, 몇 번 cluster에 포함되는지를 나타내는 realm array를 초기화합니다. 또한 각 점마다의 거리를 미리 계산하는 calDist()함수를 호출하여 size * size matrix로 저장합니다.

```
# Calculate distance from whole points
def calDist(self):
    x = self.df[['x']].values
    y = self.df[['y']].values

    x1, x2 = np.meshgrid(x, x)
    y1, y2 = np.meshgrid(y, y)

    self.dist = np.sqrt((x1 - x2)**2 + (y1 - y2)**2)
```

calDist(self)

각 점에서부터 다른 점까지의 거리를 바로 인덱싱하여 참조할 수 있는 size * size matrix를 계산합니다. 먼저 x좌표를 포함하는 배열과 y좌표를 포함하는 배열을 dataframe으로부터 가져온 후 numpy의 meshgrid 함수를 이용해 size * size크기의 matrix 2개로 확장하는데 하나는 좌표값을 세로로 복사해서 나열하고 다른 하나는 가로로 복사해서 나열합니다. 그 후 확장된 x좌표 matrix 2개와 y좌표 matrix 2개를 서로 빼면 각 점과 다른 점 까지의 x좌표 차, y좌표 차를 나타내는 size * size크기의 matrix를 구할 수 있습니다. 유클리드 거리 공식을 이용하여 거리를 계산합니다.

Ex) (0, 30)에 해당하는 원소가 0번째 point부터 30번째 point까지의 거리.

```python
# Clustering
def cluster(self):
    for i in range(self.size):

        if self.visit[i] == False:
            self.visit[i] = True
            self.neighbor = self.getNeighbor(i)

            if len(self.neighbor) > self.MinPts:
                self.ClusterIdx += 1
                print("Start Clustering", self.ClusterIdx, "th cluster with", len(self.neighbor), "neighbors")
                self.clusterExpand(i)
                self.neighCount.append(len(self.neighbor))

            else:
                self.noise[i] = True

    self.pruneCluster()
    self.df['cluster'] = self.realm
    self.writeFile()
```

cluster(self)

clustering을 수행하는 함수입니다. 주어진 점들에 대해 for문을 돌면서 visit 배열을 참조하여 방문하지 않았으면 방문으로 처리하고 getNeighbor 함수를 통해 해당 점 주변의 조건을 만족하는 점들의 list를 받아옵니다. 그 점들의 개수가 MinPts보다 많으면 cluster 조건을 충족하는 것이므로 cluster를 만들어주고, clusterExpand 함수를 통해 확장합니다. 그 후 pruneCluster함수로 cluster개수가 n개보다 많을 시 pruning을 수행합니다.

```python
# Get Neighbors from idx
def getNeighbor(self, idx):
    ret = []

    for i, temp in enumerate(self.dist[idx]):
        if temp < self.Eps:
            ret.append(i)

    return ret
```

getNeighbor(self, idx)

Eps보다 거리가 작은 neighbor list를 return합니다. 점과 점 사이의 거리를 저장하고 있는 self.dist matrix를 참조하여 neighbor를 구합니다.

```
# Cluster Expansion through neighbors
def clusterExpand(self, idx):
    self.realm[idx] = self.ClusterIdx
    tempIdx = 0

    while True:
        neighborIdx = self.neighbor[tempIdx]

        #print("Dormammu I've come to bargain")

        if self.visit[neighborIdx] == False:
            self.visit[neighborIdx] = True
            neighbor = self.getNeighbor(neighborIdx)

            if len(neighbor) > self.MinPts:
                for i in neighbor:
                    if i not in self.neighbor:
                        self.neighbor.append(i)

        if self.realm[neighborIdx] == 0:
            self.realm[neighborIdx] = self.ClusterIdx

        tempIdx += 1

        if len(self.neighbor) <= tempIdx:
            return
```

clsuterExpand(self, idx)

cluster를 확장하는 함수입니다. Self.neighbor를 참조하여 각 neighbor마다 미방문시 방문하고 getNeighbor함수로 Eps보다 작은 주변 neighbor list를 받아옵니다. MinPts보다 neighbor 수가 많을 시 self.neighbor에 neighbor list를 추가합니다. Self.neighbor의 모든 원소에 대해 이 과정을 반복합니다.

Label 예측이 끝난 test data set을 label을 포함하여 파일로 출력하는 함수입니다.

```
# Prune clusters more than self.n
def pruneCluster(self):
    if self.n < self.ClusterIdx:

        for i in range(self.ClusterIdx - self.n):

            temp = self.neighCount.index(min(self.neighCount))
            self.noiseCluster.append(temp + 1)
            self.neighCount[temp] = self.size
```

pruneCluster(self)

neighbor 수가 제일 적은 cluster를 noise cluster로 처리해주는 함수입니다.

```
# Write output file
def writeFile(self):
    fileIdx = 0
    for i in range(1, self.ClusterIdx + 1):

        if i not in self.noiseCluster:

            fileName = self.inputNum + "_cluster_" + str(fileIdx) + ".txt"
            df = self.df[self.df['cluster'] == i]
            df = df['index'].values
            np.savetxt(fileName, df, fmt='%d', delimiter='\n')
            fileIdx += 1
```

writeFile(self)

noise cluster를 제외한 cluster 구성 점 index를 text파일로 출력합니다.

3. 컴파일 환경 및 실행방법

Python 3.92버전을 사용하였고 launch.json파일을 이용하여 input file과 n, Eps, MinPts를 파일 실행 파라미터로 전달하였습니다.

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python: Current File",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "console": "integratedTerminal",
            "args": [
                "input1.txt",       //input file
                "8",                //n(number of clusters)
                "15",               //Eps(maxmimum radius of the neighborhood)
                "22"                //MinPts(minimum number of points)
            ]
        }
    ]
}
```

```
PS G:\내 드라이브\데이터사이언스\Project3>  g:; cd 'g:\내 드라이브\데이터사이언스\Project3'; & 'C:\Users\pmw14\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\pmw14\.vscode\extensions\ms-python.python-2021.5.842923320\pythonFiles\lib\python\debugpy\launcher' '55308' '--' 'g:\내 드라이브\데이터사이언스\Project3\clustering.py' 'input1.txt' '8' '15' '22'
```

파일 실행 스크린샷입니다.