Data Science Assignment #3

컴퓨터소프트웨어학부 2018062733 윤동빈

1. **Summary of your algorithm**

주어진 data objects를 hyperparatmer *n, Eps, MinPts*에 따라 clustering한다. 각 cluster에 속한 data objects는 서로 **density reachable**하며, indirectly density reachable한 관계인 경우 directly density reachable한 관계들의 chain으로 구성돼 있다고 할 수 있다. data object p, q가 있을 때, p로부터 거리가 *Eps* 이하인 object 개수가 *MinPts* 이상일 때 p는 **core object**라 하며 p와 q 사이의 거리가 *Eps* 이하일 경우 q는 p로부터 **directly density reachable**하다.

main 함수에서는 각 object p로부터 density reachable한 objects를 모아 clustering하고 있으며, 이 과정에서 `getDensityReachable()` 를 사용한다. clustering 전에 p가 core object인지 여부를 확인하기 위해 인접한 objects를 `adj`라는 list의 length로 판단하고 있으며, 각 object의 `adj` 초기화는 input을 얻어 처리하는 과정에서 이뤄진다.

2. **Detailed description**

```
class Obj:
    id = 0
    x = 0.1
    y = 0.1
    processed = False
    adj = []

    def __init__(self, id, x, y):
        self.id = id
        self.x = x
        self.y = y
        self.adj = []

    def isCore(self):
        return len(self.adj) >= MinPts-1
```

- **id** : 현재 object의 identifier
- **x, y** : 현재 object의 위치
- **processed** : 현재 object의 clustring 여부
- **adj** : 현재 object와 거리가 *Eps* 이하인 objects
- **isCore()** : 현재 object의 core object 여부 확인

```
def isAdj(p, q):
    if ((p.x - q.x) * (p.x - q.x) + (p.y - q.y) * (p.y - q.y)) <= Eps*Eps:
        return True
    return False
```

- **p, q** (paramter) : object p, q

- 각 object의 좌표(x, y)를 통해 거리를 계산하고 *Eps*와 비교해 인접 여부 확인
  `True` or `False` 반환

```
def getDensityReachable(center, cluster):
    for adjacent in center.adj:
        if adjacent.processed == False:
            adjacent.processed = True
            cluster.append(adjacent)

            if adjacent.isCore():
                getDensityReachable(adjacent, cluster)
```

- **center** (parameter) : 기준이 되는 object
- **cluster** (parameter) : 현재 clustering 중인 cluster
- center의 인접 object `adjcent`가 clustering 되지 않은 경우, clustering 진행
  만약 `adjcent`가 core object이면, `getDensityReachable()` 재귀 호출을 통해 현재 cluster에 속하는 objects의 density reachable 집합을 모두 구할 수 있도록 진행

```
# input : One command line
input_name = sys.argv[1]
n = int(sys.argv[2])
Eps = float(sys.argv[3])
MinPts = int(sys.argv[4])

allObj = []

f = open(input_name, 'r')

# input data 파싱해 allObj 업데이트
while True:

    line = f.readline()
    if not line:
        break

    id, x, y = line.split()
    id = int(id)
    x = float(x)
    y = float(y)

    allObj.append(Obj(id, x, y))

f.close()

# 각 object 이웃 전처리
for p in allObj:
    for q in allObj:
        if p == q:
            continue

        if isAdj(p, q):
            p.adj.append(q)
```

- command line 파싱해 `input_name`, `n`, `Eps`, `MinPts` 정의
- 주어진 objects 정보를 Obj `allObj` 에 저장
- core object 체크를 편리하게 하기 위해, `allObj` 에서 각 object의 `adj` 전처리

```
clusters = []

# DBSCAN
for p in allObj:
    if p.processed == False and p.isCore():
        newCluster = []
        p.processed = True

        getDensityReachable(p, newCluster)
        clusters.append(newCluster)
```

- **clusters** : DBSCAN 알고리즘에 의해 생성된 cluster 집합
- 각 object `p`에 대하여 clustering되지 않았고 core object이면 clustering 진행
  `newCluster`에 `p`로부터 density reachable한 objects 저장
  재귀 호출을 통해 density reachable한 모든 object를 구할 수 있도록 함
- 위 과정을 통해 최종적으로 얻은 `newCluster`은 `clusters`에 저장

```
# 만든 cluster 수 m이 n보다 큰 경우 작은 cluster (m-n)개 제거
if len(clusters) > n:
    clusters.sort(key=lambda c: len(c), reverse=True)

    while len(clusters) > n:
        clusters.pop()

# output file 작성
for i in range(n):
    output_name = 'input' + input_name[5] + '_cluster_' + str(i) + '.txt'

    f = open(output_name, 'w')

    for o in clusters[i]:
        f.write(str(o.id) + '\n')

    f.close()
```

- DBSCAN 알고리즘으로 만든 cluster 개수가 `m`이고 `m`이 `n`보다 크다면,
  cluster에 속한 object 개수를 기준으로 내림차순 정렬해 가장 작은 cluster $m - n$개 제거
- `n`개 cluster에 대해 output file 작성

3. **Instructions for compiling and testing**

```
C:₩Users₩동비니₩Desktop₩수업₩4학년_1학기₩데이터사이언스₩과제>python clustering.py input1.txt 8 15 22

C:₩Users₩동비니₩Desktop₩수업₩4학년_1학기₩데이터사이언스₩과제>PA3.exe input1
98.95664점
C:₩Users₩동비니₩Desktop₩수업₩4학년_1학기₩데이터사이언스₩과제>python clustering.py input2.txt 5 2 7

C:₩Users₩동비니₩Desktop₩수업₩4학년_1학기₩데이터사이언스₩과제>PA3.exe input2
94.76998점
C:₩Users₩동비니₩Desktop₩수업₩4학년_1학기₩데이터사이언스₩과제>python clustering.py input3.txt 4 5 5

C:₩Users₩동비니₩Desktop₩수업₩4학년_1학기₩데이터사이언스₩과제>PA3.exe input3
99.88194점
```

`python clustering.py input#.txt {n} {Eps} {MinPts}`

- `PA3.exe`를 통해 테스트 결과까지 출력

4. **Any other specification of implementation**

```
import sys
sys.setrecursionlimit(30000)
```

- test sample 외에 다른 test file에 대해 얼마나 많은 data가 들어올지 모르므로
  재귀 제한을 `30000`으로 연장