DBSCAN

author Dae In Lee

Quick Start

All programs are tested on macOS 10.14 & Ubuntu 18.04

Requirements

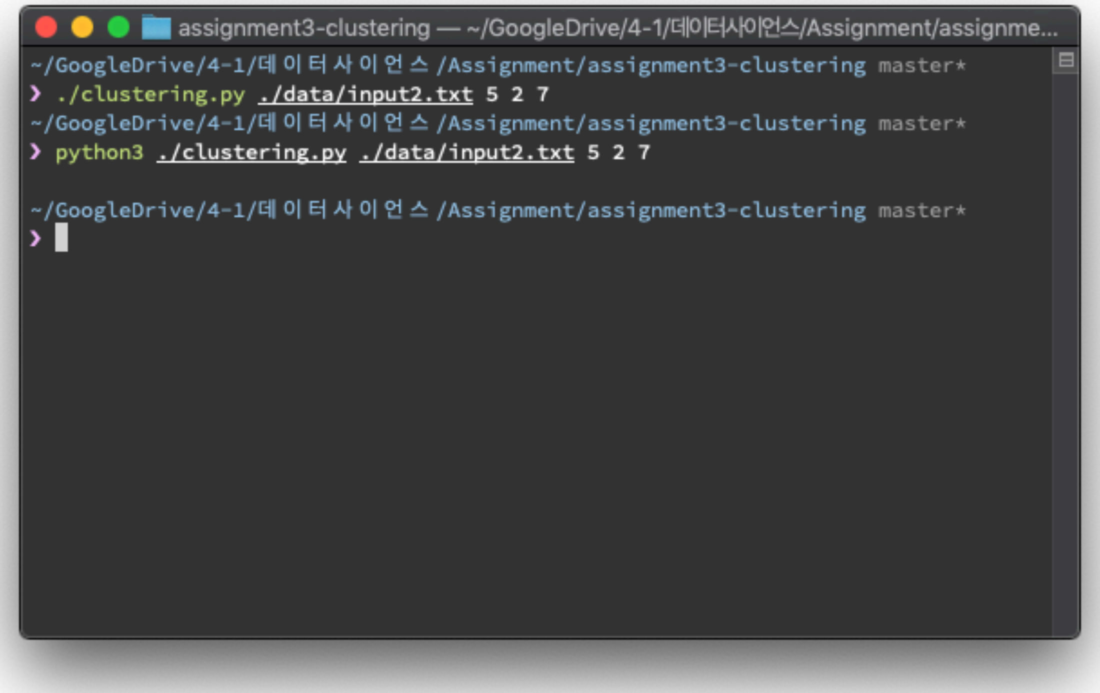
- Python3

How to Run

clustering.py

```
./clustering.py [input file] [number of clusters] [Eps] [MinPts]
python3 clustering.py [input file] [number of clusters] [Eps] [MinPts]
```

In order to exeucate program using first command, you must have execution permission for `clustering.py`.
If it gives permission error, either give it a execution permission or use second line command.


assignment3-clustering — ~/GoogleDrive/4-1/데이터사이언스/Assignment/assignme...

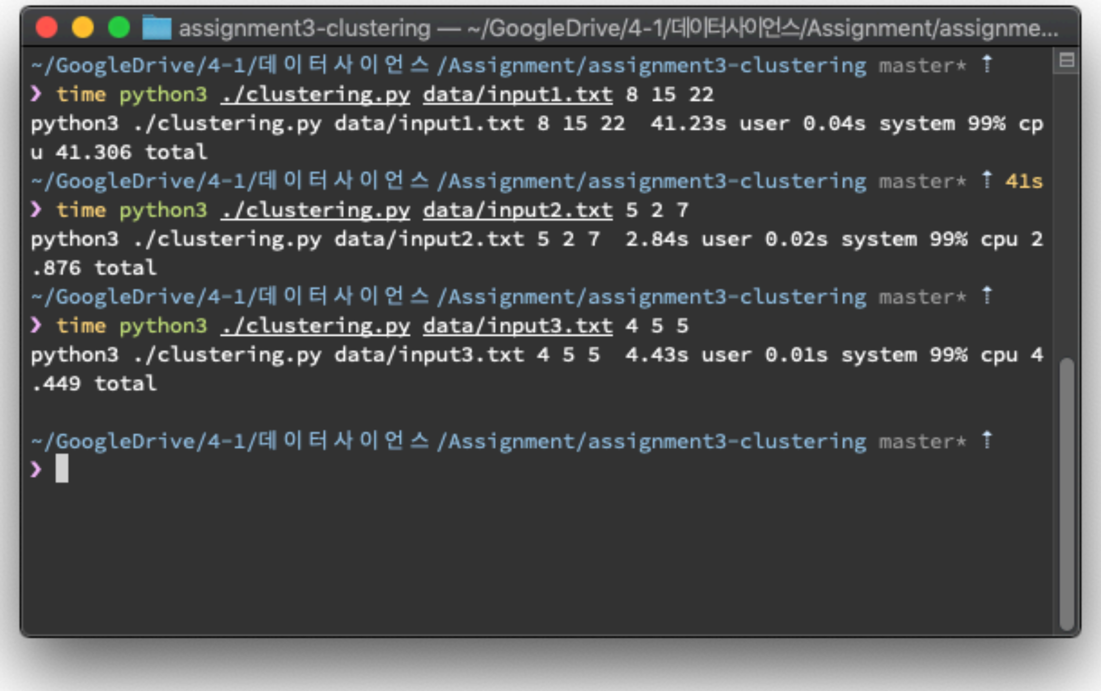
```
~/GoogleDrive/4-1/데이터사이언스/Assignment/assignment3-clustering master*
❯ ./clustering.py ./data/input2.txt 5 2 7
~/GoogleDrive/4-1/데이터사이언스/Assignment/assignment3-clustering master*
❯ python3 ./clustering.py ./data/input2.txt 5 2 7

~/GoogleDrive/4-1/데이터사이언스/Assignment/assignment3-clustering master*
❯
```

Results

Note that output files will be located on the same folder as input file. (in data folder in example.)

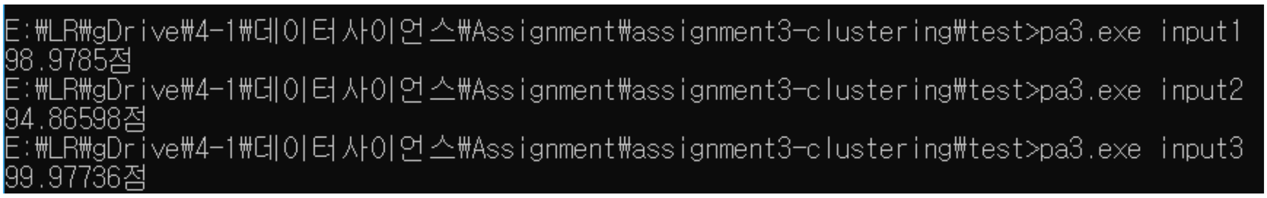
Result score may differ from each excution because DBSCAN algorithm choose *random point* to generate *density-based clustering*.



assignment3-clustering — ~/GoogleDrive/4-1/데이터사이언스/Assignment/assignme...

```
~/GoogleDrive/4-1/데이터사이언스/Assignment/assignment3-clustering master* ⭡
❯ time python3 ./clustering.py data/input1.txt 8 15 22
python3 ./clustering.py data/input1.txt 8 15 22  41.23s user 0.04s system 99% cp
u 41.306 total
~/GoogleDrive/4-1/데이터사이언스/Assignment/assignment3-clustering master* ⭡ 41s
❯ time python3 ./clustering.py data/input2.txt 5 2 7
python3 ./clustering.py data/input2.txt 5 2 7  2.84s user 0.02s system 99% cpu 2
.876 total
~/GoogleDrive/4-1/데이터사이언스/Assignment/assignment3-clustering master* ⭡
❯ time python3 ./clustering.py data/input3.txt 4 5 5
python3 ./clustering.py data/input3.txt 4 5 5  4.43s user 0.01s system 99% cpu 4
.449 total

~/GoogleDrive/4-1/데이터사이언스/Assignment/assignment3-clustering master* ⭡
❯ 
```



```
E:₩LR₩gDrive₩4-1₩데이터사이언스₩Assignment₩assignment3-clustering₩test>pa3.exe input1
98.9785점
E:₩LR₩gDrive₩4-1₩데이터사이언스₩Assignment₩assignment3-clustering₩test>pa3.exe input2
94.86598점
E:₩LR₩gDrive₩4-1₩데이터사이언스₩Assignment₩assignment3-clustering₩test>pa3.exe input3
99.97736점
```

Implementation

class Point

```python
class Point:
    def __init__(self, id, x, y):
        self.id = int(id)
        self.x = float(x)
        self.y = float(y)
        self.isVisited = False
        self.label = -1

    def __repr__(self):
        return self.__str__()

    def __str__(self):
        ret = "("
        ret += str(self.id) + ": "
        ret += str(self.x) + ", "
        ret += str(self.y) + ")"
        return ret
```

Each line from input file will be converted to a `Point` object.

`id`: unique number that represents Point object.
`x`: x-coordinate in 2D
`y`: y-coordinate in 2D
`isVisited`: boolean value that is used while DBSCAN algorithm.
label: id of cluster that it belongs to.

{isVisited, label} == {False, -1} --> not yet identified.
{isVisited, label} == {True, -1} --> outlier
Point may become {True, -1} after labelConverter is used to reduce # of clusters.

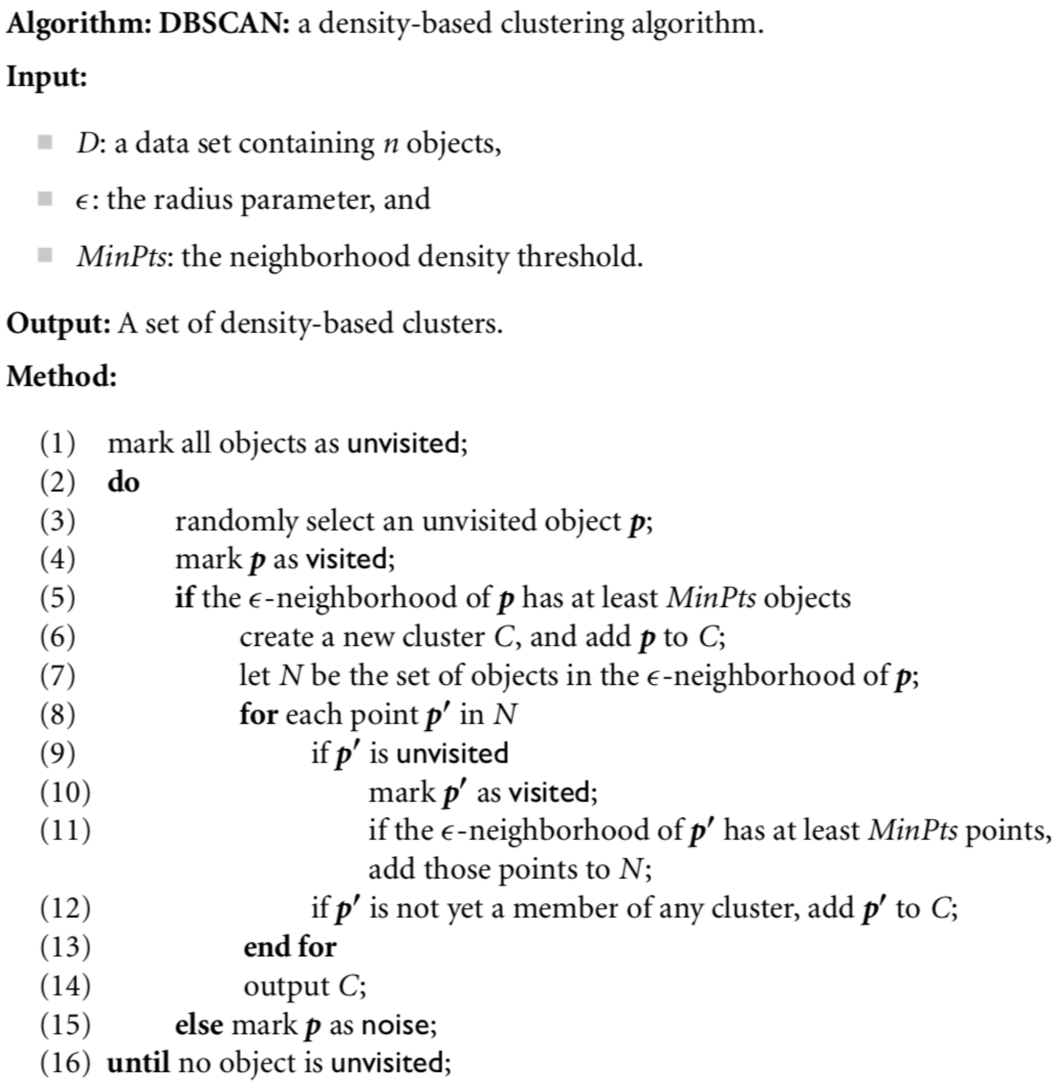
loadData()

Simply read line to line from *input file* and generate `Point` object.
Return value is list of all `Point` objects that will be used to clustering.

findNeighbor()

Find all neighbor Points from `cur` point. Neighbor must be positioned within radius of `eps`.
Time Complexity = O(n)

dbscan()



Algorithm: DBSCAN: a density-based clustering algorithm.

Input:

- D: a data set containing n objects,
- ϵ: the radius parameter, and
- $MinPts$: the neighborhood density threshold.

Output: A set of density-based clusters.

Method:

```
mark all objects as unvisited;
do
    randomly select an unvisited object $p$;
    mark $p$ as visited;
    if the $\epsilon$-neighborhood of $p$ has at least $MinPts$ objects
        create a new cluster $C$, and add $p$ to $C$;
        let $N$ be the set of objects in the $\epsilon$-neighborhood of $p$;
        for each point $p'$ in $N$
            if $p'$ is unvisited
                mark $p'$ as visited;
                if the $\epsilon$-neighborhood of $p'$ has at least $MinPts$ points,
                add those points to $N$;
            if $p'$ is not yet a member of any cluster, add $p'$ to $C$;
        end for
        output $C$;
    else mark $p$ as noise;
until no object is unvisited;
```

pseudo code of DBSCAN[1]
Implementation of DBSCAN algorithm. *Time Complexity* = O(n^2)

After creating *density-based clusters*, function also generates `labelconverter` dictionary.
Which is used to remove any extra clusters exceeding `N`(user given argument of max cluser #).

createOutputFile()

Generates `N` output files. Each file represents one *density-based cluster*.
`labelConverter` is used at this point to eliminate extra clusters(labels). Any *Point* with -1 as a label is either outlier or belongs to extra cluster.
Note that if number of created clusters is smaller than `N`, some output files are created as a blank file.

[1] Data Mining Concepts and Techniques Chapter 10.4↩