

DBSCAN

author Dae In Lee

Quick Start

All programs are tested on macOS 10.14 & Ubuntu 18.04

Requirements

- Python3

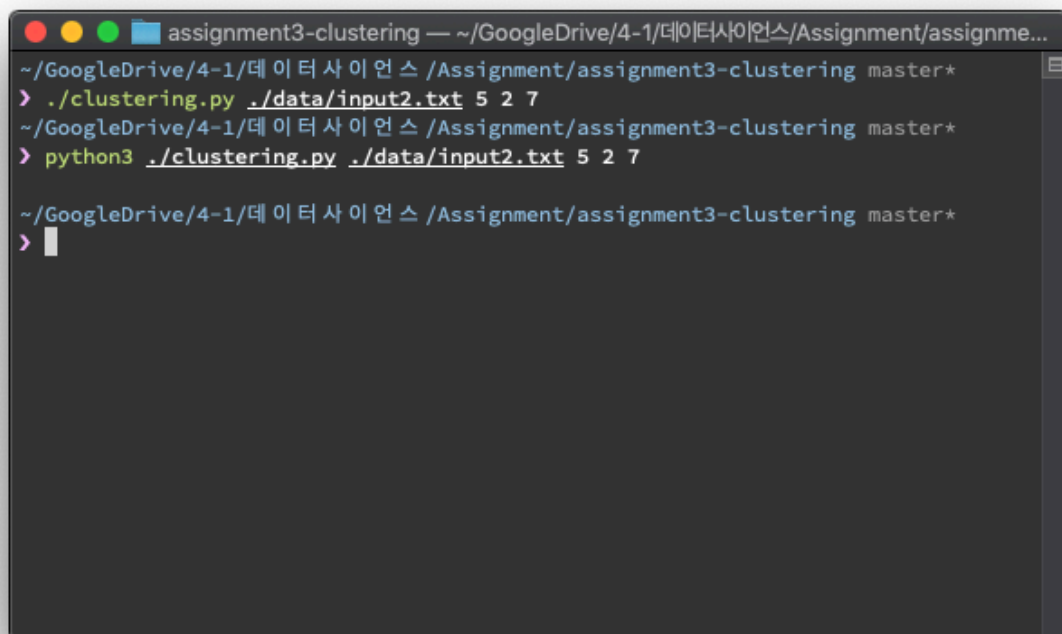
How to Run

clustering.py

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

In order to execute 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.

A terminal window titled "assignment3-clustering" with a path "~/GoogleDrive/4-1/데이터사이언스/Assignment/assignme...". The prompt is "~/GoogleDrive/4-1/데이터사이언스/Assignment/assignment3-clustering master*". The user enters two commands: first, "./clustering.py ./data/input2.txt 5 2 7", and second, "python3 ./clustering.py ./data/input2.txt 5 2 7". The terminal shows the prompt after each command, indicating successful execution without errors.

```
~/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*
> 
```

evaluation.py

```
./evaluation.py  
python3 evaluation.py
```

Running above command will give you two options to run evaluation program.

1. Use existing outputfiles at test/ folder
2. Run DBSCAN to create new outputFiles

Either way will use ideal & created output files to evaluate the accuracy.

Note that in order to run first command, you **MUST** have output files ready.

Directory of the files may be modified with below global variables.

```
inputFileDir   = "data/"  
idealFileDir   = "test/"  
outputFileDir  = "test/"
```

In order to create new tests, you must append appropriate values to below list, and ideal cluster files ready at `idealFileDir`.

```
testNameList    = [ "input1", "input2", "input3" ]  
clusterNumList  = [ 8,        5,        4        ]  
epsList         = [ 15,       2,        5        ]  
minPtsList      = [ 22,       7,        5        ]
```

Score of `evaluation.py` is equal to that of `PA3.exe`. But it gives extra information about clustered data.

```
assignment3-clustering — LazyRen@Daeln-MBP — ..t3-clustering — -zsh - -zsh...
~/GoogleDrive/4-1/데이터 사이언스/Assignment/assignment3-clustering master*
[> ./evaluation.py
***** USAGE *****
1. Use existing outputfiles at test/
2. Run DBSCAN to create new outputFiles
Input: 1

Starting Evaluation: input1
7676 points are clustered in ideal files
7485 points are clustered in output files
29155750 / 29456650 = 98.97850%
Evaluation took 10.975946 seconds

Starting Evaluation: input2
2000 points are clustered in ideal files
1942 points are clustered in output files
1896371 / 1999000 = 94.86598%
Evaluation took 0.735864 seconds

Starting Evaluation: input3
2100 points are clustered in ideal files
2099 points are clustered in output files
2203451 / 2203950 = 99.97736%
Evaluation took 0.814148 seconds
```

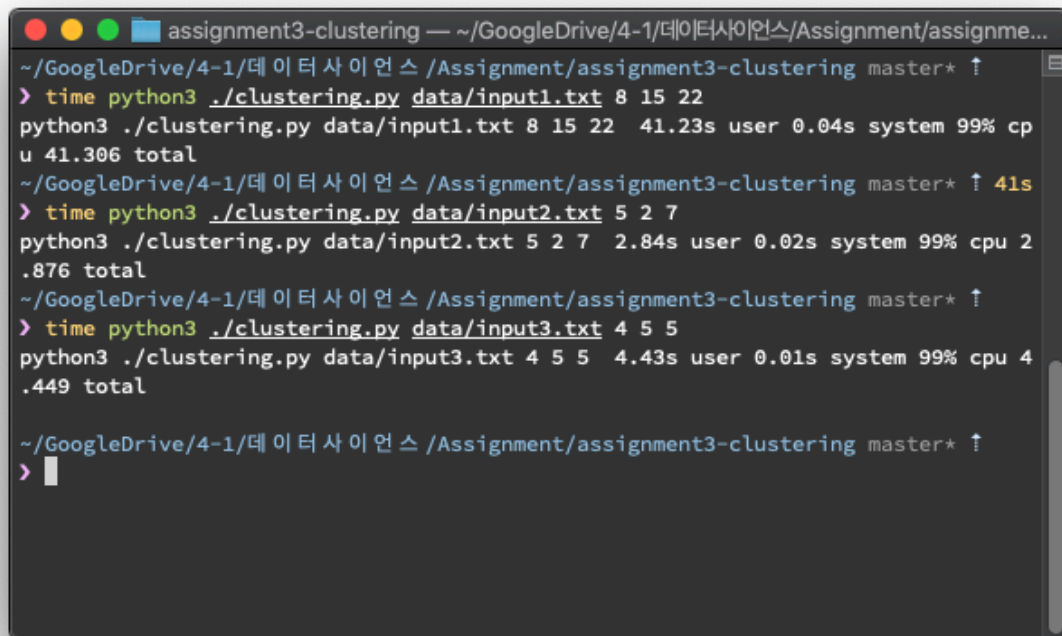
```
명령 프롬프트
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\LazyRen>E:
E:>cd E:\LR\gDrive\4-1\데이터 사이언스\Assignment\assignment3-clustering\test
E:\LR\gDrive\4-1\데이터 사이언스\Assignment\assignment3-clustering\test>PA3.exe input1
98.9785점
E:\LR\gDrive\4-1\데이터 사이언스\Assignment\assignment3-clustering\test>PA3.exe input2
94.86598점
E:\LR\gDrive\4-1\데이터 사이언스\Assignment\assignment3-clustering\test>PA3.exe input3
99.97736점
E:\LR\gDrive\4-1\데이터 사이언스\Assignment\assignment3-clustering\test>_
```

Results

Note that output files will be located on the same folder as input file. (in data folder in example)
< If `evaluation.py` is used to create output files(by choosing option 2), it will generate output files at test folder. >

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



```
~/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

```
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:

- (1) mark all objects as **unvisited**;
- (2) **do**
- (3) randomly select an unvisited object p ;
- (4) mark p as **visited**;
- (5) **if** the ϵ -neighborhood of p has at least $MinPts$ objects
- (6) create a new cluster C , and add p to C ;
- (7) let N be the set of objects in the ϵ -neighborhood of p ;
- (8) **for** each point p' in N
- (9) **if** p' is **unvisited**
- (10) mark p' as **visited**;
- (11) **if** the ϵ -neighborhood of p' has at least $MinPts$ points,
 add those points to N ;
- (12) **if** p' is not yet a member of any cluster, add p' to C ;
- (13) **end for**
- (14) output C ;
- (15) **else** mark p as **noise**;
- (16) **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 cluster #).

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.