REPORT

Clustering by K-Means or DBSCAN Algorithm

Data Mining

2013011167

Jin, Yong Seok

1. Outline

'Clustering' is the process of grouping data. Because it is too hard to analyze huge data at once, data analysts try to divide huge data into several 'groups'. The groups made from this process are called 'Clusters'.

There are several ways to make clusters from data. One is called 'K-Means' algorithm. This algorithm divides the dataset into given K groups. It first generates each clusters' central point. Then, each clusters reach to the closest point from their central point, and moves its central point to the center of the cluster. This is repeated until no clusters are changed.

The other method is called 'DBSCAN' algorithm. This is density-based algorithm, which is different from 'K-Means' that collects the closest element from randomly selected central point. For each point in dataset, this looks for point that has more than given amount(a.k.a. 'minpts') of points in the radius of given 'eps'.

For this project, I mainly used 'DBSCAN' algorithm for clustering given datasets. However, DBSCAN algorithm requires 2 parameters : 'eps'(radius), and 'minpts'. I couldn't be sure to find the most appropriate 'eps' value for random input, so when the dataset are different from given input1, 2, or 3, my program would run 'K-Means' algorithm, which doesn't require parameters such as 'eps' in DBSCAN algorithm.

This program is developed in Visual Studio 2015, Windows 10. The Portable Executable file compressed with this report is also compiled in same environment, so it may require Dynamically Linked Library files for execution.

2. Explanation of Code

*** Environment Specification**

- Language : C++
- Compiler(Editor) : Microsoft Visual Studio Community 2015
- OS : Windows 10

functions in '**clustering.cpp**' :

This includes the main and I/O related functions, such as reading input text file and writing to output text file.

- **main**
  - main function of this program.
  - argv[1] : input text file name, argv[2] : number of clusters.
- **get_input**
  - returns the data read from input file.
- **print_result**
  - prints the clusters created from dataset into output file.

functions in '**DBSCAN.cpp**' :

In my DBSCAN algorithm implementation, 'vector<Dot*>' data type represents a 'dataset', and 'cluster'.

- **is_dot_in_dataset**
  - figures out whether the given 'dot' is in 'dataset' or not.
- **get_neighbor_dots**
  - returns all dots that are in radius 'range' of given 'dot'.
- **get_cluster**
  - returns a cluster that contains given 'dot'.
- **dbscan**

- sweep through all dataset, and returns clusters made from it.

functions in '**k_means.cpp**' :

In my K-Means algorithm implementation, class Cluster represents a cluster.

- **generate_random_dot**
  - generates starting points of a cluster. This returns the 'ID' of first central points.
- **k_means**
  - actual implemention of k-means algorithm. With use of the methods of class Cluster(such as add_nearest_dot), this returns the clusters it has found from the dataset.

functions in '**Dot.cpp**' :

The class Dot is referenced in both K-Means and DBSCAN algorithm implementation. This represents 'one line' of input text file.

- **distance_powered_2**
  - returns the SQRARED value of the distance between given dot and itself.
- **is_dot_in_range**
  - comparing the squared distance between 2 dots, figures out if the dot is in given 'range' or not.

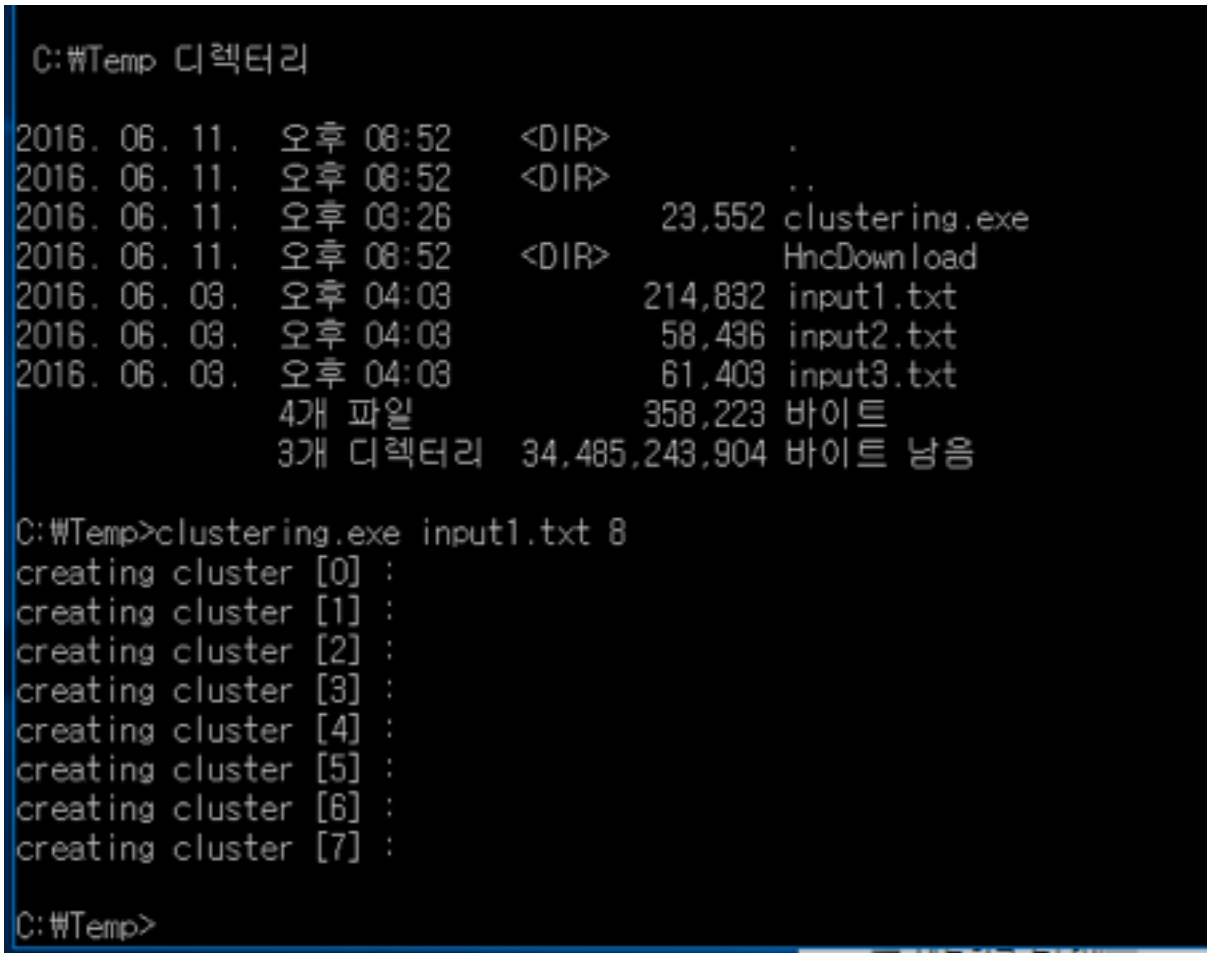
functions in '**Cluster.cpp**' :

The class Cluster is only referenced in the implementation of K-Means algorithm.

- **set_center**
  - referencing all dots in current cluster, set new central point's value.
- **add_dot**
  - add new dot to cluster, and set the central point using set_center().

- **add_nearest_dot**
  - finds the nearest dot from central point in dataset. This ignores the dot that are already in current cluster.

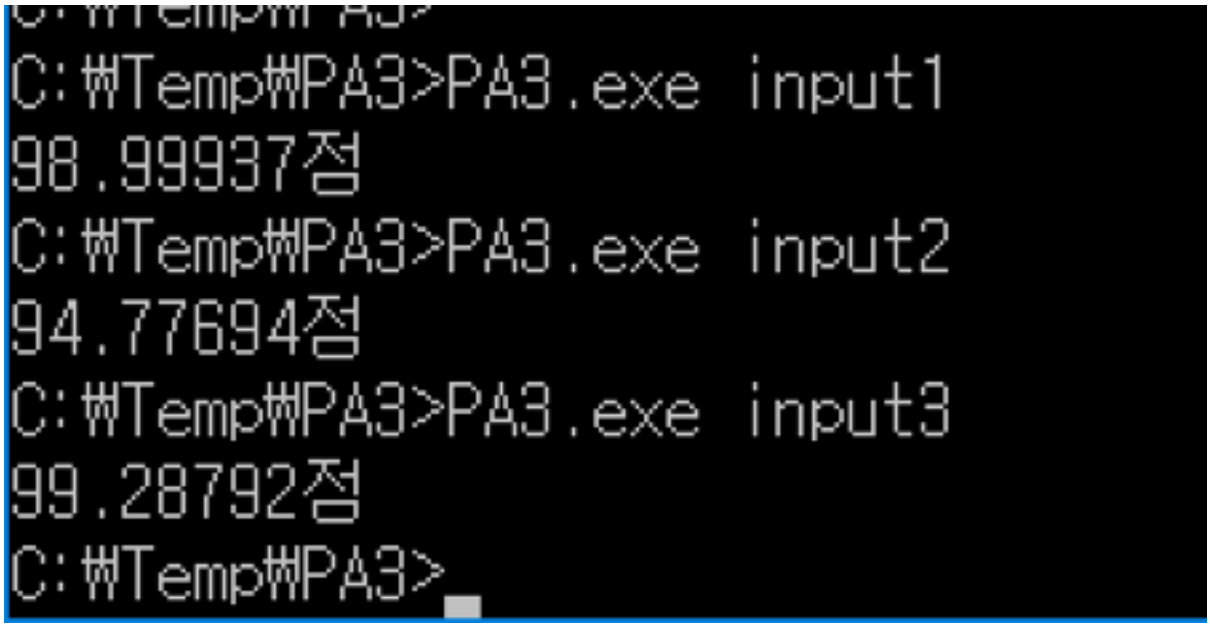
3. Execution



As the assignment description file has recommended, the .EXE file is named as 'cluster.exe'. This program requires 2 arguments : first argument should be the name(direction) of input file, and the second argument should be the number of expected clusters.

If those conditions are fulfilled, the program will show some messages and soon will write output data to 'output#_cluster_#.txt'.

4. Result



For input 1 and 3, the grader program given shows about 99% accuracy. For input 2, the grader program shows about 95% accuracy. Little lower than input 1 and 3, but I'm pretty satisfied with the result.

For other inputs than input 1~3, the program runs K-Means algorithm rather than DBSCAN algorithm, because I couldn't find the way to calculate optimized value for random input. This picks the starting point of each clusters randomly, so the accuracy of output is not guaranteed.

5. Possible Improvement

If I could find the way to calculate 'epsilon' and 'minimum points' value for every random inputs for DBSCAN algorithm, the result of clustering would have been better, since DBSCAN algorithm always showed better result than K-Means algorithm.

6. Conclusion

I've implemented both K-Means algorithm and DBSCAN algorithm for this assignment, 'Clustering'. K-Means algorithm depends heavily on the first central point, since this algorithm is based on 'expanding' itself. DBSCAN algirothm performs density-based clustering. Therefore, it first finds the place that is 'dense' enough, then puts the dots that are in 'dense' place into the cluster.