INF 553 – Spring 2018

Homework #4: Clustering

Due: April 6, Friday
100 points

In this homework, we consider implementing the CURE clustering algorithm **FirstName_LastName_cure.py** in Python (**Without Spark**). Remember from class that it's an efficient clustering algorithm for handling large data, and how it can identify clusters having non-spherical shapes and size variances.

CURE works in two passes:

1. Pick a sample from data given, cluster it hierarchically, and determine k clusters from dendrogram. Identify n Representative points from each cluster and move them p% towards the centroid of cluster
2. Scan complete data and assign points to closest cluster by taking distance between point x and cluster C

Note: Ideally, sample is chosen randomly from the full data in pass 1. However, for everyone to arrive at the same results, I am giving the sample file (subset of data) along with full data.

You will be given with two data files – sample and full data in Euclidean space. Your task is to do hierarchical clustering on the sample data and come up with k clusters (user specified). Remember in hierarchical clustering, each point begins in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. Cluster distance is given by the distance of two closest points, one from each cluster. From the resulting Dendrogram, you can find the k clusters. A Naive implementation of hierarchical clustering should do given that the sample file in our case will be smaller.

This is how CURE will work step by step:

- Find initial clusters from a sample
  - It will take a sample of data from disk and perform hierarchical clustering on the sample
  - Cluster distance is given by the distance of two closest points, one from each cluster.
  - It then finds k initial clusters by cutting the dendrogram properly

- Finding representatives for each cluster
  - Each cluster is represented by n representative points where n can be user specified
  - Representatives are chosen by first picking a smallest coordinate point (Smallest x and smallest y in set of 2D points) in the cluster (formed by the sample), then repeatedly find other points farthest away from representative points already selected
  - It then moves representative p% towards the centroid of cluster, where p can be user specified.

- Assign remaining data points (loaded from disk) to the closest clusters by taking distance from point x to the closest representative point in C

Execution Format:

python FirstName_LastName_cure.py <sample_data> <full_data> <k> <n> <p> <output_file>

1. sample_data, a text file containing subset of 2D points taken randomly from full data
2. full_data, a text file containing 2D points to be clustered using CURE

   Both Input data files will look like below: Points in each line (x coordinate followed by y coordinate)

```
-0.552447,-0.458022
-0.0969878,-0.853309
0.839883,-0.0245745
0.955918,-0.0863494
0.901937,-0.104899
```

3. k, an integer specifying number of clusters
4. n, an integer specifying number of representative points in clusters formed by sample
5. p, a floating point by which representative points need to move towards the centroid
6. output_file is name of file you will write cluster number for each data point in full_data.txt

Example execution:

python FirstName_LastName_cure.py sample_data.txt full_data.txt 3 4 0.2 output.txt

Which means, program should find k=3 clusters from the sample file by identifying n=4 representative points, moving them by p=20% towards the centroid of clusters, and write the cluster number for each data point in full data file to **output.txt (Do not hardcode. Read it via sys.argv)**

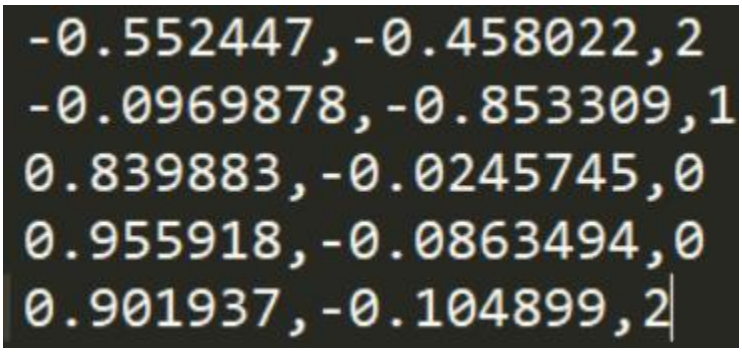
Output Format:

Along with writing cluster number to file, program should print representative points for each cluster on console. For e.g. for above sample execution (3 clusters and 4 representative points), each line in the console below (Just a dummy example) is a Python list containing 4 representative points, and there are3 lines (3 clusters).

```
[[1, 2], [3, 4], [5, 6], [7, 8]]
[[11, 12], [13, 14], [15, 16], [17, 18]]
[[21, 22], [23, 24], [25, 26], [27, 28]]
```

Order does not matter for clusters. But make sure representative points for each cluster are in the same order you found them (First point chosen is a smallest coordinate in that cluster, next point is farthest from first point, third point farthest from first two etc.)

Example output file:



For every point in full_data.txt, you need to assign a cluster number and write to output file as a 3rd column as above (0,1,2 for 3 clusters, and 0,1,2,3 for 4 clusters etc.). Again, order doesn't matter for cluster number. Points in same cluster should have the same cluster number. So, among 3 clusters, points in first cluster can even have cluster number 2, and other 2 clusters can have 1 and 0.

Submission on Blackboard:

A single file **FirstName_LastName_cure.py** which can be executed with parameters mentioned above. If you are using Python 3, please rename the file to **FirstName_LastName_cure3.py**

Grading Criteria:

- **Grade breakdown:**
  Output up to representative points – 50 points
  Output file containing the clustered data points with cluster number – 50 points
- If order for representative points is not followed as mentioned above, there will be 20% penalty.
- There will be 20% penalty if output format for output file is not followed.
- Program execution time needs to be under **3 minutes.** 30% penalty otherwise. We will also test with a different data set of similar size as the one given.
- 10% penalty if naming convention for python scripts is not followed.
- Assignment is due at 11:59 pm on 04/06 (2 weeks). Late submission will have 10% of penalty for every 24 hours that it is late. No credit will be given after 72 hours of its due time.

Note:

- You may use numpy package in this homework. To install it on EC2, execute "sudo pip install numpy".
- **We will be using Moss for plagiarism detection**. Do not share your code with anyone, or copy from others!