Big Data Computing

Homework 3

In this homework, you will run a Spark program on the CloudVeneto cluster. The core of the Spark program will be the implementation of **2-round coreset-based MapReduce algorithm for k-center with z outliers**, which works as follows: in **Round 1**, separately for each partition of the dataset, a *weighted coreset* of k+z+1 points is computed, where the weight assigned to a coreset point p is the number of points in the partition which are closest to p (ties broken arbitrarily); in **Round 2**, the L weighted coresets (one from each partition) are gathered into one weighted coreset of size (k+z+1)*L, and one reducer runs the sequential algorithm developed for Homework 2 (**SeqWeightedOutliers**) on this weighted coreset to extract the final solution. In the homework you will test the accuracy of the solution and the scalability of the various steps.

TASK 1

Implement the 2-round MapReduce algorithm described above. Structure to follow:

- Receives in input the following command-line (CLI) arguments:
  - **A path to a text file containing point set in Euclidean space**. Each line of the file contains, separated by commas, the coordinates of a point. Your program should make no assumptions on the number of dimensions!
  - **3 integers**: k (number of centers), z (number of outliers), and L (number of partitions).
- Reads the input points into and RDD of Vector called **inputPoints**, subdivided into L partitions, sets the Spark configuration, and prints various statistics.
- Runs a method **MR_kCenterOutliers** to compute the solution (i.e., a set of at most k centers) to the k-center problem with z outliers for the input dataset. The method implements the 2-round algorithm described. In Round 1 it extracts k+z+1 coreset points from each partition using method **kCenterFFT** which implements the Farthest-First Traversal algorithm, and compute the weights of the coreset points using method **computeWeights**. In Round 2, it collects the weighted coreset into a local data structure and runs method **SeqWeightedOutliers**, "recycled" from Homework 2, to extract and return the final set of centers (you must fill in this latter part).
- Computes the value of the objective function for the returned solution (i.e., the maximum distance of a point from its closest center, excluding the z largest distances), using method **computeObjective**.
- Prints the value of the objective function and the time taken by computeObjective.

TASK 2

Write a Java program (*hw3.java*) and **complete the code** as follows (*parts where to add code are marked in template*):

1. Insert the code for SeqWeightedOuliers from your Homework 2.
2. Complete Round 2 of MR_kCenterOutliers to extract and return the final solution. **IMPORTANT**: ***you must run SeqWeightedOutliers on the weighted coreset using alpha=2.***
3. Add suitable istructions to MR_kCenterOutliers, so to measure and print separately the time required by Round 1 and Round 2. Please be aware of the Spark's lazy evaluation.
4. Write the code for method computObjective. It is important that you keep in mind that the input dataset may be very large and that, in this case, any structure of the size of this dataset may not fit into local memory.

The output of your code should be the same as the provided example file *OutputFormat.txt.*

TASK 3 & 4

Test and debug your program in local mode on your PC *to make sure that it runs correctly.* For this local test you can use the 16-point dataset *testdataHW3.txt* which you can download here and the datasets *uber-small.csv* and *artificial9000.txt* that you used also for Homework 2 (available in Homework 2 directory).
Test your program on the cluster using the datasets which have been preloaded in the HDFS available in the cluster.