Big Data Computing

Homework 2

Homeworks 2 and 3 will focus on the **k-center with z outliers problem**, that is, the robust version of the k-center problem which is useful in the analysis of noisy data (a quite common scenario in big data computing). Given a set P of points and two integers k and z, the problem requires to determine a set $S \subset P$ of k centers which minimize the maximum distance of a point of P-Z from S, where Z are the z farthest points of P from S. In other words, with respect to the standard k-center problem, in the k-center with z outliers problem, we are allowed to disregard the z farthest points from S in the objective function. Unfortunately, the solution of this problem turns out much harder than the one of the standard k-center problem. The 3-approximation sequential algorithm by Charikar et al. for k-center with z outliers, which we call **kcenterOUT**, is simple to implement but has superlinear complexity (more than quadratic, unless sophisticated data structures are used).
The two homeworks will demonstrate that in this case a coreset-based approach can be successfully employed. In Homework 2 you will implement the 3-approximation sequential algorithm and will get a first-hand experience of its inefficiency. In Homework 3, you will implement a 2-round MapReduce coreset-based algorithm for the problem, where the use of the inefficient 3-approximation is confined to a small coreset computed in parallel through the efficient Farthest-First Traversal.

TASK

We will work with points in Euclidean space (real cooordinates) and with the Euclidean L2-distance. You must:

1. **Develop a method SeqWeightedOutliers(P,W,k,z,alpha) which implements the weighted variant of kcenterOUT** (the 3-approximation algorithm for k-center with z-outliers). The method takes as input the set of points P, the set of weights W, the number of centers k, the number of outliers z, and the coefficient alpha used by the algorithm, and returns the set of centers S computed as specified by the algorithm. It is understood that the i-th integer in W is the weight of the i-th point in P. **Java**: represent P and S as *ArrayList<Vector>*, and W as *ArrayList<Long>*.
2. **Develop a method ComputeObjective(P,S,z)** which computes the value of the objective function for the set of points P, the set of centers S, and z outliers (the number of centers, which is the size of S, is not needed as a parameter). Hint: you may compute all distances d(x,S), for every x in P, sort them, exclude the z largest distances, and return the largest among the remaining ones. ***Note that in this case we are not using weights!***
3. **Write a Java program** (*hw2.java*) which 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!
   - **An integer k** (the number of centers).
   - **An integer z** (the number of allowed outliers).

The program must do the following:

- Read the points in the input file into an ArrayList<Vector> called **inputPoints**.
- Create an ArrayList<Long> called **weights** of the same cardinality of inputPoints, initialized with all 1's. (In this homework we will use unit weights, but in Homework 3 we will need the generality of arbitrary integer weights!).
- Run **SeqWeightedOutliers(inputPoints,weights,k,z,0)** to compute a set of (at most) k centers. The output of the method must be saved into an ArrayList<Vector> called **solution**.
- Run **ComputeObjective(inputPoints,solution,z)** and save the output in a variable called **objective**.
- Return as output the following quantities: n = |P|, k, z, the initial value of the guess r, the final value of the guess r, and the number of guesses made by SeqWeightedOutliers(inputPoints,weights,k,z,0), the value of the objective function (variable objective), and the time (in milliseconds) required by the execution of SeqWeightedOutliers(inputPoints,weights,k,z,0). IT IS IMPORTANT THAT ALL PROGRAMS USE THE SAME OUTPUT FORMAT AS IN THE PROVIDED EXAMPLE FILE: *OutputFormat.txt*.

To test your program you can use the 3 provided datasets, in the same section as this specification, together with the outputs of some runs of the program on the datasets.