Homework 2

The purpose of this second homework is to demonstrate the effectiveness of a coreset-based strategy to get high-quality solutions from a small sample of a dataset, when the analysis of the whole dataset would be computationally too costly (or even unfeasible). You will focus on the **Max pairwise distance** problem which, given a set S of points from a metric space, requires to determine the maximum distance between two points. (You can review the problem from Slides 38-40 of the set of Slides on MapReduce.)

For this problem you have to implement the following 3 approaches:

- *Exact solution*;
- *2-approximation by taking k random points* and returning the maximum distance between these points and all other points of S;
- *Exact solution* of a subset of C of S, where C are *k centers retruned by Farthest-First Traversal*;

and compare their accuracy and running times. This homework asks you to devise a **sequential program**, and no Spark RDDs will be involved. However, Java users will be using some Spark features to represent points (see next section).

Representation of points

We will work with *points in Euclidean space* (real cooordinates) and with the standard *Euclidean L2-distance*.

For Java users. In Spark, points can be represented as instances of the class `org.apache.spark.mllib.linalg.Vector` and can be manipulated through static methods offered by the class `org.apache.spark.mllib.linalg.Vectors`.

Be careful to use the classes from the `org.apache.spark.mllib` package. There are classes with the same name in `org.apache.spark.ml` package which are functionally equivalent, but incompatible with those of the `org.apache.spark.mllib` package.

For example, method `Vectors.dense(x)` transforms an array x of double into an instance of class `Vector`, while method `Vectors.sqdist(x,y)` computes the `(d(x,y))^2` between two `Vector` x and y, where "d(.,.)" is the standard Euclidean L2-distance. Details on these classes can be found on the Spark Java API (for Java users).

For Python users. We suggest to represent points as the standard `tuple` of `float` (i.e., point = (x1, x2, ...)). Although Spark provides the class Vector also for Python (see pyspark.mllib package), its performance is very poor and its more convenient to use tuples for points from low-dimensional spaces.

Assignment

You must

1. **Develop the following 3 methods (functions in Python)**

   `exactMPD(S)`: receives in input a set of points S and returns the max distance between two points in S.

   `twoApproxMPD(S,k)`: receives in input a set of points S and an interger k < |S|, selects k points at random from S (let S' denote the set of these k points) and returns the maximum distance d(x,y), over all x in S' and y in S. Define a constant `SEED` in your main program (e.g., assigning it one of your university IDs as a value), and use that value as a seed for the random generator. **For Java users:** `SEED` must be a

long and you can use method `setSeed` from your random generator to initialize the seed. **For Python users:** you can use the method `random.seed(SEED)` from the module random.

`kCenterMPD(S,k)`: receives in input a set of points S and an integer k < |S|, and returns a set C of k centers selected from S using the Farthest-First Traversal algorithm. It is important that `kCenterMPD(S,k)` run in O(|S|*k) time (see exercise on Slide 23 of the set of Slides on Clustering, Part 2.

The input point set S (as well as the output C of `kCenterMPD(S,k)`) must be represented as instances of `ArrayList<Vector>` (`list` of `tuple` in Python).

2. **Write a program GxxHW2.java** (for Java users) or **GxxHW2.py** (for Python users), where xx is your two-digit group number, which receives in input a path to a text file containing a set of points in Euclidean space, and an integer "k". The file must contain one point per line, with coordinates separated by comma. The program incorporates the methods/functions developed above and does the following:

   - Reads the input

     Java users: reads the points from the file into an `ArrayList<Vector>` called "inputPoints". For reading the points you can use the code and auxiliary methods provided in the file VectorInput.java.

     Python users: reads the points from the file into a `list` of `tuple` called "inputPoints". For reading the points you can use the function `readTuplesSeq` provided in the file TupleInput.py.

   - Runs `exactMPD(inputPoints)`, measuring its running time (in ms), and prints the following lines:

     ```
     EXACT ALGORITHM
     Max distance =  max distance returned by the method
     Running time =  running time of the method.
     ```

   - Runs `twoApproxMPD(inputPoints,k)`, measuring its running time (in ms), and prints the following lines:

     ```
     2-APPROXIMATION ALGORITHM
     k =  value of k.
     Max distance =  max distance returned by the method
     Running time =  running time of the method.
     ```

   - Runs `kcenterMPD(inputPoints,k)`, saves the returned points in an `ArrayList<Vector>` (`list` of `tuple` for Python users) called "centers", and runs `exactMPD(centers)`, measuring the combined running time (in ms) of the two methods. Then, it prints the following lines:

     ```
     k-CENTER-BASED ALGORITHM
     k =  value of k.
     Max distance =  max distance returned by exactMPD(centers)
     Running time =  combined running time of the two methods.
     ```

     As test datasets, you can use the ones contained in the following webpage.

     Make sure to add short but explicative comments to your code, when needed.

     SUBMISSION INSTRUCTIONS. **Each group must submit only one homework** using the submission form of the **Homework 2 section** in the Moodle page of the course. Specifically, Each group has to submit a single file (**GxxHW2.java** or **GxxHW2.py** depending on whether Java or Python is used, where xx is your ID group). Make sure that your code be free from compiling and run-time errors, otherwise your score will be penalized.

*If you have questions about the assignment, contact the teaching assistants (TAs) by email to bdc-course@dei.unipd.it . The subject of the email must be "**HW2 - Group xx**", where xx is your ID group. If needed, a zoom/skype meeting between the TAs and the group will be organized.*

Last update: 29/04/2020

Back to home page