Lab Course: Distributed Data Analytics
Exercise Sheet 3

Prof. Dr. Dr. Schmidt-Thieme, Daniela Thyssens
Information Systems and Machine Learning Lab
University of Hildesheim
Submission deadline: Friday May 20, 23:59PM (on LearnWeb, course code: 3116)

Instructions

Please following these instructions for solving and submitting the exercise sheet.

1. You should submit two items a) a zipped file containing python scripts and b) a pdf document.
2. In the pdf document you will explain your approach (i.e. how you solved a given problem), and present your results in form of graphs and tables.
3. The submission needs to be made before the deadline, only through learnweb.
4. Unless explicitly stated, you are not allowed to use scikit, sklearn or any other library for solving any part. All implementations must be done by yourself.
5. In each lab you have to show your solution works for $P = \{2, 4, 6, 8 \ldots\}$ and provide the timing results whether it is stated in the question or not !

Complex Data Lab: K-means clustering in a Distributed Setting

In this exercise sheet, you are going to apply distributed computing concepts using Message Passing Interface API provided by Python (mpi4py) to implement a clustering algorithm i.e. K-means clustering.

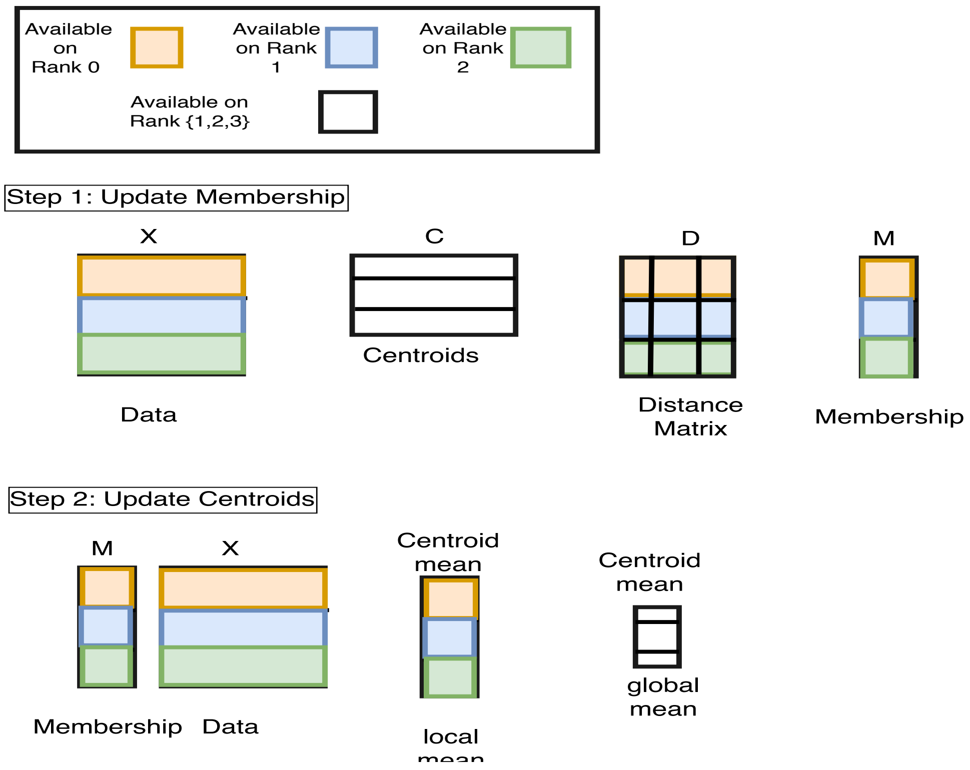
For this task you will use "Absenteeism at work Data Set " `https://archive.ics.uci.edu/ml/datasets/Absenteeism+at+work` provided at UCI repository. **Note:** use .csv file.

Distributed K-means Clustering (20 Points)

The k-means algorithm clusters the data instances into k clusters by using euclidean distance between data instances and the centroids of the clusters. The detail description of the algorithm is listed on slides 1-10 `https://www.ismll.uni-hildesheim.de/lehre/bd-16s/exercises/bd-02-lec.pdf`. However, in this exercise sheet you will implement a distributed version of the K-means. Figure below explains a strategy to implement a distributed K-means.
Suppose you are given a Dataset $X \in \mathbb{R}^{M \times N}$ and a random initial centroids $C \in \mathbb{R}^{M \times K}$, where M are the number of features of a Data instance, N are the number of Data instances and K the number of clusters (K **is a hyperparameter**). Lets assume you want to implement a distributed version with 3 workers. (**Note** your solution should be generic and should work with any number of workers.) In the figure below three workers are given colors i.e. Rank 0 = orange, Rank 1 = blue, and Rank 2 = green. If a data is represented in white color this means it must be available on all the workers. The algorithm progress as

- Initialize K centroids
- Divide Data instances among P workers (X shown in figure, different colors represent parts at different workers)
- Until converge
  - step 1

    * calculate distance of each Data instance from each centroid using the euclidean distance. (populate Distrance matrix shown in the figure)
    * Assign membership of each data instance using the minimum distance in the distance matrix.
  - step 2
    * Each worker calculates the new centroids (local means) using the new membership of data instance.
    * collect updated centroids (local mean) information from each worker and find the global centroids (global mean).
    * redistribute new centroids of clusters to each worker.



Implement K-means 10 Points

You have to implement distributed K-means clustering using MPI framework. Your solution should be generic and should be able to run for arbitrary number of clusters. It should run in parallel i.e. not just two workers working in parallel but all should participate in the actual work.

Performance Analysis 10 Points

You have to do a performance analysis and plot a speedup graph. First you will run your experiments with varying number of clusters i.e. $P = \{1, 2, 4, 6, 8\}$. To plot the speedup graph please follow the lecture slides 15 (`https://www.ismll.uni-hildesheim.de/lehre/bd-16s/exercises/bd-02-lec.pdf`).

Annex

1. Palach, Jan. Parallel Programming with Python. Packt Publishing Ltd, 2014.
2. To time your program you have to use MPI.Wtime(): `http://nullege.com/codes/search/mpi4py.MPI.Wtime`
3. MPI4PY tutorial 1(Python): `http://mpi4py.scipy.org/docs/usrman/tutorial.html`
4. MPI4PY tutorial 2(Python): `https://portal.tacc.utexas.edu/c/document_library/get_file?uuid=be16db01-57d9-4422-b5d5-17625445f351&groupId=13601`

5. MPI tutorial (C/C++): `https://computing.llnl.gov/tutorials/mpi/`
6. Matrix Operation: `http://stattrek.com/matrix-algebra/matrix-multiplication.aspx`