Final project of Big Data - 2022

Boris Teabedjomgwe (boris.teabedjomgwe@enseeiht.fr),
Sixin Zhang (sixin.zhang@toulouse-inp.fr)

November 2022

Introduction

The goal of this project is to solve a Big data problem by using your knowledge of virtual machines, clouding computing and optimization. The students are assumed to use and adapt what are learnt from the class and TP. This project is composed of two parts: the first part is to pre-process a large volume of data, which will then be used in the second part of the project.

Part I: Big data

The objective is to make a pre-processing of the data to generate the sub-set used in the part II which deals with the optimization.

Skills developed. At the end of the project you will be able to: reserve an infrastructure on Amazon with virtual machines (VMs), deploy Spark in cluster mode on these VMs, upload data into a Spark cluster, design and implement a map-reduce algorithm to process data and generate output files as results.

Description. The objective is to perform a pre-processing on the data to extract the sub-set for the optimization step. As we are in a big data class (although the data you will use will not be very huge) , you will deploy Spark on a distributed environment. You will use Amazon AWS to configure your cluster environment. Therefore, the first step is the creation of an account on AWS by each group and associate this account to a Amazon Educate account. Amazon Educate is a free offer which provides free hours of AWS usage to students for educational purposes. A work session will be organized, if needed, for the creation and configuration of your accounts. The next step will be the configuration of a four VM cluster and the deployment of Spark: one name node and three data nodes. You will use t2.micro type VMs which are free. The final steps will be to load data into you spark cluster and implement a program to process them. The archive at [1] contains the input data (out.csv.lz4 in the archive) ant it is compress using lz4 compression algorithm. Take the time to read the Readme file and understand the structure of the data. The total data size is closed to 5 GBytes when decompressed (search on how to decompress a lz4 file on linux). For the map-reduce program, the output will be a file having 3 columns: User Id (an ID for users), movie ID (movies ID), and finally the ratings. The input data contains several non-usable lines (lines without ratings, or username or movie name) and several lines that are repeated (capital letters but with the same information). You have to implement the map-reduce program which allows you to process this raw data file and extract only the information that will be useful for part II of the project. The file rating.csv of the archive contains the expected output (the solution). Use this file to validate your map-reduce program.

Part II: Optimization for collaborative filtering

The goal of this part is to implement a gradient-descent method to solve the collaborative filtering problem which is mentioned on the class. We have a data file which contains the ratings of certain films from various users collected by MovieLens[1]. As each user does not rate all the films, the challenge is to predict unknown ratings. The idea of collaborative filtering is to recommend unseen films to a user, based on the ratings of other users.

Skills developed. You are asked to write a program in Python to solve an optimization problem for collaborative filtering. It should be able to reproduce all the results with a readme file on how to run your code. You also need to write a short report (about 1 page) on what you have done.

Description. From part I, you store the data file in csv. It can then be parsed into a partially observed rating matrix $R = \{r_{i,j}\}_{(i,j)\in\Omega}$, where a row i represents the user $i \in \{1, \cdots, n\}$, and a column $j \in \{1, \cdots, m\}$ represents a film j. Let Ω be a support set of the observed entries (a set of (i, j)). The value $r_{i,j}$ for $(i, j) \in \Omega$ is the rating of the user i on the film j. The idea is then to fill-in the missing values of the matrix R that are outside Ω. To figure out the unknown $r_{i,j}$ for $(i, j) \notin \Omega$, we follow a classical idea which aims to find a low-rank matrix approximation of R. Any rank k matrix can be represented by a product of two matrices $P = (p_i)_{i\leq n}$ and $Q = (q_j)_{j\leq m}$, where $p_i \in \mathbb{R}^k$ and $q_j \in \mathbb{R}^k$ are the columns of P and Q. This problem can formalized as minimizing a regularized MSE loss, given by

$$\min_{P,Q} \frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} \left[(r_{i,j} - q_j^\intercal p_i)^2 + \lambda(\|h(p_i)\|^2 + \|h(q_j)\|^2)\right] \tag{1}$$

where $|\Omega|$ is the cardinality of the (non-empty) set Ω. The function $h(a) = \min(a, 0)$ is applied to each element of a vector a, in order to penalize the situation where the vectors p_i and q_j have negative values (as the rating is always non-negative). The λ is a non-negative regularization parameter. When $\lambda = 0$, (1) is the standard MSE loss, and its squared root is the standard RMSE score.

[1] `https://grouplens.org/datasets/movielens/`, see more on F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19

To help you solve this problem, you are given a jupyter notebook which guides you to

- Pre-process the dataset for cross-validation.
- Solve the optimization problem (1) to find an optimal P^* and Q^*.
- Evaluate your predictions P^*Q^* on the validation set, using the RMSE[2].

Data and code

[1]. Data of Part I: https://drive.google.com/file/d/1thuJ5E7H3tERjZfpbTpK5P5Ck8m3IC93/view?usp=sharing
[2]. Code of Part II: https://cloud.irit.fr/index.php/s/ODmjLE9cJrA2YYO

[2] https://www.kaggle.com/general/215997