Homework 3 (Final Exam): Collaborative filtering using optimized data structures on Spark

Published Date:
Dec 1, 2022

Due Date:
Dec 14, 2022 @2:30pm

Description:

****
This is an individual assignment.
****

Overview and Assignment Goals:

The objectives of this assignment are the following:

- Use Apache Spark to build a Recommender system and predict user ratings
- Compare CF using LSH with a matrix decomposition method
- RMSE will be used to test your submission

Detailed Description:
Develop a Recommendation system to predict as accurately as possible the user item ratings.

Collaborative Filtering (CF) systems measure similarity of users by their item preferences and/or measure similarity of items by the users who like them. For this CF systems extract Item profiles and user profiles and then compute similarity of rows and columns in the Utility Matrix. (In this assignment you are given a number of ratings, from which it is possible to build a utility matrix.) The main tasks for this assignment are:

- **Item-item or User-user CF using LSH to find the similarity between items (or users)**
- **CF using latent factors (matrix decomposition)**
- **Compare the RMSE score on the test data for the two implementations**
- **Report the training time for both implementations (LSH index building time for LSH)**
- **Report the time it takes to predict ratings on the test data or both methods**

Use various similarity measures for finding the most similar items or users. We encourage you use functions available in spark libraries for similarity computation, LSH, SVD decomposition etc. **However, you cannot use the spark ALS package!** Perform these tasks in parallel on multiple cores is required as the dataset is quite large.

The goal of this assignment is to allow you to develop collaborative filtering models that can **predict the rating** of a specific item from a specific user given a history of other ratings.

To evaluate the performance of your results we will use the Root-Mean-Squared-Error (RMSE).

Caveats:

+ Use the data mining and data structure knowledge you have gained until now, wisely, to optimize your results.

+ The default memory assigned to the Spark runtime may not be enough to process this data file, depending on how you write your algorithm. If your program fails with
`java.lang.OutOfMemoryError: Java heap space`
then you'll need to increase the memory assigned to the Spark runtime. In general spark uses --driver-memory to set the runtime memory. On the HPC we will be running spark on top of Slurm, thus spark can only get as much memory as slurm allocates.

Data Description:

The training dataset consists of 85724 ratings and the test dataset consists of 2154 ratings. We provide you with the training data ratings and the test ratings are held out. The data are provided as text in train.dat and test.dat, which should be processed appropriately.

train.dat: Training set (UserID <comma separator> ItemID <tab separator> Rating (Integers 1 to 5) <tab separator> Timestamp (Unix time stamp).
test.dat: Testing set (RatingID<comma separator> UserID <comma separator> ItemID, no rating provided).
format.dat: A sample submission with 2154 ratings being all 1 (The one values shall be replaced with your predicted ratings in the order of the test.dat file).

Deliverables:

- Valid submissions (code and output) to the Leader Board website: http://coe-clp.sjsu.edu/
- **Canvas submission of the report:**
- Valid submissions to this assignment for both tracks on CLP (i.e. your predictions) (follow the format.dat file)
    - There are two tracks: 1) The LSH track; 2) The Latent factors track
- The report should describe the steps you followed for developing your final solution. Be sure to include the following in the report:
    - Name, SJSU ID

    - Data pre-processing steps and other approaches.
    - A description of the algorithm(s) you used as well as any associated parameters.
    - Results:
        - Compare the RMSE score on the test data for the two implementations
        - Report the training time for both implementations (LSH index building time for LSH)
        - Report the time it takes to predict ratings on the test data or both methods

- Compare the RMSE score on the test data for the two implementations
- Report the training time for both implementations (LSH index building time for LSH)
- Report the time it takes to predict ratings on the test data or both methods

- Failure to follow the above guidelines (missing files, etc.) will cost you points.

Rules:

- **This is an individual assignment**. Discussion of broad level strategies are allowed but any copying of prediction files and source codes will result in an honor code violation.
- The recommender system should be implemented in Spark. Feel free to use the programming language of your choice for this assignment (Python, Scala, or Java).
- The public leaderboard shows results for 50% of randomly chosen test instances only. This is a standard practice in data mining challenges to avoid gaming of the system. The private leaderboard will be released after the submission deadline, based on all the entries in the test set.
- In a given day (00:00:00 to 23:59:59), you are allowed to submit a prediction file only 5 times.
- The final ranking will always be based on the last submission, not your best submission. Carefully decide what your last submission should be.
- Once you select the final submission file, submit the source code that produced that result to Canvas.
- **No late submission will be allowed.**

Grading:

Grading for the Assignment will be split on your implementation (40%) and report (60%).

Extra credit (1% of final grade) will be awarded to the top-3 performing algorithms on each track. However, before assigning the extra credit, the code will be re-run to verify that the submitted solution file corresponds to that generated by your code.

Files: On Canvas, you can find

- *Training Data:* train.dat
- *Test Data:* test.dat
- *Format File:* format.dat