INF 553 Fall 2017

Assignment 3 Recommendation System

Deadline: 10/25/2017 11:59 PM PST

Assignment Overview

This assignment contains two parts. First, you will implement a Model-based Collaborating Filtering(CF) recommendation system using Spark MLlib. Second, you will implement either a User-based CF system or Item-based CF system without using a library. The datasets you are going to use are the MovieLens datasets. The task sections below will explain the assignment instructions in detail. The goal of the assignment is to make you understand how different types of recommendation systems work and more importantly, try to find a way to improve the accuracy of the recommendation system yourself.

Write your own code!

For this assignment to be an effective learning experience, you must write your own code! I emphasize this point because you will be able to find Python implementations of most or perhaps even all of the required functions on the web. Please do not look for or at any such code! **Do not share code with other students in the class!! Please note that this is the system they can improve upon for the extra credit recommendations contest.**

Datasets

The Movie Lens datasets can be found in the following link:
https://grouplens.org/datasets/movielens/

You will download dataset: ml-latest-small.zip. Once, you extract the zip archive, you will find multiple data files. In this assignment, we will only use *ratings.csv*. However, you can combine other files to improve the performance of your recommendation system.

You will also download testing file from Blackboard: *testing_small.csv.* The testing dataset is a subset of the original dataset, each containing two columns: <userId> and <movieId>. The file *testing_small.csv (20256 records)* is from *ratings.csv* in *ml-latest-small*. Your goal is to predict the ratings of every <userId> and <movieId> combination in the test files. You CANNOT use the ratings in the testing datasets to train your recommendation system. Specifically, you should first extract training data from the *ratings.csv* file downloaded from Movie Lens using the testing data. Then by using the training data, you will need to **predict** rate for movies in the testing dataset. You can use the testing data as your ground truth to evaluate the accuracy of your recommendation system.

Example: Assuming ratings.csv contains 1 million records and the testing_small.csv contains two records: (12345, 2, 3) and (12345, 13, 4). You will need to first remove the ratings of user ID 12345 on movie IDs 2 and 13 from ratings.csv. You will then use the remaining records in the ratings.csv to train a recommendation system (1 million – 2 records). Finally, given the user ID 12345 and movie IDs 2 and 13, your system should produce rating predictions as close as 3 and 4, respectively.

Task1: Model-based CF Algorithm (30%)

In task1, you are required to implement a Model-based CF recommendation system by using Spark MLlib. **You can only use Scala to implement this task**. You can learn more about Spark MLlib by this link: http://spark.apache.org/docs/latest/mllib-collaborative-filtering.html

You are going to predict ratings for small testing dataset mentioned above. In your code, you can set the parameters yourself to reach a better performance. You should aim to make your classifier as accurate as possible, within the time constraints, **your code must finish in less than 45 seconds**. **Please print the time taken by your program** on the command line from reading the files to the output of the predicted files.

After achieving the prediction for ratings, you need to compare your result to the correspond ground truth and **compute the absolute differences**. You need to divide the absolute differences into 5 levels and count the number of your prediction for each level as following:

```
>=0 and <1:14239
>=1 and <2:4449
>=2 and <3:1166
>=3 and <4:367
>=4:35
RMSE = 1.0993961216250527
The total execution time taken is 1.5678 sec.
```

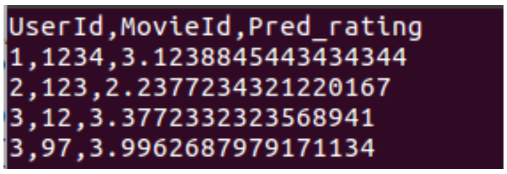
Additionally, you need to compute the RMSE (Root Mean Squared Error) by using following formula:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i}(Pred_i - Rate_i)^2}$$

Where $Pred_i$ is the prediction for movie i, $Rate_i$ is the true rating for movie i, n is the total number of the movies.

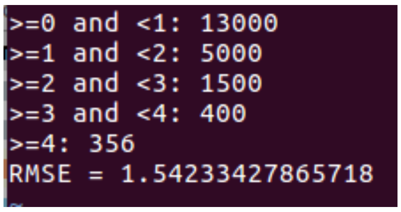
Result format:

1, **Save the predication results in a text file**. The result is ordered by <userId> and <movieId> in ascending order. This is a sample case.



```
UserId,MovieId,Pred_rating
1,1234,3.1238845443434344
2,123,2.2377234321220167
3,12,3.3772332323568941
3,97,3.9962687979171134
```

2, **Print the accuracy information** in terminal, and **copy this value** in your description file.



```
>=0 and <1: 13000
>=1 and <2: 5000
>=2 and <3: 1500
>=3 and <4: 400
>=4: 356
RMSE = 1.54233427865718
```

Task2: User-based CF / Item-based CF Algorithm (70%)

In this part, you are required to implement **either** a User-based CF recommendation system **or** Item-based CF recommendation system **with Spark**. You can use **Scala** or **Python** for this task.

You are going to predict for **the small testing datasets** mentioned above. You can make any improvement to your recommendation system: **speed, accuracy** (e.g., Hybird approaches). It's your time to design the recommendation system yourself, but first you need to beat the baseline.

After achieving the prediction for ratings, you need to compute the accuracy in the same way mentioned in Task 1. Result format is also the same as Task1.

Description File

Please include the following content in your description file:

1. Explain the implementation how you have handled the missing users, outlier rating when predicted value exceeds [0,5] range.
2. Describe how to run your program for both tasks please follow the order of the arguments:
   For example, to run jar package, you should to write the commend as:
   `Priyambadas-MacBook-Pro:spark root# ./bin/spark-submit --class Priyambada_Jain_Task1 Priyambada_Jain_Task1.jar ratings.csv testing_small.csv`
3. The accuracy for both tasks. The format is described above.
4. If you make any improvement in your recommendation system, please also describe it in your description file.

Submission Details

Your submission must be a .zip file with name: *<Firstname>_<Lastname>_hw3.zip*
Please include all the files as following:
1. A description file : *<Firstname>_<Lastname>_desription.txt (or pdf)...*
2. A Scala script for task1: *<Firstname>_<Lastname>_task1.scala*
3. A jar package for task1: *<Firstname>_<Lastname>_task1.jar*

4. One result file for task1 and name it as:

    <Firstname>_<Lastname>_result_task1.txt

5. A Scala or Python script for task2: *<Firstname>_<Lastname>_task2.scala* or *<Firstname>_<Lastname>_task2.py*

6. If you use Scala in task2, please submit the jar package as well and name it as *<Firstname>_<Lastname>_task2.jar*

7. One result file for task2: <Firstname>_<Lastname>_result_task2.txt

Grading Criteria:

1. If your programs cannot run with the commands you provide, your submission will be graded based on the result files you submit, and there will be a 80% penalty
2. **If the files generated are not sorted based on the specifications or does not follow the format for result files there will be 20% penalty.**
3. If your program generates more than one file, there will be 20% penalty.
4. **If your prediction result files miss any records, there will be 30% penalty**
5. **If you don't provide the source code, especially the Scala scripts, there will be 80% penalty.**
6. **If you don't state inside the description file that how to run your code or the accuracy result, there will be a penalty of 30%.**
7. There will be 20% penalty for late submission.
8. **We will grade on this assignment based on your accuracy**. **You are expected to beat the baseline in both tasks to receive full credit.**
9. **There will be a testing dataset against which we will check your code so don't try to beat the baseline by overfitting the model. If your program is failed to run against this test dataset 30% marks will be deducted.**

Baseline

Ranges	Task 1	Task2
>=0 and <1	13195	13937
>=1 and <2	5027	4878
>=2 and <3	1525	1211
>=3 and <4	407	218
>=4	102	12
RMSE	1.21686778	1.039897994

Appendix ALS

Kindly refer to the following links for more details about ALS Matrix Factorization:

- https://blog.insightdatascience.com/explicit-matrix-factorization-als-sgd-and-all-that-jazz-b00e4d9b21ea
- https://bugra.github.io/work/notes/2014-04-19/alternating-least-squares-method-for-collaborative-filtering/