Recommender Systems: Homework

ALEJANDRO BARRIENTOS SESSAREGO[1,2]

[1]*Joint ALMA Observatory, Santiago, Chile*
[2]*Universidad Técnica Federico Santa María, Valparaíso, Chile*
September 16, 2015

Abstract

In this report, the results of the Homework for the Recommender Systems course will be presented, an exploratory analysis of the input data will be made, then the implemented algorithms (Popularity, User-Based Collaborative Filtering, Item-Based Collaborative Filtering and Slope One) will be explained, finally the testing results evaluated at the Codalab competition will be presented. The best results for rating were obtained using the Slope One method, the best ones for Ranking were obtained using Wilson Score.

Keywords: Recommender Systems, User Based Collaborative Filtering, Item-Based Collaborative Filtering, Slope-One

1 Introduction

The objective of this homework, is to learn how to implement four of the basic Recommender Systems that we discussed in class, to be able to construct a prediction and a list of recommendations based on an input file, provided by the professor.

2 Exploratory Analysis

2.1 Dataset

The data set is a Comma Separated Values (CSV) file, that contains originally 4 columns, userID, itemID, styleID and rating. There are 44379 rows. Here are some further details:

Number of rows in dataset	44379
Number of users in dataset	8320
Number of distinct Items in dataset	1836
Number of distinct Styles in dataset	210
Average Rating in dataset	3.865
Ratings standard deviation	0.712
Sparsity	0.997
Average number of ratings per item	24.171
Average number of ratings per user	5.334

Table 1: Statistics of the dataset

2.2 Information extracted from the dataset

some charts, I only selected the top 20 so the charts can be readable.

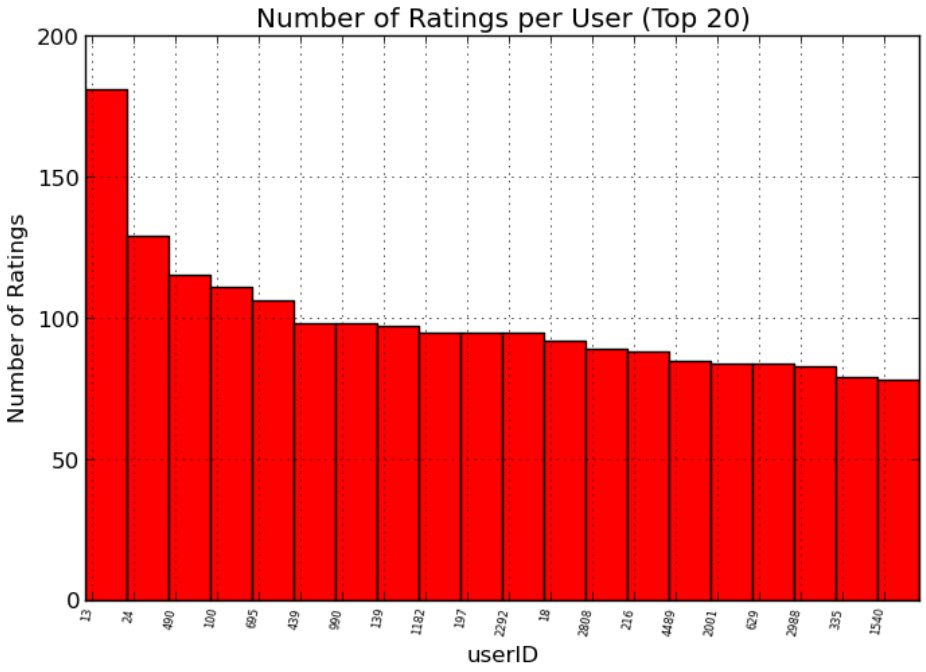

Figure 1: Number of Ratings per user (top20)

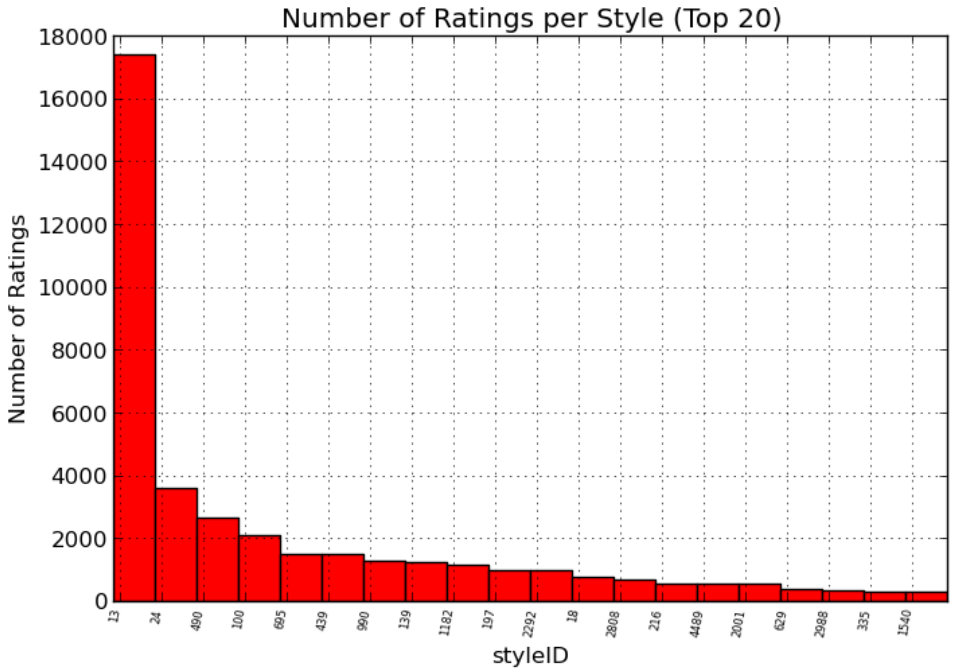

Figure 2: Number of Ratings per Style (top20)

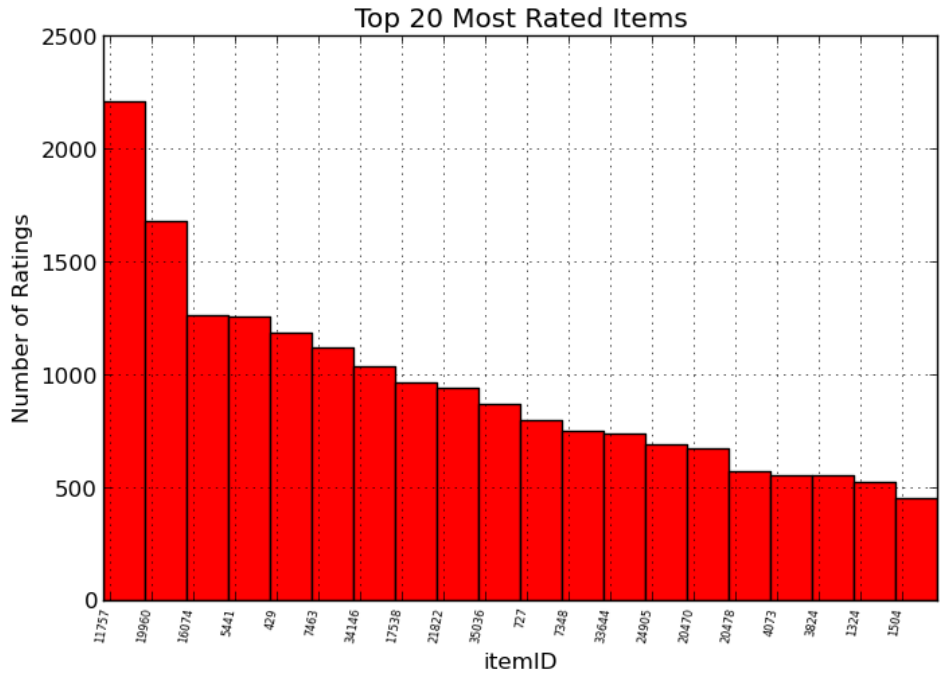

Figure 3: Most Rated Items (top20)

From observing these statistics one could say that it might be possible to cluster the items based on their styleID to calculate similarities or to recommend similar items, however I decided not to go with this approach, because it

might make the calculations more complex. However, the calculation of the similarity matrices was in some cases very, very long, so I probably should have considered this at some point.

---

3 Development of the solution

For all the algorithms, some things are in common:

- The dataset is loaded into a SQLite database, residing in memory, this allows to obtain some statistics and rows based on certain criteria depending on each algorithm.
- In order to construct an user/item rating matrix, the user and item names were mapped to 4 dictionaries, for users, userMap, which takes a user name and returns a correlative index in the matrix; the same way you can use the userAntiMap, which takes an index and returns the user Name. This is the same for the items, with itemMap and itemAntiMap.
- A userRatings matrix was constructed of size (users x items) that allows you to retrieve a particular rating for a particular item of a particular user.
- A dictionary of users vs average rating was constructed, to be used in the different algorithms that require it when creating a prediction.
- The training data was also loaded into a numpy array, to obtain some extra statistics.

3.1 User Based Collaborative Filtering

Description

This method takes into consideration the opinions of similar users in order to make a prediction.

I chose Pearson similarity to create the similarity matrix between users. It was necessary to construct the entire 8320x8320 matrix so I can find out the k closest neighbors, k is a parameter that can be moved in the algorithm to include more neighbors to calculate the prediction. It took me around 27 hours to build the similarity matrix, considering that the similarity is a symmetric matrix, that is, sim(i,j) = sim(j,i).

The script was constructed the following way:

- Load pearson Similarity Matrix into memory from csv file.
- Load data to rate into memory.
- Load data to rank into memory.
- Open the output files for rating and ranking with their respective writers.
- For each row in data to rate, try to calculate a prediction with ubcf, if it is not possible, assign the default prediction which is 0.

To generate the UBCF prediction, the following steps were made:

- The function receives the userIndex and the itemIndex.
- Load the full neighborhood of the active user (from the pearson Matrix, calculated offline).
- Select the first K users from the neighborhood, sorted by similarity, from higher to lower.
- For each neighbor, create a numerator and a denominator.
- The numerator is made by adding similarity of the active user with its neighbor and multiply by the difference between the rating that the neighbor gave the item, minus the average rating of the neighbor.
- The denominator is calculated just by adding the similarities between the active user with its neighbors.

- If the numerator is zero, then there is no correlation between the users, just return the average rating of the active user.

In order to generate the ranking, I decided to calculate a full matrix of users vs items, predicting all the items that had not been rated by the users. this gave me a full 8320x1836 matrix that I could use to find out the best rating items for each user and make a good prediction of items that the user should like. Sounds like a lot of work, considering that the users in average only rate around 5 or 6 items, however I hope that this approach, even slower should help me get better predictions.

The next step is to create a top 10 ranking for each user in the data to rank file, the way to do this would be:

- For each user in data to rank:
- Verify if the user exists, since the list is short we should get results quick.
- If the user exists, get all the ratings for that particular user from the predicted full matrix.
- Sort the row based on predicted rating, from higher to lower
- Get the top ten items indexes.
- For each item index, get the item name from the map.

3.2 Performance and Results

According to codalab:

User	nDCG	RMSE	MAP
MadScientist	0.036 (1)	3.350 (4)	0.000 (1)

Table 2: Testing Results in Codalab: UBCF

4 Item Based Collaborative Filtering

4.1 Description

This approach takes only the similarity between items in order to make a prediction.

The structure of this script is similar to the User-Based one, with a few differences:

- Instead of using a User similarity matrix, in here I used a Item Similarity matrix, with Cosine Based distance, instead of Pearson.
- The selected calculation to build the prediction was Weighted Sum.
- For each user to build a top ten ranking, I selected the best rated item from the user, and based on that item, built a top ten list with the most similar items.

In order to construct the Item similarity matrix, I had to generate a vector with the common users that rated two different items, so the vectors are of similar size, then used the cosine distance function integrated into the scipy library to get the similarity value. This took several days to complete, initially I had just used the columns from the user ratings and compared column pairs, however, as most columns were filled with zeroes, the similarities between vectors was affected, that is why I had to redesign the script so it construct vectors without zeroes.

4.2 Performance and Results

The approach of using columns with zeroes included provided the following results:

RMSE:	3.67307450318
nDCG:	0.0160025759418

Table 3: Initial Item-Based results

5 Popularity Ranking

5.1 Description

Used Wilson Score to recommend the same top 10 items to all the users. Implementation was as described on Evan Miller's blog, it was implemented in the file statistics.py as it was a fast algorithm, did not require any matrices to calculate, just classify the ratings for each item into positive (rating >= 2.5) and negative (rating < 2.5). The blog provided the calculation of the score. I then sorted the score and wrote the ranking file.

5.2 Performance and Results

ItemID	Wilson Score
16074	0.993022995371
11757	0.992264075529
5441	0.991820384574
21822	0.990658782155
7348	0.990303686162
19960	0.989083816779
34146	0.984807362915
20478	0.984641119528
47658	0.983976654971
28350	0.982604726646

Table 4: Top Ten Items by Wilson Score

After calculating the Top Ten items by Wilson score, I recommended this same list to all the users in the data to rank file.

The codalab checker returned nDCG: 0.0732512279026

6 Slope One

6.1 Description

In this algorithm I took the easiest path, I chose the average rating of the user, to predict what score the user will put to the next item. Based on this, I calculated the entire matrix of users / items.

To generate the ranking, for each user to rank, I sorted the slope one predicted full rating matrix, and created a top 10 list per user.

6.2 Performance and Results

RMSE	0.724719576892
nDCG	0.0439507367416

Table 5: Slope One Codalab Results

7 Conclusions

According to the testing results, the best combination would be Slope One for the item predictions and wilson score for the ranking.

Acknowledgments

The Atacama Large Millimeter/submillimeter Array (ALMA), an international astronomy facility, is a partnership of Europe, North America and East Asia in cooperation with the Republic of Chile. ALMA is funded in Europe by the European Organization for Astronomical Research in the Southern Hemisphere (ESO), in North America by the U.S. National Science Foundation (NSF) in cooperation with the National Research Council of Canada (NRC) and the National Science Council of Taiwan (NSC) and in East Asia by the National Institutes of Natural Sciences (NINS) of Japan in cooperation with the Academia Sinica (AS) in Taiwan. ALMA construction and operations are led on behalf of Europe by ESO, on behalf of North America by the National Radio Astronomy Observatory (NRAO), which is managed by Associated Universities, Inc. (AUI) and on behalf of East Asia by the National Astronomical Observatory of Japan (NAOJ). The Joint ALMA Observatory (JAO) provides the unified leadership and management of the construction, commissioning and operation of ALMA.