Ryan Wedoff

Zongsheng Sun

Professor Stewart

4/18/16

# Group 3: Recommendation Systems Report

Companies like Netflix and Amazon use recommendation systems to predict products that the users will use next. The goal of this project was to replicate the algorithms used by these companies.

For our recommendations system, we emulated Netflix and their movie recommendations. Essentially, a user provides ratings for various movies and with that data, we predicted which movies they will want to watch, and “like”, next.

### Computation

Our data was found on Grouplens.org (citation below). The data was a list with user ids, ratings, and movies ids. With this data, we created a matrix, with each row being a user, each column being a movie, and each entry being a rating. This was a sparse matrix because not every user rated every movie. With our sparse matrix, we used Python’s Numpy and ran a singular value decomposition on the matrix. We also calculated the general weighted average rating for each movie and the “exceed average rating” for each user. The weighted average was calculated by

This method was used so that as the number of ratings for the movie goes up, the weight also increases. The weight approximately equals 1 when the number of ratings for the movie is sufficiently large.

The “exceed average rating” is the average rating that a user has for each movie. This value determines how far off the user is from the general average rating for each movie. With singular values decomposition, we took the orthogonal matrix U, the diagonal of the singular value vector and the transpose of the right singular values and multiplied them together to get the ratings for each person.

We then added in the “exceed average rating” for the chosen user. We also subtracted out the movies that a user has already rated.   
Lastly, we added the weighted average for each movie. The final prediction algorithms was



This final prediction was a list of recommended ratings for each movie, where the index of the rating represents the movie. We sorted this array and kept the indices of the top ten recommended movies. We used the movie file and obtained the titles of the top ten recommended movies for each user. The results for each user were saved in a separate file for quick access.

### Frontend Work

On the frontend side, we created a Web Application using Python’s Flask library. The web application, if in debug mode, computes the whole SVD and ratings. Otherwise, the web application reads in the saved results from the previous computation. This was done to increase web page loading times.

### Testing

To test our algorithm, we used the 100,000 Ratings dataset from Grouplens.org. The 100,000 Ratings dataset came broken up into a test and base set. We ran our algorithms on the base set, calculating the recommendations for every user. We then calculated the total mean squared error to determine how well our algorithm worked. We ran this test of 6 different splits on the 100,000 Ratings dataset.

We ran the same test on various parts of our algorithm to determine how we can improve our work. We started with just the SVD on the original matrix. Then we tested including the unweighted average movie rating. Then the “exceed average rating”, and lastly, the weighted average for each movie.

### Testing Results

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SVD on | u1 | u2 | u3 | u4 | ua | ub | Average |
| Original matrix | 2.87659289 | 2.76577005 | 2.69006711 | 2.71137178 | 2.8481686 | 2.86360473 | 2.7925959 |
| Unweighted movie average rating | 0.99705938 | 0.98746208 | 0.98036265 | 0.97728907 | 1.00559303 | 1.01613083 | 0.9939828 |
| Weighted rating per user (AER) | 0.95048582 | 0.93893182 | 0.9387754 | 0.93524512 | 0.95632111 | 0.97042699 | 0.9483644 |
| Weighted Avg. Movie Rating | 0.94751591 | 0.93480794 | 0.92772242 | 0.92600982 | 0.9530763 | 0.96527947 | 0.942402 |

For our testing, we calculated the mean squared error for each dataset. Our best model was the last test with the “exceed average rating” and the weighted average movie ratings included. The mean squared error shows that we are making predictions on people will be rating the movies as with an error of ~1 star. While these results may not seem the best, they do provide excellent direction in if a user will like or dislike a movie in general.

While the weighted average didn’t improve the mean squared error all that much; however, the weighted average greatly improved the titles that were recommended for each user. Instead of titles that have only been rated by only a handful of users being recommended, the weighted average recommended popular movie titles that have been rated by many people, thus making our algorithm, from a user stand point, more effective.

### Observations

For most of our work we ran the 100,000 rating data set. We found that this data set was too small. We made this conclusion, before we added the weighted average, because we noticed that one movie was being recommended for almost every user. This movie, *Santa with Muscles*, was only rated by 2 people. These 2 people both rated this movie a 5 out of 5. According to our old model, *Santa with Muscles* is one of the best movies and it is recommended that everyone watches it. According to IMDB, *Santa with Muscles* was rated a 2.4/10 based on over 7,000 ratings. Upon testing, we solved this issue with a larger dataset and the weighted average. With the weighted average solution, a movie that is only rated twice, like *Santa with Muscles* won’t be as important as a movie that has been rated lots of times like *Titanic*.

We also tried our work on the 1 million ratings dataset. This set done on a Windows 10 Laptop with 8 GB of RAM took some time, but finished in about 10 minutes. We also tested the 10 million and the 20 million ratings dataset, and that was too large for the laptop and Python threw an out of memory exception. Solutions for this would be to try other computers with more memory, and to work the algorithm on a GPU to increase processing speed.

### Conclusions

The goal of the project was to come up with a recommendation system that emulates the algorithms used by Netflix. With our algorithm and application we demonstrated the process of creating a recommendation system that does indeed emulate the process used at Netflix.

Citations:

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872