Recommendation System from the MovieLens 10M Dataset

Matthew White

April 2025

1 Introduction

The focus of this project is to create a recommendation system for movies based on the MovieLens 1M dataset. This dataset consists of one million entries, where each entry consists of a user ID, a movie ID, and a rating on a 5-point scale. There are approximately 72,000 unique users and 10,000 unique movies. Each user has rated at least 20 movies. The dataset was published in 2009 and only contains movies that were released beforehand. GroupLens has published other versions with differing numbers of entries along with new movie titles as they were released. For this project, the 10M dataset was chosen because it was the largest dataset that could be used without a different software stack (see §2). The topic of recommendation systems has been studied extensively. One notable publication on the subject, titled "Application of Dimensionality Reduction in Recommender System – A Case Study" (Sarwar, et al., 2001), provides an overview of numerical methods commonly used in recommendation systems. The goal of this project is to apply these methods and evaluate their efficacy on the given dataset.

2 Methods

For the implementation of this project, we used Python 3 with the following modules: Pandas, NumPy, PyTorch, MatPlotLib, and Sci-Kit Learn.

The first step in creating our recommendation system was to change the shape of the data by pivoting. In this case, the result of the pivoting is a matrix R, where Ri,j contains the ith user's rating on the jth movie. The implementation of the pivoting function in the Pandas module is limited in that it cannot create a matrix with more than 231-1 entries (the maximum value for a 32-bit integer, approximately 2.1 billion). Our pivoted matrix has approximately 750M entries, which is well below this threshold.

Although our ratings matrix is very large, it is still 98% sparse. To address this problem, each empty entry was filled with its corresponding column average. Next, each row was normalized by subtracting the row average from each

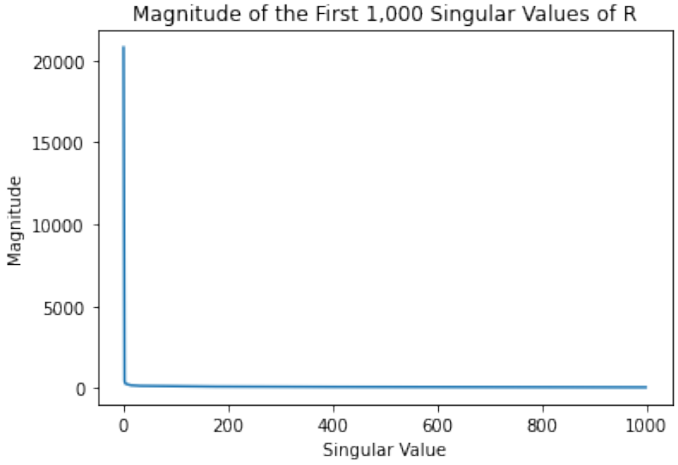

Figure 1: Magnitude of the First 1000 Singular Values of R

entry. With the normalized matrix ready, the next step was to find a low rank approximation of this matrix using its singular value decomposition (SVD).

Using PyTorch, we found the SVD while making use of GPU-acceleration to speed up the process. To further speed up the computation, we set the full_matrices parameter to false, thereby computing a lower rank approximation of the SVD from the start. Still, these computations took approximately 30 minutes to complete. After finding the SVD, we inspected the magnitudes of the singular values to determine what rank to use for the low-rank approximation.

Although we see a large drop off in the magnitude of singular values in just the first 10 values, we proceeded to use a rank of 500 for our low-rank approximation. With our low-rank approximation ready, we could then create our new prediction matrix, P defined as follows:

$$P := M + U_k\sqrt{S_k}\sqrt{Sk}V^T$$

where M is the m x n matrix where each row is the average rating associated with that user, $U_k\sqrt{S_k}$ is the user space, and $\sqrt{S_k}V^T$ is the movie space. The final step in creating the recommendation system was to generate predictions using P. The first method we used was to simply recommend the movies with the highest predicted score that the user had not yet rated. This method was based on the assumption that users have only seen the movies that they had rated. Upon inspecting the recommendations for a few random users, we found that the recommendations were very similar to each other. In fact, we found that the movies "Satan's Tango" (1994), "Shrek the Halls" (2007) and "Shadows of Forgotten Ancestors" (1964) appeared in 80% of the top 10 recommendations for a random sampling of 10 users. Although this was only a small sample, it was enough for us to pursue other methods for generating recommendations.

The next approach to recommendations we used was to recommend movies

based on those highly rated by similar users. To do this we found the cosine similarity between users from the user space. We then recommended the movies based on the movies with the highest average ratings from the 100 most similar users. While these recommendations are difficult to evaluate without user testing, we believe that this method provides better results than the previous approach.

3 Observations

As mentioned above, recommendation systems are difficult to evaluate without deploying to users and observing how they interact with the recommendations. Still, we were able to find a few statistics in an attempt to capture our results.

Firstly, we found the root mean square error (RMSE) of our predictions, defined as $\frac{||R_{avg}-P||_F}{\sqrt{mn}}$, where R_{avg} is the ratings matrix imputed with the movie averages, m is the number of users, and n is the number of movies. We found the RMSE to be .0724. While this may seem low, thereby indicating good predictions, this is likely attributed to the fact that most of the entries are close to the average ratings. We also found the mean absolute error from the average value of $R_{avg} - P$. This error came out to be .0157. Again, this is a very small value for the same reason as the RMSE.

The next set of statistics we used to evaluate our recommendation methods was to look at the item coverage of our two methods. Item coverage in this case is defined as the number of unique movie titles recommended for all of the users divided by the total number of movies in the dataset. The coverage of our recommendations based on the predicted ratings was .0658, which confirms our suspicions that this method was only recommending a very small set of movies. On the other hand, our recommendation method based on similar users had a coverage of .6182, suggesting that this method was capable of recommending a much larger number of movies.

4 Conclusion

While this recommendation system shows promising results, there are still areas for improvement. The best way to further improve the system is to have real feedback from users on the quality of the recommendations. Unfortunately, this is not possible for this dataset as the users are anonymous. Other areas of improvement include creating a hybrid recommendation system that uses the collaborative filtering methods alongside a content-based approach that gives recommendations based on the attributes of the movies themselves, without respect to user interactions with them.