Building an Offline Movie Recommender System: Final Project Report *

Janaan Lake

May 3, 2019

1 Introduction

Recommender systems are among the more popular applications of data science today. They are used to predict the rating or preference that a user would give to an item. Almost every major tech company has applied them in some form: Amazon uses it to suggest products to customers, YouTube uses it to decide which video to play next on autoplay, and Facebook uses it to recommend pages to like and people to follow. For some companies such as Netflix and Spotify, the business model and its success revolves around the efficacy of their recommendations. In fact, Netflix even offered a million dollars in 2009 to anyone who could improve its system by 10% [1].

For the final class project I created an offline movie recommender system using the popular MovieLens dataset found at *https://grouplens.org/datasets/movielens/*. This dataset contains user, movie and ratings information. The ratings data consists of 100,000 samples, which was divided into a training set (80%) and a test set (20%). Using this data I constructed several models to predict the rating a user would give a movie. A description of the algorithms used and their results follow.

2 Methods

Machine learning tools have proven very effective in implementing recommender systems. Many of these algorithms employ a utility matrix, which is a m x n matrix that contains m rows for each user and n columns for each movie. This matrix is populated by the ratings in the training set that are known, which creates a very sparse matrix in most cases. The rest of this matrix is filled in using a variety of methods. The techniques I used for my project include simple estimation, content-based and collaborative filtering methods, matrix factorization, neural networks, and a hybrid of some of these methods.

2.1 Simple Approach

The most basic approach is to suggest items that are liked or used by the most number of users. In this vein, the average of each movie's rating in the training set is used as the predicted rating. These results are used as the baseline for measuring any improvements in the other methods.

*CS 5350/6350; Spring 2019 Instructor: Dr. Shandian Zhe, University of Utah

2.2 Content Based

Content-based algorithms focus on the similarity of the items being recommended. If a user likes an item, then the assumption is that the user would like a similar item. Certain features of items are stored in a vectorized format. Then the relationships of these vectors are measured using different techniques. One such measure is the cosine similarity metric, which calculates the angle between the two vectors. The smaller the angle the more similar the vectors are determined to be [2]. The movie genres are the attributes I used to compute the similarity matrix. For example, let m_{ij} denote the similarity between movie i and movie j. The cosine similarity matrix is then:

$$cos(m_i, m_j) = \frac{m_i m_j^T}{\|m_i\| \, \|m_j\|}$$

If r_j represents the average rating of movie j in the training set, then the prediction for movie i is calculated by the following formula:

$$r_i^* = \frac{\sum_j m_{ij} \cdot r_j}{\sum_j m_{ij}}$$

2.3 Collaborative Filtering

Collaborative filtering algorithms are some of the most widely-used methods for recommender systems. These techniques do not require any information about the items or the users themselves. Rather, they are based on extracting the relationships among users and items. They make the assumption that people who agreed in the past will agree in the future and that users will like similar kinds of items as they liked in the past. There are two main types of collaborative filtering methods: user-item filtering and item-item filtering. A user-item filtering method takes a particular user, finds users that are similar to that user based on similarity of ratings and recommends items that the similar users' liked. In contrast, item-item filtering will take an item, find users who liked that item, and find other items that those users also liked.

For the user-item based approach, a cosine similarity matrix between users is calculated. The approach is very similar to the content-based approach except that the similarity matrix for users is determined by the utility matrix. Let $u_{i,k}$ denote the similarity between user i and user k in the utility matrix. Let $v_{i,j}$ denote the rating that user i gives to item j with $v_{i,j} =?$ if the user has not rated that item. The cosine similarity matrix between users is calculated as follows:

$$cos(u_i, u_j) = \frac{u_i {u_j}^T}{\|u_i\| \, \|u_j\|}$$

The cosine similarity matrix is then used as the weights for the movies that are already rated to determine a score for each movie that isn't rated. The prediction is shown by the following equation:

$$v_{ij}^* = \frac{\sum_{v_{kj} \neq ?} u_{jk} v_{kj}}{\sum_{vk \neq ?} ujk}$$

I used two approaches for this method: one using the cosine similarity matrix weights of all users and one using the cosine similarity weights for the closest 50 users to the user for which the prediction was made.

A similar methodology was used for the item-item filtering. First, the cosine similarity matrix was computed on the items based on the utility matrix. This similarity matrix was used as weights for each user that had rated the movie. Again, all of the movie data was used and then only the 50 most similar movies were used to weight the score[3].

2.4 Matrix Factorization

The next machine learning technique employed is based on matrix factorization. Matrix factorization algorithms work by decomposing the utility matrix into the product of two lower-dimensionality rectangular matrices, M and U, similar to the decomposition used in SVD. The matrix M has a row for each user, while the second matrix U has a column for each item. The columns of matrix M and the rows of matrix U are referred to as the latent factors. The number of latent factors can change the expressive power of the model. The predicted rating user u will give to item i is computed as:

$$\hat{r}_{ui} = \sum_{\mathrm{f}=0}^{nfactors} M_{u,\mathrm{f}} U^{T}_{\mathrm{f},i}$$

The matrices U and M can be determined if the vectors p_u and q_i are found, such that p_u makes up the rows of M and q_i makes up the columns of U^T. Finding such vectors p_u and q_i for all users and items can be done by solving the following optimization problem for the known ratings in the training set:

$$\min_{p_u, q_i} \sum_{r_{ui} \in R} \left(r_{ui} - p_u \cdot q_i\right)^2$$

Stochastic gradient descent is used to approximate a solution [4][5].

2.5 Neural Network

The utility matrix can be learned using a neural network. For this approach, the cross entropy loss function is used. The cross entropy formula takes in two distributions: $p(x)$, the true distribution, and $q(x)$, the estimated distribution. Both of these are defined over the discrete variable x and are given by:

$$H(p, q) = -\sum_{\forall x} p(x) log\ q(x)$$

Therefore, the optimization function used for the neural net is:

$$\min_{N} \sum_{\forall x} p(x) log\ q(x)$$

where N represents the number of samples and $q(x)$ represents the tanh classifier. The neural network in my final project is a fully-connected neural network with 3 hidden layers, each layer a

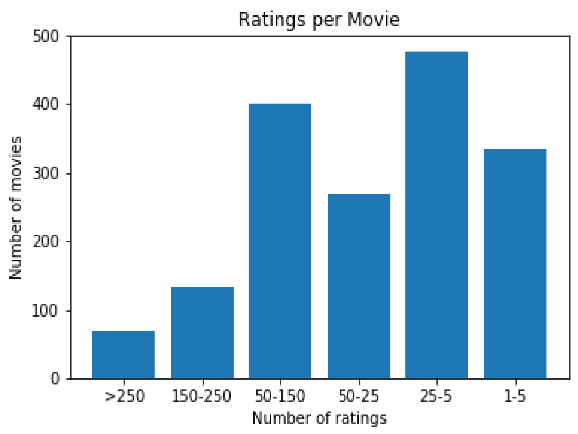

Figure 1: The number of ratings per movie in the MovieLens 100K dataset.

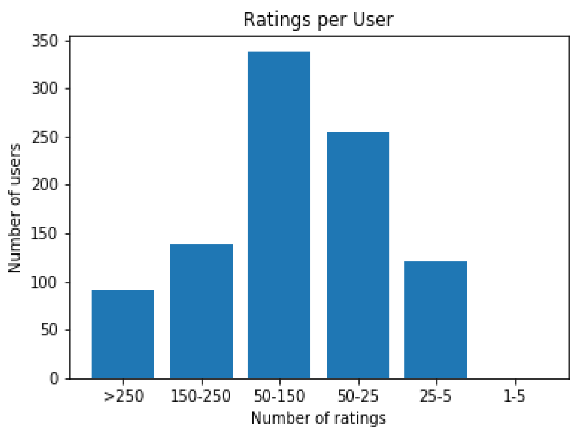

Figure 2: The number of ratings per user in the MovieLens 100K dataset.

size of 200. The hidden layers have a tanh activation function, and the weights are initialized to random values using the Xavier initialization technique. The Adam optimization method is used for the gradient descent step. The input vector $\mathbf{x}$ contains the age, gender and zip code for each user and the genre of each movie.

2.5.1 Hybrid Algorithm

Hybrid algorithms employ an ensemble of methods to improve both the accuracy and the diversity of recommendations. For a hybrid method, I utilized the observation that different techniques have varying accuracy rates for each rating. I combine the results of the user-item and item-item collaborative filtering methods with the matrix factorization model, using weights for the relative accuracy of each method for each rating. Essentially, the methods that are more accurate for a certain rating are given more weight when the prediction is closer to that rating.

3 Results

The results were run on the MovieLens dataset containing 100,000 samples. The dataset had 943 unique users and 1,682 movies. Figures 1 and 2 display how many ratings each user has and how many ratings each movie has respectively. The distribution of the ratings is shown in Figure 3. Most of the movies have between 5 and 25 ratings, while most users have between 50 and 150 ratings. The utility matrix for this dataset has 943 users (rows) and 1682 movies (columns) with a sparsity of 5.06%. This lack of data impedes the performance of the algorithms. None of the methods achieve over a 50% accuracy rate. However, some modest improvements are observed for the different algorithms and more information is gleaned by dissecting the accuracy ratings further.

3.1 Measuring Performance

There are many ways to measure the performance of recommendation systems. These metrics include measurements for rating accuracy, rating correlation, classification accuracy, diversity, coverage, and satisfaction. The Mean Squared Error (MSE) is used to measure the closeness of predicted ratings to the true ratings in the test results. Lower MSE scores correspond to higher prediction accuracy. Accuracy percentages are also recorded for each method, reflecting the number of ratings that are correct in the testing dataset. However, these scores alone don't provide enough data. Recommender systems are most effective when suggesting something the user would like and conversely avoiding recommenations of items that the user would not like. This means that ratings with lower and higher values are more important than the middle values. Therefore, for each model I broke down the accuracy for each rating number, which provides a better analysis of the performance of each model.

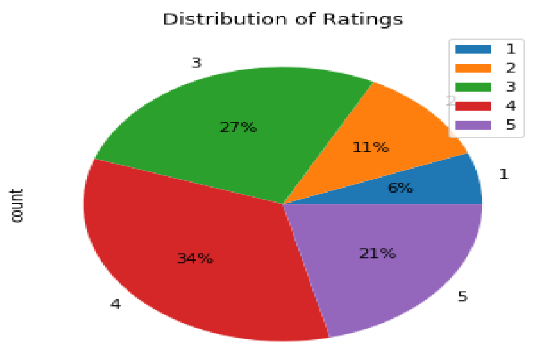

Figure 3: The distribution of ratings in the MovieLens 100K dataset. Ratings are between 1 and 5.

3.2 Analysis

The accuracy of each model is shown in Figure 4. The average estimate method, the baseline method, has a 36% accuracy rate. The best performance is observed by the matrix factorization and hybrid methods, with a nearly 42% accuracy rate. The item-based collaborative filtering methods perform better than the user-based methods, which suggest a stronger correlation among the items compared to the users in this dataset. The content-based technique performs worse than random guessing. The MSE rates for all methods follow a similar pattern and are displayed in Figure 5.

However, the overall accuracy rates do not tell the whole picture. As mentioned earlier, the ratings at the end of the spectrum are usually the most important to predict correctly. When the accuracy rates by rating are observed, a more interesting pattern emerges. See the table below. For instance, the baseline average estimate only performs well for the ratings 3 and 4. These are the most common ratings in the dataset, so this result isn't surprising. The results of the matrix factorization technique are the most evenly distributed, which suggests a better generalization of the data to the model. Also, for the ratings of 1 and 5 the matrix factorization method performs best. In the training dataset, ratings 1 and 2 comprised a total of 17% of the total ratings, so it isn't surprising that these two ratings had the lowest accuracy rates.

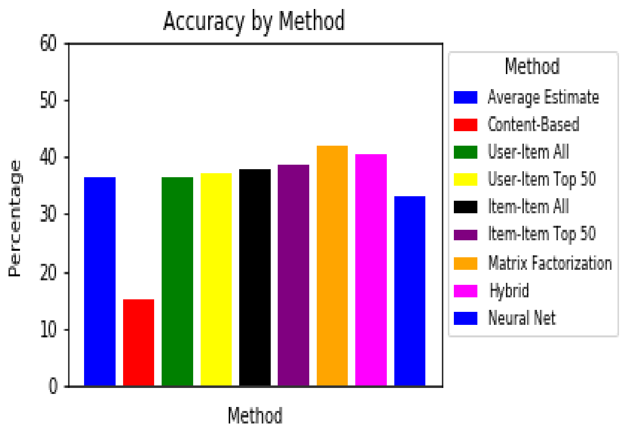

Figure 4: Accuracy rates per method using the full dataset.

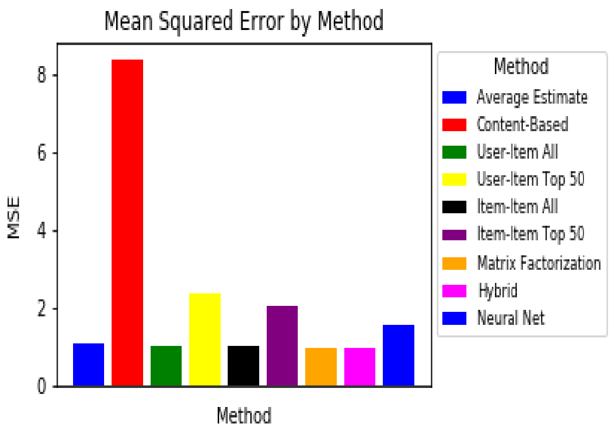

Figure 5: Mean squared error per method using the full dataset.

Accuracy percentage rates by rating for each method

Method	Average Estimation	Content-Based	User-Item All	User-Item Top 50	Item-Item All	Item-Item Top 50	Matrix Factorization	Hybrid	Neural Net
Rating 1	1.58	62.75	1.74	4.50	1.03	7.11	14.38	5.77	0.00
Rating 2	10.09	0.09	10.13	12.22	3.32	8.45	25.90	13.13	0.00
Rating 3	46.44	40.98	47.73	43.82	46.44	39.66	49.28	49.36	2.29
Rating 4	67.44	0.10	67.09	63.97	71.00	72.16	55.15	71.02	96.29
Rating 5	0.35	0.00	0.61	7.73	3.59	20.11	26.2	11.23	2.10

The sparsity of the data limits the performance of all the algorithms. How well would these methods perform if the data were less sparse? To answer this question, the data was filtered to produce a more dense utility matrix. The users with less than 100 ratings each and the movies with less than 100 ratings each were removed from both the training and testing sets. This produced a utility matrix with a sparsity of 29.86%. However, it reduced the overall size of the utility matrix to 334 users and 338 movies (from 938 users and 1682 movies). Even though the amount of data is reduced to less than half, the accuracy actually increased slightly for all methods, with the content-based algorithm displaying a marked increase in accuracy. See Figure 6. This shows that users and/or items with lots of ratings are much more predictive than users and/or items that are not associated with a lot of data points.

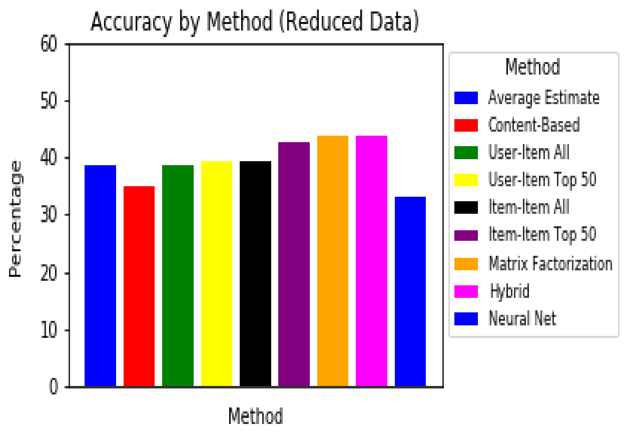

Figure 6: The accuracy percentages by method on the smaller dataset. The dataset was filtered to contain users with more than 100 ratings and movies with more than 100 ratings.

These users and items provide stronger correlations for prediction. The datapoints at the tails are much noisier and more difficult to predict. These results highlight the importance of these types of users and items in any recommender system. The challenge of recommender systems, though, is to find better models that utilize these smaller and less predictive datapoints.

4 Conclusion and Future Work

While the overall accuracy percentages display some slight improvements, the underlying percentages of each rating reveal a lot more about the algorithms. The ratings at the top and bottom end of the spectrum are the most important. The matrix factorization method performs best overall and provides the best generalization. Not surprisingly, the relative lack of 1 and 2 ratings in the data makes those ratings less predictive. However, matrix factorization's performance is the worst of all the methods on rating 4, which suggests combining it with other methods could probably yield stronger results. This is the idea behind the hybrid algorithm. While the hybrid algorithm didn't achieve the performance boost that I was expecting, there are many more options to explore that can potentially exploit the different predictive strengths of each of these models using different combinations.

The neural net's results are unexpected, with the distribution being concentrated at the 4 rating. The data shows the losses converging, and when a random set of ratings is run on the neural net a prediction rate of approximately 20% is observed. Therefore, I believe this is a local minimum to which the data converges. Still, this result needs further exploration, which could include other optimization methods and different types of layers and activation functions.

How to extract useful information and utilize a sparse dataset are questions that arose during my research in this project and are areas of future interest. This project highlights the importance of users and items with many datapoints versus the challenge of users and items found at the tails of the distribution. Successful recommender systems will exploit both types of data. Working with sparse datasets is a challenging feature of building effective recommender systems.

Please refer to the following github repository for the project notebook:
https://github.com/JanaanL/recommender_systems.git

References

[1] The netflix prize. `https://www.netflixprize.com/`. Accessed: 2019-04-25.

[2] L. Lü, M. Medo, C.H. Yeung, Y.C. Zhang, Z.K. Zhang, and Tao Zhou. Recommender systems. *Physics Reports*, 519:1–49, 2012.

[3] Frankowski D. Herlocker J. Schafer, J. and S. Sen. *Collaborative Filtering Recommender Systems*, pages 291–324. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[4] D. Bokde, S. Girase, and D. Mukhopadhyay. Matrix factorization model in collaborative filtering algorithms: A survey. *Procedia Computer Science*, 49:136–146, 2015.

[5] R. Salakhutdinov and A. Mnih. Probabalistic matrix factorization. *NIPS'07 Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 1257–1264, 2007.