Team IDK Final Report

Movie Recommender System

Members: Ming Feng Chua, Rena Pei Qi Chong, Zijun Li, Yingxuan Wu

1. Introduction

1.1. Background & Motivations

The popularity of online streaming services has surged since COVID-19. In 2021, the number of subscribers reached 1.3 billion, a 14% increase compared to the previous year, and 53% of adults reported using online streaming services in the United States. Online streaming generated revenue of 17.9 billion USD, which is the second largest in the movie industry [1]. Unlike traditional satellite TV, online streaming platforms can generate revenue through in-app purchases and advertisements. They collect data from multiple websites and mobile applications, as users use the same email, IP address, or device ID. Based on factors like review, cast, plot, crew, genre, popularity, etc., programmers build multidimensional models and recommend content to users [2]. However, such models can be challenging to perform on large data sets, so reducing variables is beneficial. In this paper, we aim to build a movie recommender system based on rating variables from MovieLens. We employed item-based collaborative filtering with cosine, correlation-based, and adjusted cosine similarities, as well as matrix factorization with the latent factor model to identify the most similar movies from previous ratings. Our experimental results shed light on the time-consuming and low accuracy of item-based collaborative filtering in sparse rating matrices and confirm the potential application of matrix factorization for movie recommendation. The latent factor model developed in this paper can be used to select movies with positively predicted ratings and makes recommendations when customers are using online streaming services. The study contributes a novel recommender system to the streaming platforms and saves users time in finding the desired content.

1.2. Dataset

Our dataset (ml-latest-small) used for this project is obtained from MovieLens (http://movielens.org). It contains a total of 100836 ratings for 9742 movies, rated 610 users between 29 March 1996 and 24 September 2018. Users selected for this dataset had rated at least 20 movies [3].

The files included in this dataset are *links.csv*, *movies.csv*, *ratings.csv*, *genome-scores.csv*, *genome-tags.csv* and *tags.csv*. In our project, we used *ratings.csv*, which contains all the ratings for movies by users. Each line of this file after the header row represents one rating for one movie by one user, and has the following format:

Column	**Data Description**
userId	User id of user who rated the movie. User id is unique for each user.
movieId	Movie id of rated movie. Movie id is unique for each movie.
rating	Rating given by user for the movie. Ratings are made on a scale of 0.5 stars to 5.0 stars, with increments of 0.5 stars.

1.2.1. Data Pre-processing

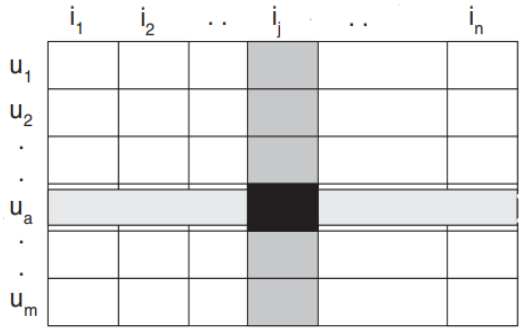

Figure 1: Ratings matrix of dimensions m × n for user-item data [4]

The ratings data was transformed into a ratings matrix R, where each $R_{i,j}$ value represents the rating given by the *i-th* user on the *j-th* item. This gives a m × n ratings matrix, where there are m unique users and n unique items. By transforming our original data into the ratings matrix, it allows us to easily obtain all ratings on a particular item or by a particular user. It is worthy to note that the ratings matrix generated from our dataset had a significantly high sparsity level.

2. Models

2.1. Item-Based Collaborative Filtering

Item-based collaborative filtering recommends similar items the user may prefer based on the preferences of similar users. The similarity of items is essential for this method, which is calculated based on the user ratings for the items.

For this method, we adapted the algorithm proposed in the paper by Sarwar et al., "Item-Based Collaborative Filtering Recommendation Algorithms". Firstly, similarity of item-pair (i, j) is calculated based on users who have rated both items. Following which, k most similar items to target item i were selected, which were used to compute the predicted ratings by taking the weighted average of the target user's ratings on these k most similar items [4].

2.1.1. Methodology

2.1.1.1. Similarity Computation

To compute the similarity between pair of items, users who have rated both items i and j are isolated. Similarity computation technique is applied on ratings of users who have rated both items to obtain the similarity of item-pair (i, j). Three similarity computation techniques were used to find the best technique for similarity computation. These techniques are cosine similarity, correlation similarity, and adjusted-cosine similarity.

Cosine similarity is calculated based on the cosine angle between rating vectors from two items. The formula for cosine similarity, corresponding to the m × n ratings matrix is given by:

$$\text{cos. sim(i, j)} = \cos\left(\vec{i}, \vec{j}\right) = \frac{\vec{i} \cdot \vec{j}}{\|i\|_2 \cdot \|j\|_2}$$

$$\text{where } \vec{i} \text{ and } \vec{j} \text{ is the rating vector of item } i \text{ and } j \text{ respectively.}$$

Correlation similarity is calculated using the Pearson-r correlation between two items. The formula for correlation similarity, where U is the set of users who have rated both items i and j, is given by:

$$\text{corr. sim(i, j)} = \frac{\sum_{u \epsilon U}\left(R_{u,i} - \overline{R}_i\right)\left(R_{u,j} - \overline{R}_j\right)}{\sqrt{\sum_{u \epsilon U}\left(R_{u,i} - \overline{R}_i\right)^2}\sqrt{\sum_{u \epsilon U}\left(R_{u,j} - \overline{R}_j\right)^2}}$$

$$\text{where } R_{u,i} \text{ is the rating by user } u \text{ on item } i,\ \overline{R}_i \text{ is the average rating of item } i.$$

Adjusted-cosine similarity is measured by subtracting the average of a user's ratings from each rating, $R_{u,i}$ and $R_{u,j}$ and then applying the cosine similarity formula. It accounts for differences in users' rating scale [4]. The formula for adjusted-cosine similarity, where U is the set of users who have rated both items i and j, is given by:

$$\text{adj cos. sim(i, j)} = \frac{\sum_{u \epsilon U}\left(R_{u,i} - \overline{R}_u\right)\left(R_{u,j} - \overline{R}_u\right)}{\sqrt{\sum_{u \epsilon U}\left(R_{u,i} - \overline{R}_u\right)^2}\sqrt{\sum_{u \epsilon U}\left(R_{u,j} - \overline{R}_u\right)^2}}$$

$$\text{where } R_{u,i} \text{ is the rating by user } u \text{ on item } i,\ \overline{R}_u \text{ is the average rating by user } u.$$

2.1.1.2. Prediction of Ratings

Once the similarity of each item-pair is computed, we predict rating on item i for a user u. For each item, we select the k most similar items to item i. These k most similar items will be used to compute the predicted rating. Weighted sum formula is used to compute the prediction on an item i for a user u, using only the ratings given by the user u on the k most similar items to item i. Each rating is weighted by its corresponding similarity value, $s_{i,j}$, between item i and j [4]. Mathematically, the weighted sum formula is given by:

$$P_{u,i} = \frac{\sum_{\text{all similar items, N}}(s_{i,N} \times R_{u,N})}{\sum_{\text{all similar items, N}}(|s_{i,N}|)}$$

The weighted sum formula follows the same range of the original ratings data, since the weighted formula is scaled by the sum of similarity terms [4].

2.1.2. Experimental Procedure

The ratings data is split into training and test sets, where 70% was used as the training set and the remaining 30% was used as the test set. The training set was transformed into the ratings matrix R, represented in the form of a sparse matrix using `scipy.sparse::csr_matrix`. From there, similarity computation and prediction was performed to obtain the predicted ratings on the test set. The mean squared error (MSE) was chosen as the evaluation metric to compare between different similarity computation techniques and evaluate the effect of k on the predicted rating.

2.1.3. Results

2.1.3.1. Comparison of Similarity Computation Techniques

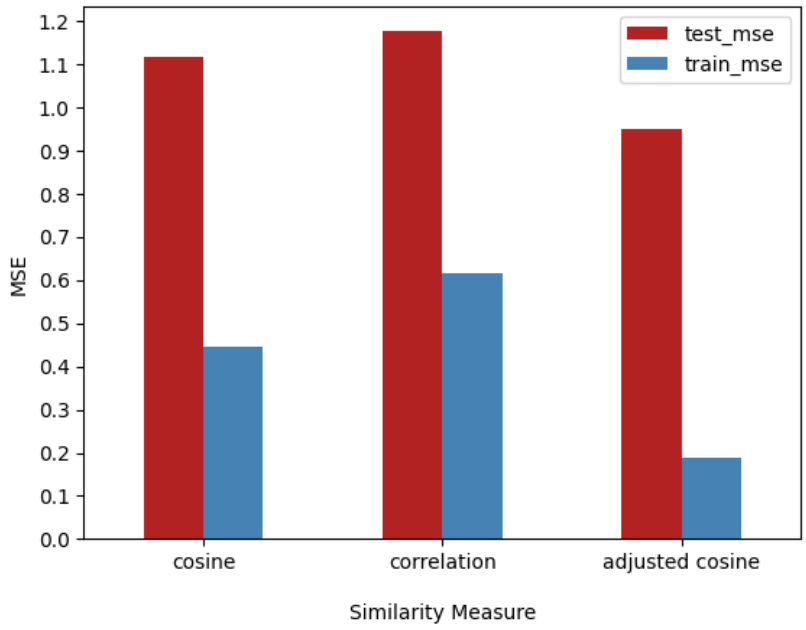

Figure 2: Test & training MSE for different similarity measures (k = 70)

For each similarity measure, we implemented the algorithm to compute the item-pair similarity values and used the weighted sum formula to predict the ratings on the test set. The test and training MSE were computed to compare the results from the different similarity measures. From Figure 2, it can be observed that the adjusted-cosine similarity computation performed the best as its MSE was the lowest among the three techniques.

2.1.3.2. Effect of Number of Most Similar Items, k, on Prediction

The number of most similar items, k, used in ratings prediction affects the accuracy of the prediction significantly. To observe how different values of k can affect the predicted ratings, the value of k was varied during the ratings prediction and the test MSE was calculated.

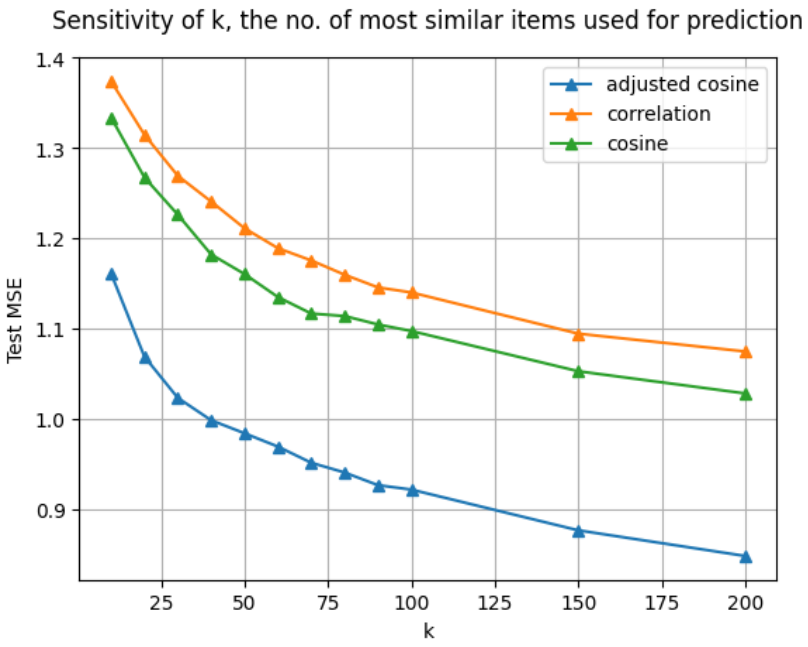

Figure 3: Effect of k on ratings prediction, measured by test MSE

From Figure 3, it can be observed that using a greater number of most similar items in the prediction of ratings improves the accuracy of the prediction. However, the rate of improvement in accuracy diminishes as the the number of most similar items, k, increases.

2.1.4. Limitations of Method

Similarity computation of item-pairs was significantly slow, taking approximately four hours for almost 10,000 movies. As such, the algorithm was sped up pre-computing the similarity values and storing them into a csv file. While the similarity values used for the recommender system may not be accurate real-time, it allowed us to generate predictions fairly quickly with little trade-off as the similarity values are fairly static, according to Sarwar et al. [4].

2.2. Matrix Factorization

Matrix factorization is a model-based collaborative approach using latent factor models in movie recommender systems. In a latent factor model, the original user-item rating matrix can be represented as a combination of latent factors, these hidden latent factors can capture and explain underlying preferences, tastes, characteristics of user and items, which are not directly represented in the observed data. These latent factors can represent certain attributes of movies or users that affect the ratings, this may include genre, director, movie cast, etc.

Matrix factorization involves decomposing the original user-item rating matrix into 2 smaller matrices, a matrix representing user preferences in terms of latent factors and a matrix representing the latent factors associated with each movie. These matrices can be obtained through an optimization process, to minimize the error between the predicted and observed user-item rating matrix through estimation techniques such as stochastic gradient descent (SGD) or alternating least squares (ALS).

Once the matrices are obtained, they can be used to generate predictions and provide personalized recommendations for each user. The movie recommender system predicts the user's preferences for movies that they have not seen yet based on the user's preferences for the latent factors associated with the movies.

2.2.1. Methodology

The original $n \times m$ user-item rating matrix can be approximated using 2 low rank user and item matrices P and Q respectively in the joint latent factor space. The latent factor space would be of dimension k and the estimated ratings can be obtained through the inner product of these 2 matrices. Matrix P, of dimension $k \times n$, represents the users in the joint latent factor space and matrix Q, of dimension $k \times m$. The predicted rating for user u for item i is $\hat{r}_{ui} = p_u q_i$, where $\hat{r}$ is $n \times m$.

$$\begin{bmatrix} \hat{r}_{11} & \cdots & \hat{r}_{1m} \\ \vdots & \ddots & \vdots \\ \hat{r}_{n1} & \cdots & \hat{r}_{nm} \end{bmatrix} = \begin{bmatrix} p_1^T \\ \vdots \\ p_n^T \end{bmatrix} \cdot \begin{bmatrix} q_1 & \cdots & q_m \end{bmatrix}$$

k, in this context, is the number of latent factors, where the greater k is, the greater the amount of information captured by the approximation. To estimate both matrices P and Q, the evaluation metric mean-squared error (MSE) of the observed ratings will be minimized:

$$\min \sum_{u,\, i \in \Omega} (r_{ui} - \hat{r}_{ui})^2,$$

$$\text{where } \Omega \text{ is the set of user-item pairs for observed ratings.}$$

To prevent overfitting and improve the performance of the model on unseen test data, we included a penalty term to the above minimization problem. By balancing the bias-variance

tradeoff between fitting the training data well and performance on unseen test data, regularization can help improve the predictive capabilities of the latent factor model. The objective function then becomes:

$$\min \sum_{u,\, i \in \Omega} (r_{ui} - \hat{r}_{ui})^2 + \lambda\Big(\|p_u\|^2 + \|q_i\|^2\Big),$$
$$\text{where } \|\cdot\| \text{ is the Frobenius norm and } \lambda \text{ is the regularization parameter.}$$

To estimate both matrices P and Q, the optimization technique SGD was used to minimize the above objective function with penalty by iteratively adjusting the values of the model parameters to reduce the MSE. Through multiple epochs, the optimal P and Q can be obtained for prediction. We define:

$$e_{ui} = r_{ui} - \hat{r_{ui}},$$

and the vectors p_u and q_i can be updated as such:

$$p_u \leftarrow p_u + \gamma(e_{ui} q_i - \lambda p_u),$$
$$q_i \leftarrow q_i + \gamma(e_{ui} p_u - \lambda q_i).$$

2.2.2. Experimental Procedure

The ratings data is split into training and test sets, where 80% was used as the training set and the remaining 20% was used as the test set. The training and test set were transformed into the ratings matrix R, represented in the form of sparse matrices using the method `scipy.sparse::csr_matrix`.

Matrices P and Q were initialized with the method `np.random.rand` and populated with values in the range $(0,\ 1]$ and $k = 40$.

From there, SGD was performed on the 2 estimated P and Q matrices, with epoch set to 500, learning rate $\gamma = 0.0015$ and regularization parameter $\lambda = 0.06$. Test MSE was chosen as the evaluation metric to compare the errors between different epochs to determine the optimal P and Q, with early stopping once there are no observed improvements in the test MSE.

2.2.3. Results

The matrix factorization model described in section 2.2.1 was implemented and the test MSE was measured across different epochs. The results are presented below in the line chart.

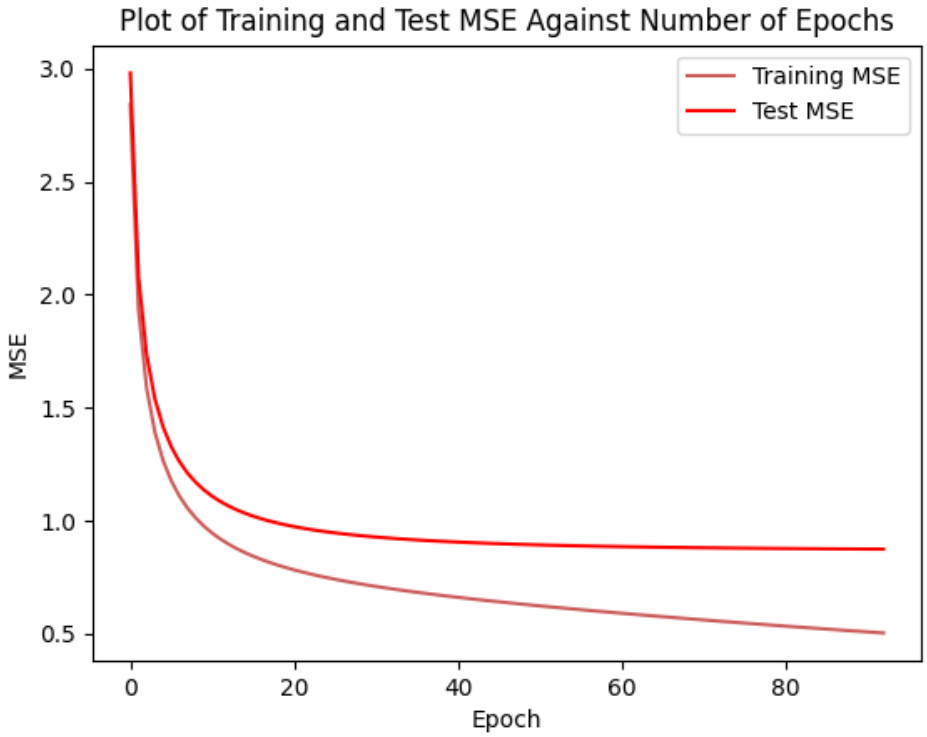

Figure 4: Plot of Training and Test MSE against number of Epochs

As seen in Figure 4, as the number of epochs increases, both training and test MSE decrease rapidly initially, where the rate of decrease in the training MSE is significantly greater. We observe that from 60 epochs onwards, the rate of decrease for test MSE relatively stagnated. Our algorithm yielded the optimal P and Q after epoch 98 as there were no significant improvements in the test MSE. The final training MSE was 0.503 and test MSE was 0.874.

2.2.4. Future Improvements

This latent factor model can be further improved in the future by using cross validation to find the optimal hyperparameters. This involves optimizing the the learning rate γ, regularization strength λ, and the number of latent factors k. Beyond that, one could utilize methods and techniques such as Bayesian optimization and grid search to find the best hyperparameters for their matrix factorization model.

One can also utilize a hybrid model, incorporating matrix factorization and non-linear methods such as neural networks. Neural matrix factorization models (NMF) have several advantages over the traditional matrix factorization models. For instance, it can capture more complex and non-linear relationships between users and items, leading to more accurate predictions and

recommendations. NMF can be more scalable than matrix factorization, especially when dealing with large datasets. This is because NMF can be parallelized across multiple GPUs or machines, allowing for faster processing of large amounts of data. NMF can also potentially outperform traditional matrix factorization in terms of accuracy of recommendations and predictions. This is particularly true in applications where non-linear relationships or diverse data types are important factors.

3. Discussion

From the experimental evaluation of item-item collaborative filtering schemes, we made some important observations. Firstly, the adjusted cosine similarity provides better quality of prediction than the predictions generated by cosine similarity and correlation similarity, with lower train_mse and lower test_mse. However, the improvement is not significantly large. Secondly, we observed that the number of most similar items, k, used in ratings prediction affects the accuracy of the prediction significantly. Generally, the rate of improvement in accuracy diminishes as the the number of most similar items, k, increases.

From the implementation of matrix factorization, we discovered that as the number of epochs increases, both training and test MSE decrease rapidly until the number of epochs reaches 60. After 60 epochs, the rate of decrease gets slower.

In comparison, the matrix factorization method is less time-consuming and more efficient than the item-based collaborative filtering algorithm in practice, while having lower test MSE and being more suitable for analysis in situations with a large amount of sparse data such as our movie recommendation system.

4. Conclusion

In conclusion, movie recommendation systems are widely used to suggest movies to users based on their preferences nowadays. Through our analysis, we have found that matrix factorization algorithms outperformed item-based collaborative filtering, while being a powerful technique that can somehow accurately predict user ratings and suggest new movies based on their interests. As movie recommendation systems have become an essential component of modern streaming services and e-commerce platforms and movie libraries continue to grow, the need for accurate and efficient recommendation systems will only increase. Therefore, it is important to continue to research and develop new, hybrid techniques for improving movie recommendation systems.

References

[1] “2021 Theme Report.” Motion Picture Association, March 14, 2022. https://www.motionpictures.org/research-docs/2021-theme-report/

[2] Rajarajeswari, S., Sharat Naik, Shagun Srikant, M. K. Sai Prakash, and Prarthana Uday. “Movie Recommendation System.” Emerging Research in Computing, Information, Communication and Applications, 2019, 329–40. https://doi.org/10.1007/978-981-13-5953-8_28

[3] Movielens Latest Datasets. GroupLens. (2021, March 2). https://grouplens.org/datasets/movielens/latest/

[4] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based Collaborative Filtering Recommendation Algorithms. Proceedings of the 10th International Conference on World Wide Web. https://doi.org/10.1145/371920.372071