Construction of a Movie Recommendation System Using the Movielens Database

Tiglath Moradkhan

2023-05-29

Executive Summary

Recommendation systems function by evaluating data collected about the likely behaviors and patterns of customers and then provide suggestions based on the evaluations. These suggestions are helpful in allowing users to easily search for products or services based on the least available information. Recommender systems have varied applications, of which one of them is suggesting movies to watch based on data collected from many customers.

In this project, we used the 10M Movilens dataset developed and hosted by the GroupLens research group to build a movie recommendation system. The 10M Movielens datset was split into an **edx** and **final_holdout_test** sets. Two different machine learning algorithms were trained on further splits of the **edx** dataset. These were linear regression and matrix factorization. The linear regression algorithm fitted a linear equation which included bias effects such as movie and user effects together with regularization penalty terms. In contrast, for matrix factorization we used the LIBMF method in the **recosystem** package. Based on reported RMSE values, we found that matrix factorization was a better algorithm to use to build a movie recommendation system. We then applied matrix factorization on the **final_holdout_test** dataset and obtained an RMSE of approximately 0.78, which is below the required threshold of 0.86.

1. Introduction

The movielens dataset is a dataset containing approximately 11 million user ratings of about 27,000 movies, developed by the GroupLens research group (see Movilens Dataset).

For this project, the 10M version of the dataset was used to build a recommedation system. The 10M version contains 10 million ratings of 10,000 movies (see 10M Movielens Dataset).For the purposes of this excercise, the dataset was split into an **edx** and a **final_holdout_test** dataset. The **edx** dataset was used for preliminary data visualizations and for building the recommendation system, while the **final_holdout_test** dataset was used to evaluate how close the predictions from the recommendation system were to the true values.

2. Methods

The 10M Movilens dataset was downloaded and split into two sets; an **edx** dataset which contains 10% of the original 10M movilens data and **final_holdout_test** which was used as a validation dataset to test the predictions made using the **edx** dataset.

2.1 Data Visualization

Preliminary data visualizations were performed on the **edx** dataset.

2.2 Building a recommendation system

A movie recommendation was built using two algorithms; linear regression together with regularization and matrix factorization. These algorithms are defined and further discussed below. All discussions of linear regression applied to movie recommendation comes from the Harvard edx course on machine learning.

2.2.1 Linear Regression

In general, linear regression assumes that the data can be separated into a linear combination of weights, w given to each feature x as shown by equation 1 below. In addition, linear regression also includes a bias term which is equal to b (Jovel and Greiner 2021).

$$Y = w_1 \times x_1 + w_2 \times x_2 + ... + w_n \times x_n + b \tag{1}$$

In the case of movie recommendations, the linear model is defined by the following equation

$$Y_{u,i} = \mu + \epsilon_{u,i} + b_i \tag{2}$$

In this model, $Y_{u,i}$ represents the rating for movie i by user u, μ is the true rating for all users and movies, b_i is the average ranking of movie i and occurs as a result of the movie effect where some movies are generally rating higher than others, and $\epsilon_{u,i}$ represent independent errors for movie i by user u.

In the case of modeling movie predictions, the simplest assumption is that the rating is the same for all movies and users and any differences are due to random variation. With this assumption, equation 2 simplifies to:

$$Y_{u,i} = \mu + \epsilon_{u,i} \tag{3}$$

However, movie ratings may be influenced by two additional effects defined as the movie and the user effect. The movie effect as explained above, arises as a result of some movies receiving higher rates than others. The inclusion of the movie effect in the linear model is depicted in equation 2 above. In contrast, the user effect occurs as a result of different rating patterns among different users. For example, some users may tend to consistently rate movies they like as 4 and 5; while other users may dislike most movies and give rates of 1 or 2. In the linear model, the user effect is defined by the variable b_u. With the user effect, equation 2 becomes as shown below.

$$Y_{u,i} = \mu + \epsilon_{u,i} + b_i + b_u \tag{4}$$

The movie and user effects are themselves calculated as depicted below.

$$\hat{b_i} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{\mu}) \tag{5}$$

$$\hat{b_u} = \frac{1}{N}\sum_{i=1}^{N}(y_{u,1} - \hat{b_i} - \hat{\mu}) \tag{6}$$

2.2.2 Regularization

Although the linear model is a good algorithm for movie predictions, it does not take into account the fact that there may be movies which are rated by few users. Regularization can help to penalize large estimates that result from small samples sizes, thus constraining the total variability of effect sizes such as those coming from either the movie or user effect defined in Section 2.2.1. In regularization the equation shown below is minimized.

$$\frac{1}{N}\sum_{u,i}(y_{u,i}-\mu-b_i)^2+\lambda\sum_i b_i^2 \tag{7}$$

In equation 7, the first part is the mean squared and the second is the penalty term which increases as the magnitude of the b values increases. The value of b which minimizes equation 7 is calculated as follows.

$$\hat{b_i}(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}(Y_{u,i}-\hat{\mu}) \tag{8}$$

where, n_i is the number of ratings b for movie i

2.2.3 Matrix Factorization

In matrix factorization, a matrix is constructed to contain the ratings that users give to a product, which in this case are movies. The rows are columns of this matrix represent users and items, respectively while the entries in the matrix contain user ratings. Since users tend to consume a portion of movies, most of the entries in the matrix will be missing. The purpose of matrix factorization is build a model that will be able to predict movies for users based on user-item observations.

The model assumes that the ratings matrix can be constructed as the product of two lower dimension matrices, known as latent factors (The R Project for Statistical Computing 2023). This is represented as follows.

$$R=P^TQ \tag{9}$$

where R is the ratings matrix with size $n\times m$ and P and Q are the latent factors. P, which has size $k\times n$, represents users while items are represented by Q which has size $k\times n$. The value k is equivalent to the number of latent features.

As discussed above for the linear model, user ratings are typically biased and are affected by small sample sizes; therefore to improve the model, bias terms together with a penalty regularization term are introduced (Strömqvist 2018). The model, therefore becomes

$$R=P^TQ+b_i+b_u \tag{10}$$

and the minimization equation to solve is given by

$$\sum_{u,i\in K}(r_{u,i}-p_uq_i^T)^2+\lambda(||q_i||^2+||p_u||^2) \tag{11}$$

In equation 10, q_i is the latent vector for item i, p_u is the latent vector for user u, λ is the regularization penalty term, and $||.||^2$ is the matrix norm.

Matrix factorization utilizes two different algorithms; memory based collaborative filtering (CF) and model-based CF (Hahsler 2022). Memory-based CF algorithms apply either the whole or a large sample of the

user database to make recommendations, while model-based CF algorithms utilize the user database to first learn a more compact model such as clusters of users which have similar preference. This compact model is then used to make recommendations.

3. Results

3.1 Exploratory Data Analysis

We first looked at the distribution of the number of ratings among users.The results of this analysis are shown in Table 1 and Figure 1 below which depict the mean distribution of ratings. According to Table 1 and Figure 1, while there are some users who give ratings of below 3, the mean for those ratings is smaller than the mean of ratings higher than 3.

Table 1: The mean distribution of the number of ratings

rating	avgnumber
0.5	85374
1.0	345679
1.5	106426
2.0	711422
2.5	333010
3.0	2121240
3.5	791624
4.0	2588430
4.5	526736
5.0	1390114

We also looked at the frequency of ratings. This analysis is shown in Table 2 and Figure 2 below. Similar to the data presented in Table 1 and Figure 1, users have a preference for ratings of 3 and 4. Additionally, according to Table 2 and Figure 2, half number ratings have smaller frequencies compared to whole number ratings, possibly indicating a higher preference for whole number ratings.

Table 2: The frequency distribution of the number of ratings

Rating	Frequency
0.5	0.0094859
1.0	0.0384085
1.5	0.0118250
2.0	0.0790464
2.5	0.0370009
3.0	0.2356919
3.5	0.0879577
4.0	0.2876016
4.5	0.0585259
5.0	0.1544562

As indicated above, there seems to be a higher preference for whole number ratings. This observation is further supported by the data in Table 3 which is a summary of the top 10 movie ratings.

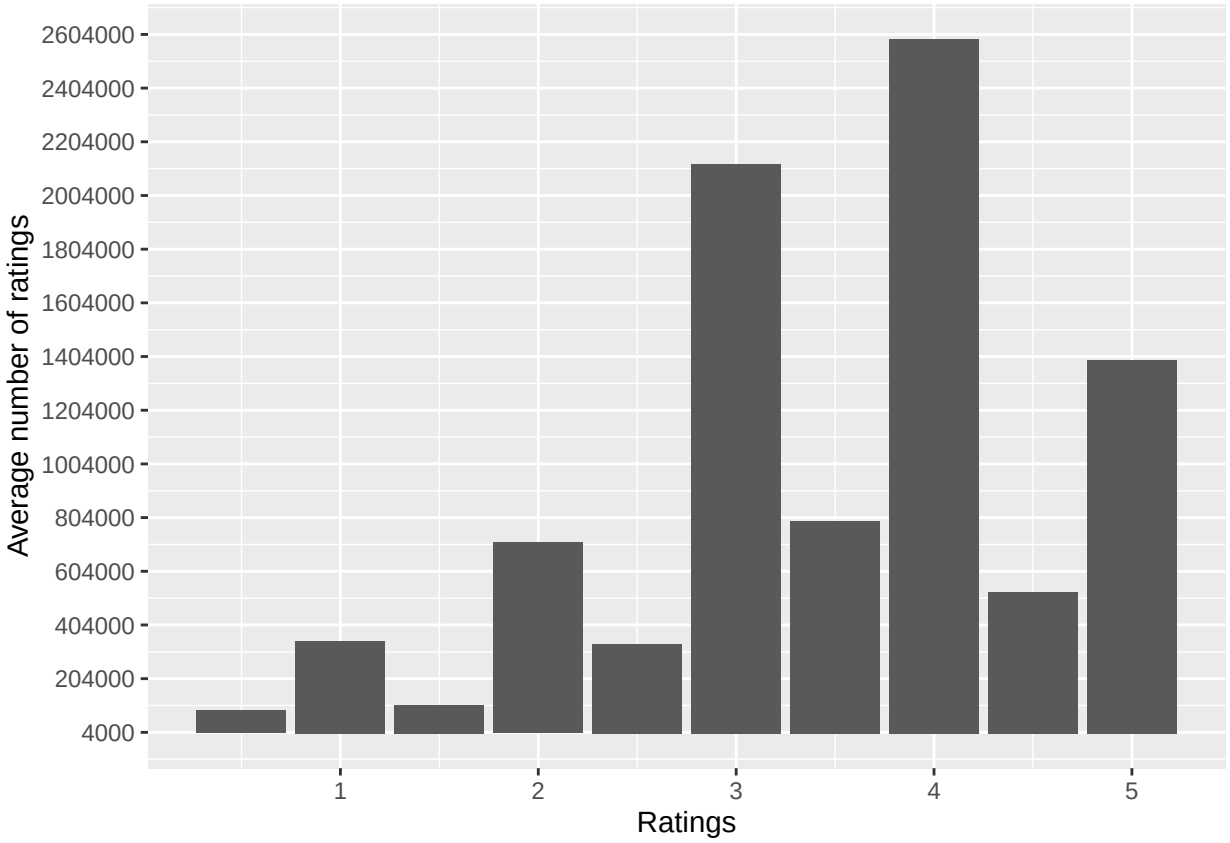

Figure 1: Mean distribution of the number of ratings

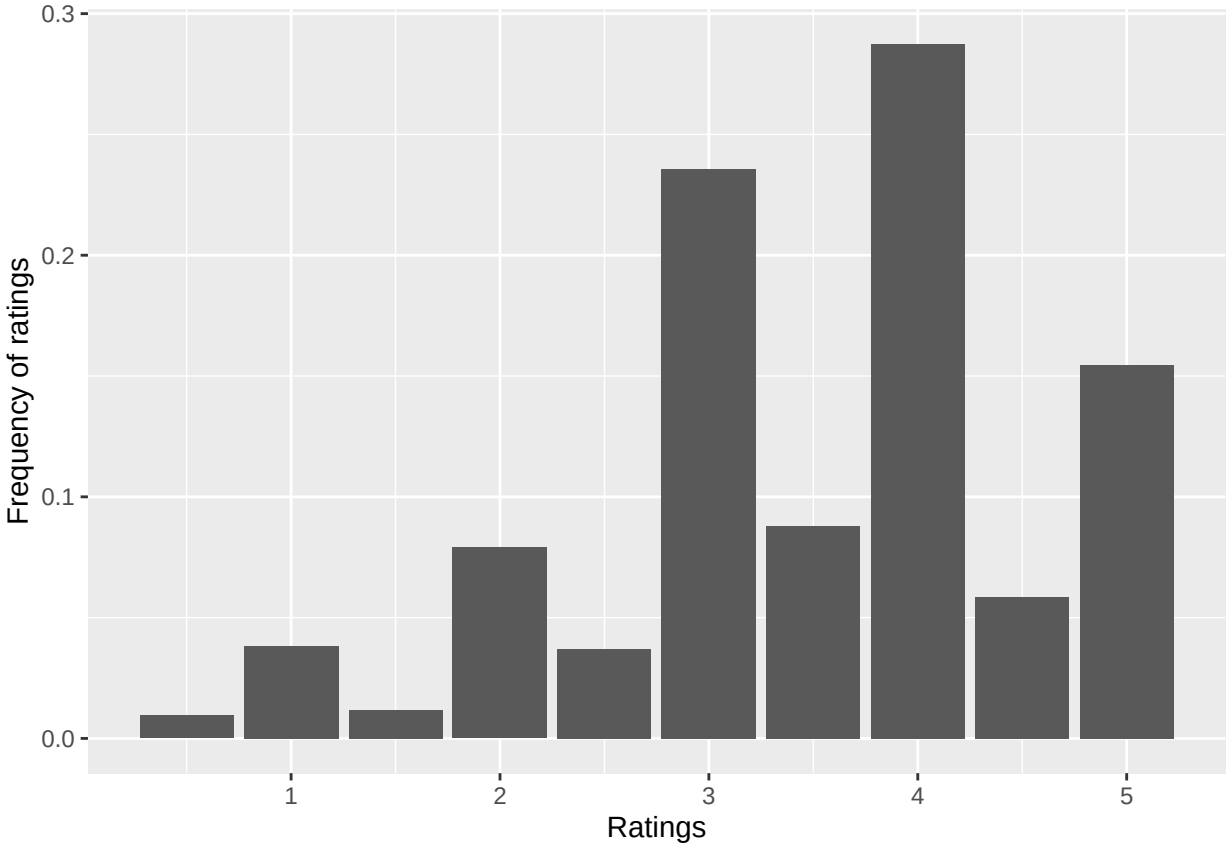

Figure 2: Frequency of the number of ratings

Table 3: A summary of the top 5 movie ratings

Rating	Number
4.0	2588430
3.0	2121240
5.0	1390114
3.5	791624
2.0	711422

3.2 Model Training

To develop our movie recommendation system, we split the edx dataset into a training set and a test set. The training set contained 10% of the edx dataset and the test set contained the remaining 90%.

We then trained the training set using the linear model and matrix factorization. To assess the strength of each model, we calculated and reported the RMSE of each model on the testing dataset. The goal was to obtain an RMSE value that is smaller than 0.86.

3.2.1 Linear Modeling

To train the linear model on the training set we tested each of the terms in equation 4 in section 2.21 and compared their effect on the RMSE. The RMSE is given by

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2} \tag{12}$$

We also experimented with the addition of the regularization penalty term to equation 4.The approach we used for for training the linear model is shown in Table 4 below.

Table 4: Linear Model Training Approach

Model	Method
Model 1	Average Only Model
Model 2	Movie Effect Model
Model 3	User Effect Model
Model 4	Regularization Model

3.2.1.1 Linear Modeling Without Regularization

First we trained the linear model without including the regularization penalty term. The modeling results are depicted in Table 5 below which compares the RMSE values. As we can see from this table, including bias effects substantially improved the RMSE value. We saw a decrease of 0.12 in RMSE between models 1 and 2, indicating that only using the average is a poor training model and that a better training model should include bias effects. Further, there was a decrease of approximately 0.2 in RMSE between models 2 and 3, suggesting that when the user effect is included in the training model, there is an additional improvement in our ability to better make movie recommendations.

Table 5: Linear Model RMSE Comparisons

Model	Method	RMSE
Model 1	Average Only Model	1.0600537
Model 2	Movie Effect Model	0.9429615
Model 3	User Effect Model	0.8646843

3.2.1.2 Linear Modeling With Regularization

Next we trained the linear model by including the regularization penalty term. To determine what regularization penalty term value would improve the RMSE, we first performed cross-validation on the training set using tuning values ranging from 0 to 25 with incremental steps of 0.25, and then used the optimal regularization penalty term from cross-validation to regularize the movie and user effects. Table 6 shown below depicts the RMSE values when regularization is introduced in the model. According to these results, regularizing the movie and user effects did not improve our ability to better predict movies. Because of this poor performance we next sought to compare the linear model with matrix factorization. The results of this analysis are shown in the next section.

Table 6: Linear Model RMSE Comparisons with Regularization Added

Model	Method	RMSE
Model 1	Average Only Model	1.0600537
Model 2	Movie Effect Model	0.9429615
Model 3	User Effect Model	0.8646843
Model 4	Regularization Model	0.8821637

3.2.2 Movie Predictions with Matrix Factorization

As was discussed in section 2, matrix factorization seeks to construct a matrix which contains user-item observations where the columns and rows of the matrix represent items and users and the entries contain the ratings users give to the items. This matrix is constructed as a product of two lower dimension latent factor matrices. First we experimented with algorithms found in the **recommmenderlab** package but found that executing them takes a very long time. Therefore, we next looked at using the **recosystem** package. The **recosystem** package is a wrapper of the LIBFM library which is used for recommendation systems using parallel matrix factorization (see Recosystem documentation). Additionally, the **recosystem** package contains many of the features of the LIBFM library, in addition to several user-friendly R functions which help to simplify data processing and model assembly. The main steps of the **recosytem** package are defined as follows:

1. construction of a model object;
2. selection of ideal tuning parameters;
3. training the model by using the ideal tuning parameters selected in step 2 and;
4. using the trained model to make predictions

According to the documentation, the **recosystem** package provides the following parameters to select for tuning:

1. *dim*: the number of latent factors;

2. *costpl1*: the L1 regularization cost applied to user factors;
3. *costpl2*: the L2 regularization cost applied to user factors;
4. *costql1*: the L1 regularization cost applied to item factors;
5. *costql2*: the L2 regularization cost applied to item factors; and
6. *lrate*: the learning rate, which is the step size in gradient descent[1]

Table 7, below, compares reported RMSE values for the linear model and matrix factorization. According to this data, matrix factorization substantially improved the RMSE value. With matrix factorization, the RMSE decreased from approximately 0.88 to 0.79. This indicates that matrix factorization improves our ability to make movie predictions.

Table 7: Linear Model and Matrix Factorization RMSE Comparisons

Model	Method	RMSE
Model 1	Average Only Model	1.0600537
Model 2	Movie Effect Model	0.9429615
Model 3	User Effect Model	0.8646843
Model 4	Regularization Model	0.8821637
Matrix Factorization	Recosystem	0.7855157

3.3 Final Model Validation

The purpose of this project was to first find an optimal machine learning algorithm by training it on the **edx** dataset, and then applying it to **final_holdout_test**. Based on the modeling results from the linear and matrix factorization algorithms above, we applied matrix factorization on **final_holdout_test**. The reported MSE value for making movie predictions using matrix factorization on the **final_holdout_test** was 0.78. This is smaller than the required threshold RMSE value of 0.86.

4. Conclusion

The goal of this project was to build a movie recommendation system using the 10M Movielens dataset. This dataset was split into an **edx** set and a **final_holdout_test** set. Two different machine learning algorithms were trained on the **edx**. The first involved fitting a linear equation that included movie and user biases together with regularization, and the second involved using the **recosystem** package which applies the LIBMF matrix factorization algorithm. For each algorithm that was trained and then used to make movie predictions, the resulting RMSE was calculated. Based on the results we found that matrix factorization had a substantial improvement on the RMSE value. For the final validation we applied matrix factorization using **edx** as the training set and **final_holdout_test** as the test set and calculated an RMSE value of approximately 0.78, which is smaller than the required threshold of 0.86.

When we trained the linear model, we saw that movie and user effect biases are important to include as they improve the movie recommender. While we only included movie and user effect biases, we did not consider the possibility that there may also be genre effect biases which could also have an impact on movie recommendation systems. Future work should expand the linear model to include these additional biases. Secondly, when we sought to use matrix factorization as our second machine learning algorithm, we first experimented with algorithms in the **recommenderlab** package; however, a major limitation that we encountered is that some of these take hours to execute with a small laptop. Therefore, we had to use the **recosystem** package which was easier to implement. It may be that there are additional matrix factorization

[1]The algoirthm used here applies gradient descent to find optimal values for the user and item matrices in order to minimize the error between actual and predicted ratings.

machine learning algorithms that may be better than the LIBMF method provided in the **recosystem** package. Therefore, with a more powerful powerful computer, other matrix factorization algorithms should be explored to determine which of these could be a better movie recommender. Another related issue with computer power was our inability to tune all of the possible tuning parameters provided in the **recosystem** package, which could possibly indicate that our reported RMSE on the validation dataset is not optimal. Thus, with a more powerful computer, all of these possible tuning parameters should be tuned. Finally, it should be noted that the 10M Movilens dataset excludes other important variables such as user behavior which influence movie ratings, and thus impact how a movie recommender is trained. To improve the movie recommendation system, a dataset similar to that used by for example Netflix should be applied.

5. References

Hahsler M (2022) Recommenderlab: An r framework for developing and testing recommendation algorithms

Jovel J, Greiner R (2021) An introduction to machine learning approaches for biomedical research. Frontiers in Medicine 8:771607. https://doi.org/10.3389/fmed.2021.771607

Strömqvist Z (2018) Matrix factorization in recommender systems. How sensitive are matrix factorization models to sparsity? PhD thesis

The R Project for Statistical Computing (2023) Matrix factorization with side info