Movie Recommender System

Principal Investigator: Meng Yuan[myuan43@wisc.edu]

Date:2022/05/04

Abstract

The system suggests a list of movies that the user might enjoy based on the watching history, rates, and tags of movies. It provides recommendations of the movies usually specific among users. The system will be able to better understand users and recommend movies that are more likely to receive higher ratings when it gets more and more information, as known as, though machine learning.

1. Introduction

An increasing number of departments in our life flood into the stream of data. It soon becomes an indivisible part of our life. With the rapid popularity of the internet, the recommendation system become commonplace. It changed the way how we treat the communication between the internet and users. For any given product, there are sometimes thousands of options to choose from. The recommendation systems among sectors make a huge influence on our society and they are very useful for people to make proper decisions.

Usually, recommendation systems are based on a rating scale from 1 to 5 grades or stars, with 1 indicating the lowest satisfaction and 5 being the highest satisfaction. Using a movie recommendation system benefits three main clusters. The streaming service platforms can increase their profits by accurate users' needs. By learning from their history information, users can quickly recognize their choices which can enjoy movies better and avoid the trial-and-error cost. Besides, the system is also convenient for movie producers to plan a new movie with the trend of their target audiences.

The aim of this project is to develop a machine learning algorithm using the inputs in one subset to predict movie ratings in the validation set. Several machine learning algorithm has been used and results have been compared to get the maximum possible accuracy in prediction.

2. Methodology

There are two types of recommendation systems. Content-based methods are based on the similarity of movie attributes. Using this type of recommender system, if a user watches one movie, similar movies are recommended. With this in mind, the input for building a content-based recommender system is movie attributes.

With collaborative filtering, the system is based on past interactions between users and movies. With this in mind, the input for a collaborative filtering system is made up of past data of user interactions with the movies they watch.

3. Data Set

In this project, I create a movie recommendation system using the MovieLens dataset. It is collected and made available rating data sets from the MovieLens web site, a movie recommendation service by GroupLens Research lab from the University of Minnesota (Harper, Konstan, 2015).

In this project, I will use the 10M version in this project. The dataset includes the encoding information of user, movie, rating and timestamp. It contains 100,000 tag applications applied to 10,000 movies by 72,000 users. These sample lines of data set are reported in Table 1. Users were selected at random for inclusion. All selected users had rated at least 20 movies. Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

In order the better view the data set, I split the release year, genres and rated year from title. The variables used in this project are the user ID, movie ID, rated year, released year and ratings. Descriptive statistics are reported in Table 2.

Table 1 Sample lines of data set

Sample lines of MovieLens

userId	movieId	rating	timestamp	title	genres
48,470	1,721	1	939,350,658	Welcome to Woop-Woop (1997)	Comedy
21,089	1,405	4	975,532,673	Meet Wally Sparks (1997)	Comedy
196	1,394	3	981,579,799	Turbulence (1997)	Action| Thriller
58,657	7,143	4	1,187,290,972	Touching the Void (2003)	Adventure| Documentary
67,532	1,213	5	907,940,436	Terminator, The (1984)	Action| Sci-Fi| Thriller

Table 2 Descriptive Statistics

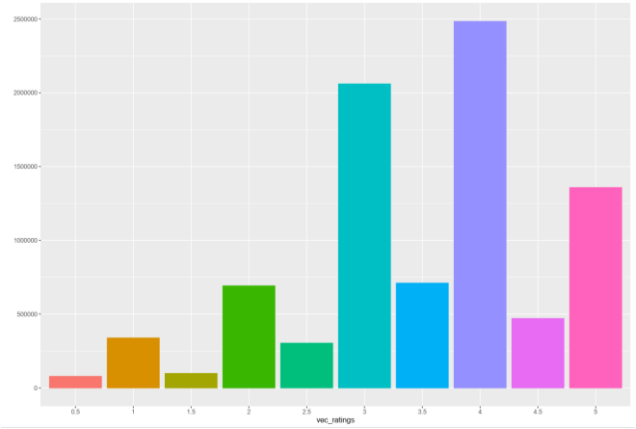
Summary of MovieLens

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
userId	9,543,704	35,851.820	20,604.360	1	18,056	53,609	71,567
movieId	9,543,704	2,274.079	2,037.376	1	595	3,258	9,019
rating	9,543,704	3.512	1.064	0	3	4	5
year	9,543,704	1,986.722	15.904	1,915	1,983	1,996	2,006

3.1 Data exploring

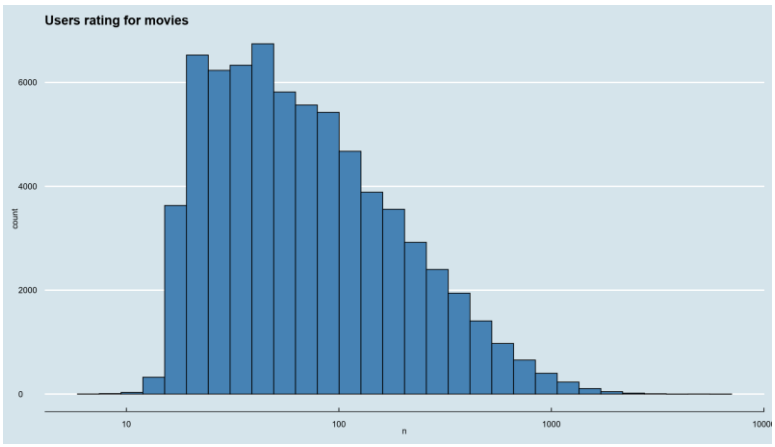
The distribution of the movie ratings shows in Graph 1. It shows a range of 0.5 to 5. The whole numbers used more often and users have a general tendency to rate movies between 3 and 4.

Graph 1 Rating Distribution



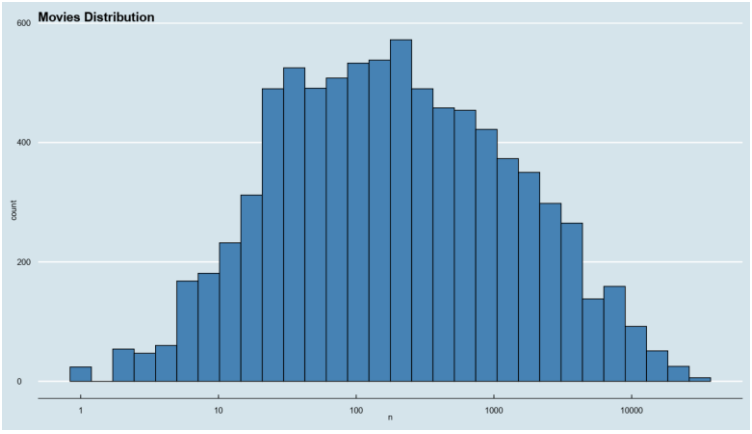
The users' ratings distribution in Graph 2 shows a right skew of normal distribution. It indicates that not every user is equally active. Some users have rated very few movies and their opinion may bias the prediction results. It represents the user bias.

Graph 2 Users' Rating Distribution



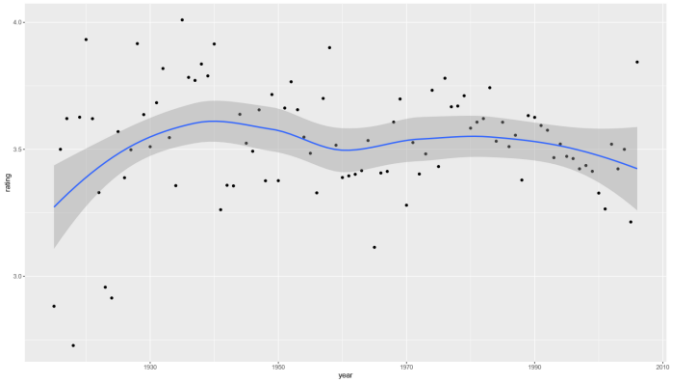
The movie's distribution is reported in Graph 3. The histogram shows some movies have been rated very few numbers of times. So, they should be given lower importance in movie prediction. It represents the movie bias.

Graph 3 Movie Distribution



Last, the Graph 4 is the rating and the movie release year. It represents a general trend of movie viewers and their rating habits can be explored.

Graph 4 Rating and Release Year Distribution



4. Model

4.1 Model Evaluation Functions

The quality of the model in this project will be assessed by the Root Mean Squared Error is the square root of the MSE. It is the typical metric to evaluate recommendation systems. it is Similar to MSE, the RMSE penalizes large deviations from the mean and is appropriate in cases that small errors are not relevant. Contrary to the MSE, the error has the same unit as the measurement.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

During the model development, I use the train set to develop the model, test set to predict the outcome. When the model is ready, then I use the validation set.

4.2 Baseline Model

$$\hat{y} = \mu + \epsilon_{u,i}$$

The easiest and simplest way to do this is to recommend the most popular items. I take this as the baseline model. It's simply a model which ignores all the feathers and simply calculates mean rating. The RMSE of this model is about 1.06 which is very high.

Summary of RMSE

Method	RMSE
Model 1: Initial Prediction	1.06394639143823

4.3 Linear Model with Movie Effect

I am building the first linear model based on the formula:

$$\hat{y} = \mu + b_i + \epsilon_i$$

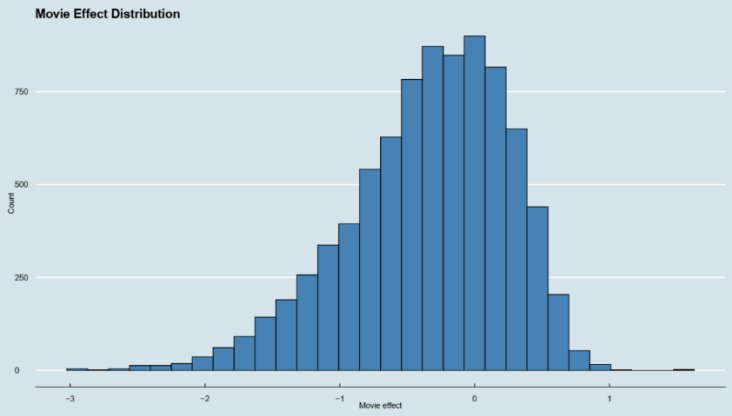
An improvement in the RMSE is achieved by adding the movie effect.

Summary of RMSE

Method	RMSE
Model 1: Initial Prediction	1.06394639143823
Model 2: With Movie bias	0.946423994112882

Different movies are rated differently. As shown in the exploration, the histogram is not symmetric and is left skewed towards negative rating effect. The movie effect can be taken into account by taking he difference from mean rating.

Graph 5 Movie Effect Distribution



4.4 Linear Model with Movie and User Effect

I am building the second linear model based on the formula:

$$\hat{y} = \mu + b_i + b_u + \epsilon_{u,i}$$

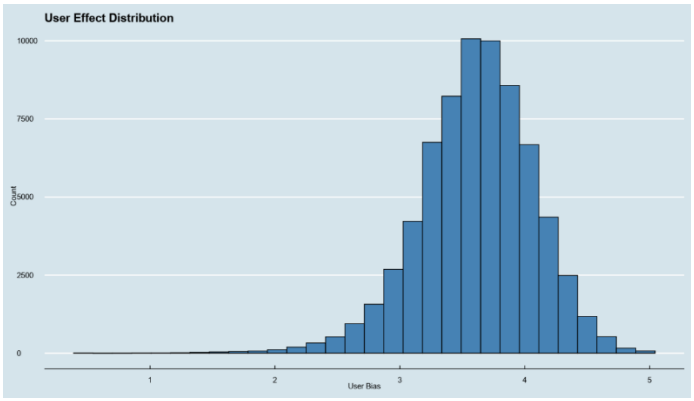
Given that movie and users biases both obscure the prediction of movie rating, a further improvement in the RMSE is achieved by adding the user effect.

Summary of RMSE

Method	RMSE
Model 1: Initial Prediction	1.06394639143823
Model 2: With Movie bias	0.946423994112882
Model 3: With Movie & User bias	0.871483095169582

Different users are different in terms of how they rate movies. Some cranky users may rate a good movie lower or some very generous users just don't care for assessment. From the Graph 6 we can see is nearly normally distributed.

Graph 6 User Effect Distribution



4.5 Regularization

We have noticed in our data exploration; some users are more actively participating in movie reviewing. There are also users who have rated very few movies. On the other hand, some movies are rated very few times. These are basically noisy estimates that we should not trust. Additionally, RMSE are sensitive to large errors. Large errors can increase our residual mean squared error. So we must put a penalty term to give less importance to such effect.

The estimated value can be improved adding a factor that penalizes small sample sizes and have have little or no impact otherwise. Thus, it can be calculated with formulas:

$$\hat{b}_i = \frac{1}{n_i + \lambda} \sum_{u=1}^{n_i} (y_{u,i} - \hat{\mu})$$

$$\hat{b}_u = \frac{1}{n_u + \lambda} \sum_{i=1}^{n_u} (y_{u,i} - \hat{b}_i - \hat{\mu})$$

The approach utilized in the above model is implemented below with the movie and user effect added genres and release year effects.

Summary of RMSE

Method	RMSE
Model 1: Initial Prediction	1.06394639143823
Model 2: With Movie bias	0.946423994112882
Model 3: With Movie & User bias	0.871483095169582
Regularized Movie and User Effect Model	0.869858850830369
Regularized Movie, User, Year, and Genre Effect Model	0.861596977091807

4.6 Random Forest

The previous models are belonging to content-based methods, next part I will focus on two collaborative methods.

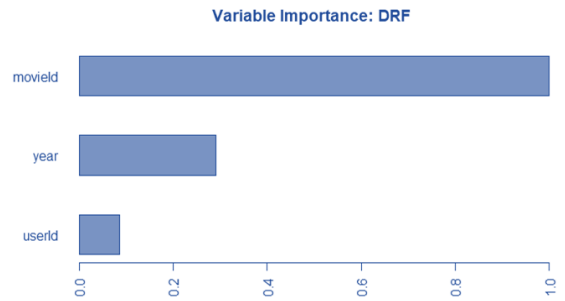
For the specification of the first algorithm model, I set the Random Forest is to repeatedly and randomly select samples from the original training sample set to generate a new training sample set and then generate 50 classification trees according to the sample set to form a random forest.

Summary of RMSE

Method	RMSE
Model 1: Initial Prediction	1.06394639143823
Model 2: With Movie bias	0.946423994112882
Model 3: With Movie & User bias	0.871483095169582
Regularized Movie and User Effect Model	0.869858850830369
Regularized Movie, User, Year, and Genre Effect Model	0.861596977091807
Random Forest Model	0.971673248435401

The lowest RMSE surprisingly was achieved only with user ID and released year and Movie ID**.** The algorithm automatically drops other variables which is not importance.

Graph 7 Variable Importance in Random Forest



4.7 K-Nearest Neighbors

Second algorithm model uses K-nearest neighbors' method. It is defined some distance metric between the items in your dataset, and find the K closest items. First, I transform the data frame of ratings into a proper format that can be consumed by a KNN model. I reshaped data frame of ratings to the wide format with movies as rows and users as columns. Then I filled the missing observations with zero. The number of nearest neighbors in this project is 25.

Summary of RMSE

Method	RMSE
Model 1: Initial Prediction	1.06394639143823
Model 2: With Movie bias	0.946423994112882
Model 3: With Movie & User bias	0.871483095169582
Regularized Movie and User Effect Model	0.869858850830369
Regularized Movie, User, Year, and Genre Effect Model	0.861596977091807
Random Forest Model	0.971673248435401
K-Nearest Neighbors	0.988195265456561

5. Results

As we can see from the result table, the best performance in training set is regularization with movie, user, year and genres. So, finally we train the complete training set with the model and calculate the RMSE in the validation set. As expected, the RMSE calculated on the validation set slightly higher than the test set but generally lower than baseline model. Therefore, the project goal is achieved.

Summary of RMSE

Method	RMSE
Model 1: Initial Prediction	1.06394639143823
Model 2: With Movie bias	0.946423994112882
Model 3: With Movie & User bias	0.871483095169582
Regularized Movie and User Effect Model	0.869858850830369
Regularized Movie, User, Year, and Genre Effect Model	0.861596977091807
Random Forest Model	0.971673248435401
K-Nearest Neighbors	0.988195265456561
Final Validation Model	0.864646479779101

6. Limitation and Future Work

The limitations in this project are mainly three types. First, only few predictors are mainly used, the movie, user and year information, not considering other features like genres in the data set. Also, there are other types of movie information like IMDB ratings and Tomatometer scores. Second, for random forest, the model might performance better when increase the numbs of tree. Last, for KNN model, 25 may not be the best neighbor value.

This report briefly describes simple models that predicts ratings. There is another widely adopted approaches not discussed here: Matrix Factorization could be use in the future work.

Reference

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872