Predicting User Preference for Movies: A Model Comparison Approach

M. Sääksjärvi

Submission for Data Science: Capstone course

1/19/2019

Executive summary

Nowadays, a wealth of data is being collected by academia, business, and governments. A popular way of using this data is for recommender systems, i.e., systems that help users find new products and services based on their prior preferences. An area in which recommender systems are widely used is in the context of movie recommendations. Movie recommendations are used for suggesting new movies to a user based on his/her past movie preferences. Using the MovieLens dataset, I create three different models for this purpose. The models are compared on the basis of their predictive accuracy on the basis of RMSE. The results show that the recommender model that includes user and movie bias outperforms both a basic model as well as a model in which only the movie bias is included based on the RMSE value.

Introduction

Recommender systems (RS) are used to help users find new items or services (e.g., books, music, transportation) based on information about the user and his/her past behavior (Adomavicius and Tuzhilin 2005). Essentially, recommender systems apply statistical techniques to make predictions about the users future behavior on the basis on previously recorded data (Sarwar, Karyptis, Konstan, and Riedl 2000). Recommender systems have become popular in recent years. Several market industry leaders use recommender systems (e.g., Amazon, Netflix, and Pandora), as such systems are good for business: they help improve conversion rates, promote cross-selling, and foster the relationship with the user (Shafer, Konstan, and Riedl 2001). The importance of being able to predict a user's future preferences was exemplified by the Netflix competition, in which the company awarded a $1M prize for improving its predictions (based on root mean square error) of 10% (Lohr 2009). Clearly, recommender systems have become big business, in which companies abilities to "guesstimate" a user's future preference is becoming a competitive advantage in many markets.

Recommender systems are a major application of machine learning. Machine learning is "the field of scientific study that concentrates on induction algorithms and on other algorithms that can be said to 'learn'." (Kohavi and Provost 1998). In other words, machine learning centers on the creation of algorithms that can make predictions based on a learning process (Silva, Jr. 2016). Based on training data (training set) that is provided as input, machine learning algorithms make data-driven predictions and decisions, the viability of which are tested by the use of a test set.

Data

In this project, we will be using the MovieLens dataset for creating a movie recommender system. The Movielens 10M dataset is available at https://grouplens.org/datasets/movielens/, and contains 10000054 ratings of 10681 movies by 71567 users (Maxwell Harper and Konstan 2015). In the movielens dataset, movie preference is provided by a rating, which ranges from 1 (a strong dislike) to 5 (a strong like). All users included in the dataset have rated at least 20 movies.

Getting the data

The dataset is imported into R from the movielens homepage (see above). The code for data import and the creation of training and testing sets is provided on the course website: https://courses.edx.org/courses/course-v1:HarvardX+PH125.9x+2T2018/courseware/dd9a048b16ca477a8f0aaf1d888f0734/e8800e37aa444297a3a2f35bf84ce452/?child=last. The imported data is split into a training and testing set using a 90-10 split, with 90% of the data being allocated to training (dataset edx), and 10% to testing (dataset validation). We will train our model on the edx dataset.

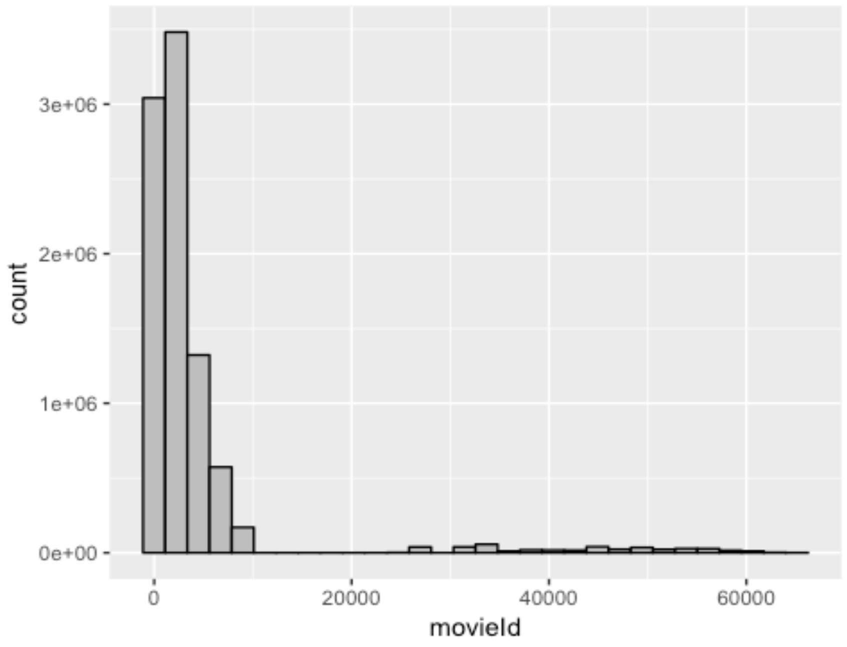
Exploratory analysis

There are 9000055 rows in the edx dataset, and 6 columns. The six columns are userId, movieId, rating, timestamp, title, and genres. There are 10677 distinct movies that are being rated, with 69878 distinct users serving as raters.

```
#check the number of rows in the edx file
nrow(edx)

## [1] 9000055

#check the number of columns in the edx file
ncol(edx)

## [1] 6

#check the number of distinct movies rated
n_distinct(edx$movieId)

## [1] 10677

#check the number of distinct users that rated the movies
n_distinct(edx$userId)

## [1] 69878
```

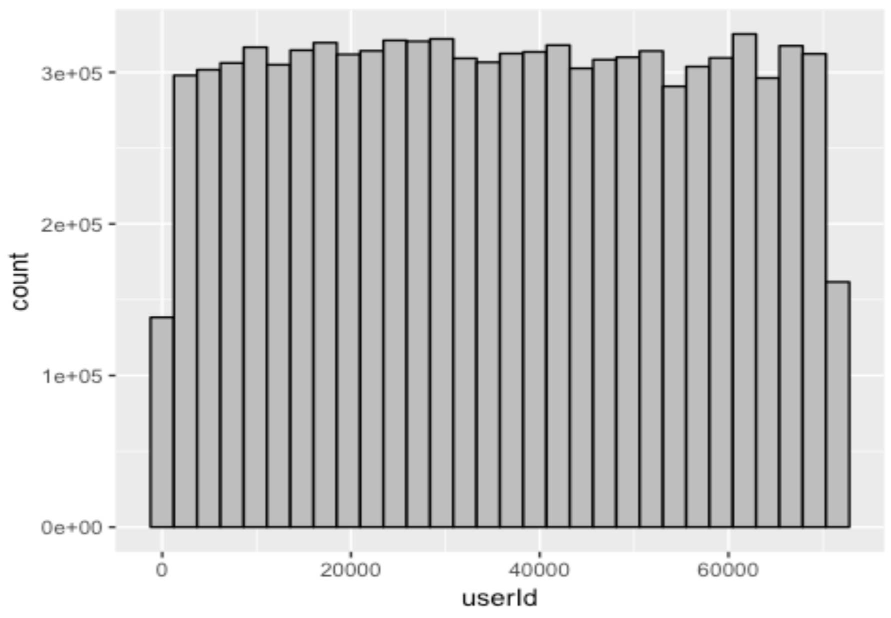
Distribution of movies and users. The distribution of the movie and the user variables provides insight into the dataset. Based on the distribution of the movie variable, we may conclude that some movies get reviewed much more than others. This is because Blockbuster movies garner a lot more attention and viewership, whereas some movies (e.g., arthouse or foreign language films) mainly cater to niche audiences.

```
#plot the distribution of movies
a <- ggplot(edx, aes(x = movieId))
a + geom_histogram(bins = 30, color = "black", fill = "gray")
```



The user variable shows that most users seem to have rated a significant number of movies. This may not be that surprising, as the dataset only included users that had rated at least 20 movies. But even within this cohort of users, we can see that some users are more active than others at rating movies.

```
#plot the disribution of users
b <- ggplot(edx, aes(x = userId))
b + geom_histogram(bins = 30, color = "black", fill = "gray")
```



For the analysis, we convert rating and userId to numeric variables.

```
edx$rating <- as.numeric(edx$rating)
edx$userId <- as.numeric(edx$userId)
```

Comparison standard. For purposes of model comparison, we need a comparison standard on the basis of which different models can be compared. To compare different models, RMSE was selected as the primary accuracy metric for model comparison.

```
#function that computes this residual means squared error for a vector of ratings and their corre
sponding predictors
RMSE <- function(true_ratings, predicted_ratings) {
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Model comparison

Naive model. Using a matrix factorization approach, we start by building the simplest possible model for a recommendation system. The simplest possible model assumes the same rating for all movies, regardless of the user and movie. The naive model that displys these assumptions would involve predicting Y on the basis of the mean rating (mu_hat, simplified to mu in the analysis) and random variation. We compute the mean on the training data (edx) and then compute the residual mean squared error on the test data (validation).

```
#simplest possible model: average rating of all movies across all users
mu_hat <- mean(edx$rating)
mu_hat

## [1] 3.512465

#compute the residual mean squared error of the simplest possible model on the test set data
naive_rmse <- RMSE(validation$rating, mu_hat)
naive_rmse

## [1] 1.061202
```

The simplest possible model yield as RMSE of 1.061202, which is above 1, the general cut-off point for RMSE. So we may conclude that this simple model does not perform that well. We would need to add other variables into our model to improve its performance.

Model 1: accounting for movie bias. We know, based on experience, that some movies get higher ratings than others. Rating sites, such as Rotten Tomatoes, display scores that range widely over different movies. So including the average rating for each movie should improve the predictive ability of our model. So we proceed by including the movie bias into our model.

```
#start optimizing model: include movie bias
mu <- mean(edx$rating)
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

We compare the prediction yielded by this improved model in the training set by comparing it with the testing set, and estimating its RMSE.

```
#compare prediction in training set with validation set
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by="movieId") %>%
  .$b_i
```

```
#calculate rmsea for model
model_1_rmse <- RMSE(predicted_ratings, validation$rating)
model_1_rmse

## [1] 0.9439087
```

This model yields an RMSE of 0.9439087, which is already an improvement over the naive model.

Model 2: accounting for user bias. We also know, based on the experience of reading reviews, that users rate movies differently. Some users tend to be very positive about the movies they see, whereas others are very critical. We can take this into account by including the average rating by each user into our model. Including the user bias into the model should improve its predictive ability, as by taking into account the user bias, we should be able to explain more of the variance in Y. So we proceed by including the user bias into our model.

```
user_avgs <- validation %>%
  left_join(movie_avgs, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

We compare the prediction yielded by this improved model in the training set by comparing it with the testing set, and estimating its RMSE.

```
predicted_ratings <- validation %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by="userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
#calculate rmse for model
model_2_rmse <- RMSE(predicted_ratings, validation$rating)
model_2_rmse

## [1] 0.8292477
```

This model yields an RMSE of 0.8292477, which is already a significant improvement over the naive model.

Final model. Based on an RMSE comparison across all three models, we may conclude that the model that includes movie bias and user bias (model_2) is superior to a model that only including avergae ratings (the naive model) and to a model that only includes movie bias (model_1). Given the significant improvement in RMSE, model_2 is the final model.

Model tested	RMSE
Naive model	1.061202
model_1	0.9439087
model_2 **(final model)**	0.8292477

Conclusions

In this report, we focused on creating a recommendation systems based on machine learning algorithms in the context of movie recommendations. Movie recommendations are used by market leaders such as Netflix to accurately predict the movies that users will like. Using a matrix factorization approach, we find that a model that accounts for movie and user bias outperforms a naive model with only average ratings included, as well as a model that only accounts for movie bias.

References

Adomavicius, G., and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749.

Harper, F. M. and J. A. Konstan (2015). The MovieLens datasets: History and context. ACM Transactions on Interactive Intelligent Systems (TiiS), 5(4), Article 19.

Kohavi, R. and Provost, F. (1998) Glossary of terms. Machine Learning, 30, 271-274.

Lohr, S. (2009) Netflix awards $1 million prize and starts a new contest. New York Times, available at https://bits.blogs.nytimes.com/2009/09/21/netflix-awards-1-million-prize-and-starts-a-new-contest/.

Mitchell, T. M. (1999). Machine learning and data mining. Communications of the ACM, 42 (11), 30-36.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In Proceedings of the 2nd ACM conference on Electronic commerce, 158-167.

Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). E-commerce recommendation applications. Data mining and knowledge discovery, 5(1-2), 115-153.