Recommendation System based on MovieLens Data

James Portolese

November 20, 2019

R Markdown

Introduction

The goal of this project is to identify machine learning techniques to predict ratings for movies and create a movie recommendation system based on an individual users ratings as well as movies the user reviewed. A model will be constructed to perform the predictions and parameters will be added to the model. The parameters will be based on statistical averages (overall, grouped by movie, grouped by user) and these will help predict the rating a user would give to a movie.

Once the model is built the Root Mean Square Error will be calculated and compared to the simple_model (assume the overall average). In a final attempt to lower the RMSE the technique or regularization will be applied to handle the bias generated by movies with few ratings and users with few reviews. The steps I took can be found in the document below.

Data Set

The project data is a small subset of a much larger dataset that includes movie ratings from a set of users. Not all movies were rated by all users and not all users rated each movie. The rating scale was between 1 and 5 stars. Training and a validation set were created using the code provided. The overall record count was approximately 10 million records. The training set (edx) contains about 9 million records and the validation set (validation) is about 1 million (10%). There is no overlap between these sets and the validation set will be held out of the steps to train the model Only after the model parameters are finalized will we use the validations set and a final RMSE will be calculated.

Methodology

I decided to outline the steps in the paragraphs below and include the code for easier reading. So lets get started.

Our first order of business is to create test and training sets of the data to use in the project. Since I didn't write the code I will not display it in this document. It can be found in my .r file submitted for the class. The results from running the code are the training set (edx) and the test set (validation) for use in the remainder of the project.

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------------------------------
------------------------------------------------------------ tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.0     v purrr   0.3.2
## v tibble  2.1.1     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'dplyr' was built under R version 3.6.1
```

```
## -- Conflicts ------------------------------------------------------------------------------
------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
nrow(edx)
```

```
## [1] 9000061
```

```
nrow(validation)
```

```
## [1] 999993
```

** The results of the above code provide us with two non-overlapping datasets edx (training set of 9,000,061 records) and validation (the validation set 999,993 records) **

Since we will test our model iterations by calculating the RMSE, we need a function to report back the RMSE

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

We start with the simplest model that assumes all unknown ratings are just the average of all ratings and we calculate the RMSE for a model that assumes the average for all unknown predictions:

```
mu_hat <- mean(edx$rating)
mu_hat
```

```
## [1] 3.512464
```

```
simple_rmse <- RMSE(validation$rating, mu_hat)
simple_rmse
```

```
## [1] 1.060651
```

With a RMSE of over 1, our predictions average being off by 1 star. We can obviously do better. The next step would be to consider the average rating grouped by movie and the average rating grouped by user. This code will add two parameters to our model and these along with the overall average will be used to make new predictions.

First the movie averages - the results of this table include each movie and the residual (amount above or below the overall average) for each movie

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu_hat))
```

and then the user averages - the results here of this table include each user and the residual (amount higher or lower then the average user rating) for each user

```
user_avgs <- edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat))
```

Now we add them to the model and use these to get more detailed predictions. So now our model takes into account user and movie variation in the dataset

```
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(prediction = mu_hat + b_i + b_u) %>%
  .$prediction

better_model <- RMSE(validation$rating, predicted_ratings)
better_model
```

```
## [1] 0.885652
```

By including the movie and user variation into the model we improve the model significantly from 1.06 to 0.885. We may be able to do better by considering Regularization. Since our model parameters are heavily influenced by obscure movies and users that rarely review movies we can apply regularization to identify a lambda that will help minimize these effects. These outliers will have less of an effect on the overall prediction and give us (hopefully) a more accurate model

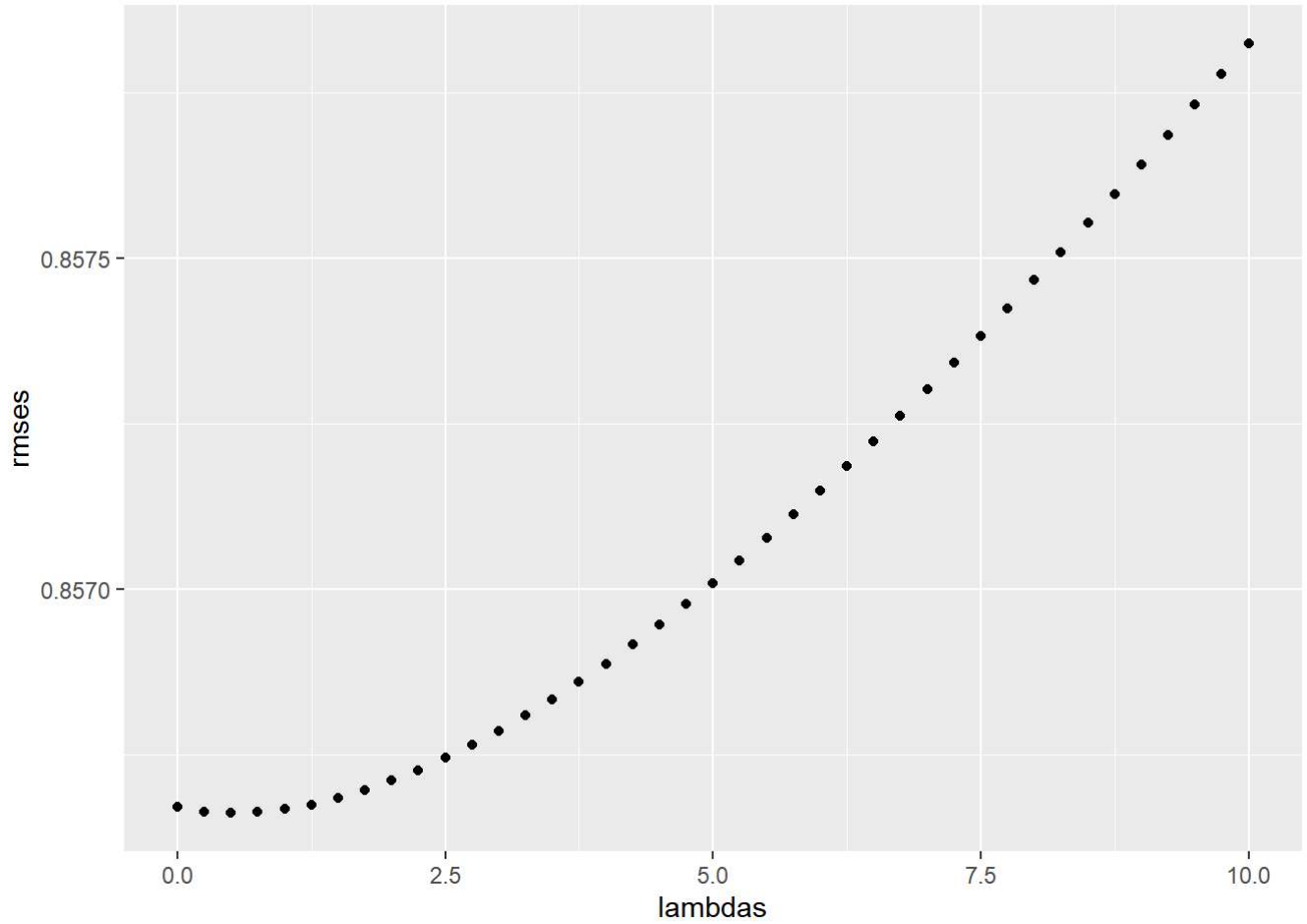
set lambda to shrink the b_i and b_u towards zero for movies with small numbers of ratings.

We will use the following code on the training set (edx) to find the appropriate lambda value

```
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/ (n()+l))

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId)  %>%
    summarize(b_u = sum(rating - b_i - mu)/ (n()+l))

  predicted_ratings <- edx %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    mutate(prediction = mu + b_i + b_u) %>%
    .$prediction

  return(RMSE(predicted_ratings, edx$rating))

})
```

Now we will plot the lambdas against the RMSE generated and pick the one that generates the lowest RMSE

```
qplot(lambdas, rmses)
```



Identify the lambda that minimized the RMSE and use it against our validation set.

```
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 0.5
```

Armed with the lambda value we can now apply it to the model using this code.

```
# apply lambda to the validation set (validation)
mu <- mean(validation$rating)

b_i <- validation %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/ (n()+lambda))

b_u <- validation %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId)  %>%
  summarize(b_u = sum(rating - b_i - mu)/ (n()+lambda))

predicted_ratings <- validation %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  mutate(prediction = mu + b_i + b_u) %>%
  .$prediction
```

Now calculate the regularized RMSE error.

```
Reg_model <- RMSE(predicted_ratings, validation$rating)
Reg_model
```

```
## [1] 0.8260111
```

Results

The initial simple_model based on only the overall average rating performed poorly as expected. A RMSE of over 1 means that our predictions had an average of 1 star difference when run over the test set.

My next step was to consider the movie effect. Some movies naturally have higher ratings than others so if we assume an overall average we don't make effective predictions.

Just as some movies have higher ratings overall than others some users give better ratings than other users. I accounted for both the movie and user effect in my model and this lowered the RMSE to 0.885

The final step was to eliminate the bias inherent in the model due to small sample sizes (movies that had few ratings and users that have few reviews), By applying regularization to the model parameters we eliminate the bias in the obscure movie reviews as well as the users who only reviewed a few movies. We see an decrease in RMSE from .885 to .826

Conclusion

Accounting for the user and movie effect made a significant improvement in the Root Mean Square Error of the model vs. simply assigning the average movie rating. By also performing regularization to reduce the bias a small but substantial improvement was seen. A final potential step to reduce the RMSE would be to identify trends in the data using PCA or SVD and add additional parameters to the model to deal with the correlation between movie and users types. There are a lot of variables that we have considered (genres, actors, and even time ratings were submitted might be trends we could identify to help increase the prediction accuacy.