MovieLens Project Report

Nabeel Arif

17/06/2020

Introduction

Recommendation systems are a widely used tool to improve user experience by making tailored suggestions based on historical data. This is done so by predicting what rating a user will give a particular product and then recommend those products that are predicted a high rating.

Netflix uses such a recommendation system to predict the rating (from 1 star to 5 stars) a user would give to a specific movie. Using the Netflix example, we will build our own recommendation system and identify how well it performs by calculating the residual mean squared error (RMSE). We will build an initial model and improve this to get the RMSE below a target of 0.86490.

Data Exploration

For this project we will be using the MovieLens 10M dataset which will be split into a training set (edx) and test set (validation), and account for 90% and 10% of the MovieLens dataset respectively. We can see the dataset below and the split of data for both sets.

```r
# overview of training dataset
edx %>% as_tibble()
```

```
## # A tibble: 9,000,055 x 6
##    userId movieId rating timestamp title                  genres
##     <int>   <dbl>  <dbl>     <int> <chr>                  <chr>
##  1      1     122      5 838985046 Boomerang (1992)       Comedy|Romance
##  2      1     185      5 838983525 Net, The (1995)        Action|Crime|Thriller
##  3      1     292      5 838983421 Outbreak (1995)        Action|Drama|Sci-Fi|T~
##  4      1     316      5 838983392 Stargate (1994)        Action|Adventure|Sci-~
##  5      1     329      5 838983392 Star Trek: Generation~ Action|Adventure|Dram~
##  6      1     355      5 838984474 Flintstones, The (199~ Children|Comedy|Fanta~
##  7      1     356      5 838983653 Forrest Gump (1994)    Comedy|Drama|Romance|~
##  8      1     362      5 838984885 Jungle Book, The (199~ Adventure|Children|Ro~
##  9      1     364      5 838983707 Lion King, The (1994)  Adventure|Animation|C~
## 10      1     370      5 838984596 Naked Gun 33 1/3: The~ Action|Comedy
## # ... with 9,000,045 more rows
```

```r
# number of rows in training set, edx
nrow(edx)
```

```
## [1] 9000055
```

```
# number of rows in test set, validation
nrow(validation)
```

```
## [1] 999999
```

From this we can see that each row represents a rating given by one user to one movie. We ensure that userId and movieID values that appear in the test set also appear in the training set by using the semi_join function. We can check if there are any missing values within the datasets.

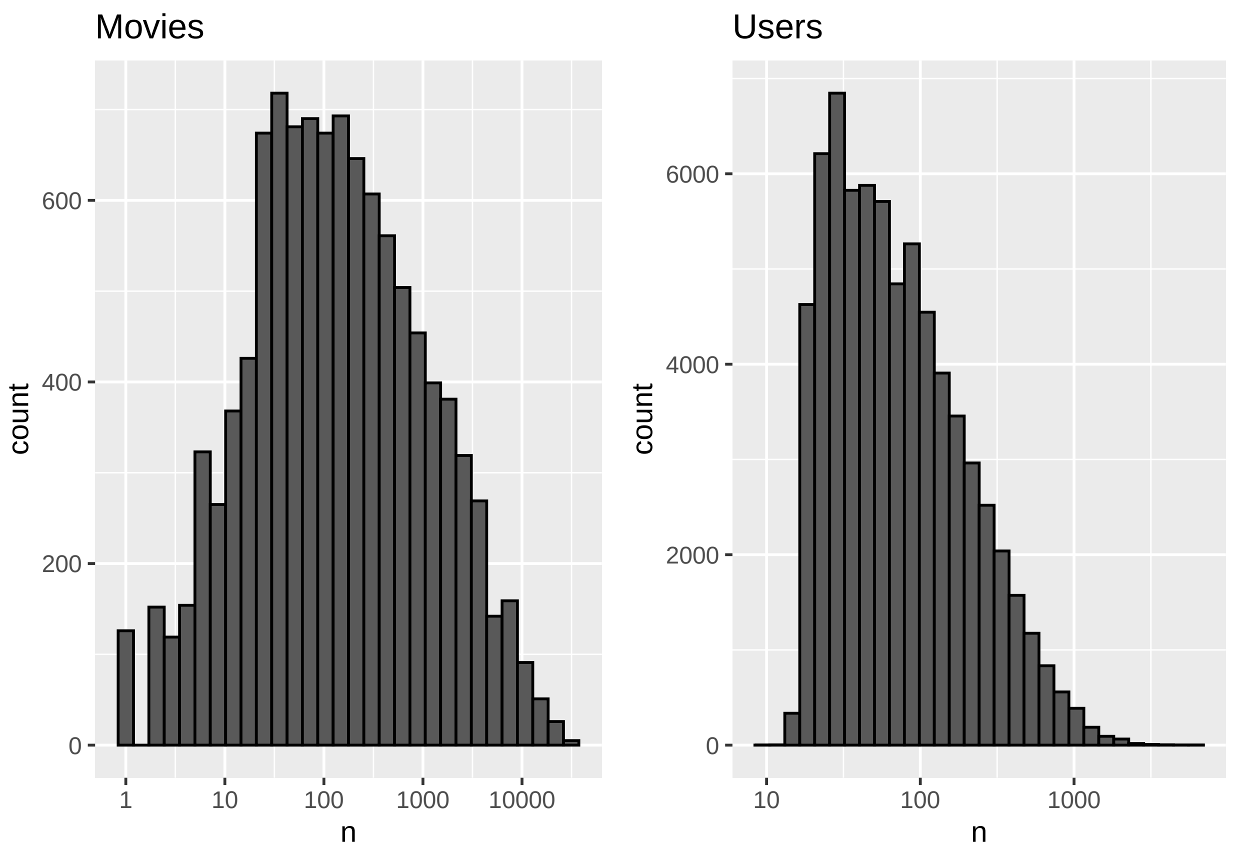
```
# check for any missing values in training set
any(is.na(edx))
```

```
## [1] FALSE
```

```
#check for any missing values in test set
any(is.na(validation))
```

```
## [1] FALSE
```

Visualizing the test data we can see that the number of ratings per movies varies as does the number of ratings submitted by each user.



Further we can see that ratings across movies differ. Below are the top 10 rated movies with at least 100 user ratings:

movieId	title	Rating_Count	Rating_Average
318	Shawshank Redemption, The (1994)	28015	4.455131
858	Godfather, The (1972)	17747	4.415366
50	Usual Suspects, The (1995)	21648	4.365854
527	Schindler's List (1993)	23193	4.363493
912	Casablanca (1942)	11232	4.320424
904	Rear Window (1954)	7935	4.318651
922	Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	2922	4.315880
1212	Third Man, The (1949)	2967	4.311426
3435	Double Indemnity (1944)	2154	4.310817
1178	Paths of Glory (1957)	1571	4.308721

Lastly, we can see the "best" rated movies by their individual average rating

movieId	title	Rating_Count	Rating_Average
3226	Hellhounds on My Trail (1999)	1	5.00
33264	Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	2	5.00
42783	Shadows of Forgotten Ancestors (1964)	1	5.00
51209	Fighting Elegy (Kenka erejii) (1966)	1	5.00
53355	Sun Alley (Sonnenallee) (1999)	1	5.00
64275	Blue Light, The (Das Blaue Licht) (1932)	1	5.00
5194	Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	4	4.75
26048	Human Condition II, The (Ningen no joken II) (1959)	4	4.75
26073	Human Condition III, The (Ningen no joken III) (1961)	4	4.75
65001	Constantine's Sword (2007)	2	4.75

and the "worst" rated movies

movieId	title	Rating_Count	Rating_Average
5805	Besotted (2001)	2	0.5000000
8394	Hi-Line, The (1999)	1	0.5000000
61768	Accused (Anklaget) (2005)	1	0.5000000
63828	Confessions of a Superhero (2007)	1	0.5000000
64999	War of the Worlds 2: The Next Wave (2008)	2	0.5000000
8859	SuperBabies: Baby Geniuses 2 (2004)	56	0.7946429
7282	Hip Hop Witch, Da (2000)	14	0.8214286
61348	Disaster Movie (2008)	32	0.8593750
6483	From Justin to Kelly (2003)	199	0.9020101
604	Criminals (1996)	2	1.0000000

We will build our models for prediction using our findings from the data exploration.

Method & Model Building

Model 1 - Simple Model

We start with the simplest model where we predict the same rating for all movies regardless of user. Such a model would be as below.

$$Y_{u,i} = \mu + \varepsilon_{u,i}$$

Where $Y_{u,i}$ is the predicted rating of user u and movie i and $\varepsilon_{i,u}$ independent errors sampled from the same distribution centered at 0 and μ the average rating for all movies. We know that the estimate that minimizes the RMSE is the least squares estimate of μ (for code simplicity we're referring to this as just mu as opposed to mu_hat).

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

```
## [1] 3.512465
```

So on average a movie is rated 3.5 out of 5. We can define RMSE using the below mathematical formula:

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

where N is the number of user/movie combinations, $y_{u,i}$ is user u's rating of movie i and $\hat{y}_{u,i}$ is our prediction.

In R we can create a function that we can use for our multiple models.

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

Using our value for the estimated average we can calculate the RMSE for our first model.

```
rmse_simple <- RMSE(validation$rating, mu)
```

Model	Predicted_RMSE
Simple/Average	**1.061202**

With a RMSE value above 1 we are quite far away from our RMSE target so we need to look at ways to improve our model.

Model 2 - Movie Bias

From the data we can see that some movies are rated higher than others. We can therefore add an extra term to our model, b_i, to represent the average ranking for movie i.

$$Y_{u,i} = \mu + b_i + \varepsilon_{u,i}$$

We can now calculate the RMSE for our second model by finding all the movie averages.

```r
# calculate movie bias
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

# predict ratings using validation dataset
predicted_ratings <- validation %>%
  left_join(b_i, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)

# calculate the RMSE
rmse_movie <- RMSE(validation$rating, predicted_ratings)
```

Model	Predicted_RMSE
Simple/Average	1.0612018
Movie Bias Model	**0.9439087**

We can continue to improve our model by taking into account other bias.

Model 3 - Movie & User Bias

Just as there is a movie bias, the data shows there is also a user bias such that some users rate movies very harshly and others more softly. This bias, b_u, can be added to our model.

$$Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i}$$

We can estimate the user bias b_u as

$$Y_{u,i} - \mu - b_i$$

.

```r
# calculate user bias using previously calculated mean and movie bias
b_u <- edx %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

We can now update our predictions and calculate the RMSE:

```r
# predict ratings using validation dataset
predicted_ratings <- validation %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

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

Model	Predicted_RMSE
Simple/Average	1.0612018
Movie Bias Model	0.9439087
Movie + User Bias Model	**0.8653488**

We can see the RMSE improving but we still need to reduce this further.

Model 4 – Regularization with Movie Bias

From the data we can see that the "best" and "worst" rated movies often had very few reviews, often just one review. This leads to large estimates of b_i which can increase RMSE. Regularization permits us to penalize large estimates that are formed using small sample sizes.

Specifically, instead of minimizing the least squares equation, we minimize an equation that adds a penalty:

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

The first term is just least squares and the second is a penalty that gets larger when many b_i are large. Using calculus we can actually show that the values of b_i that minimize this equation are:

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

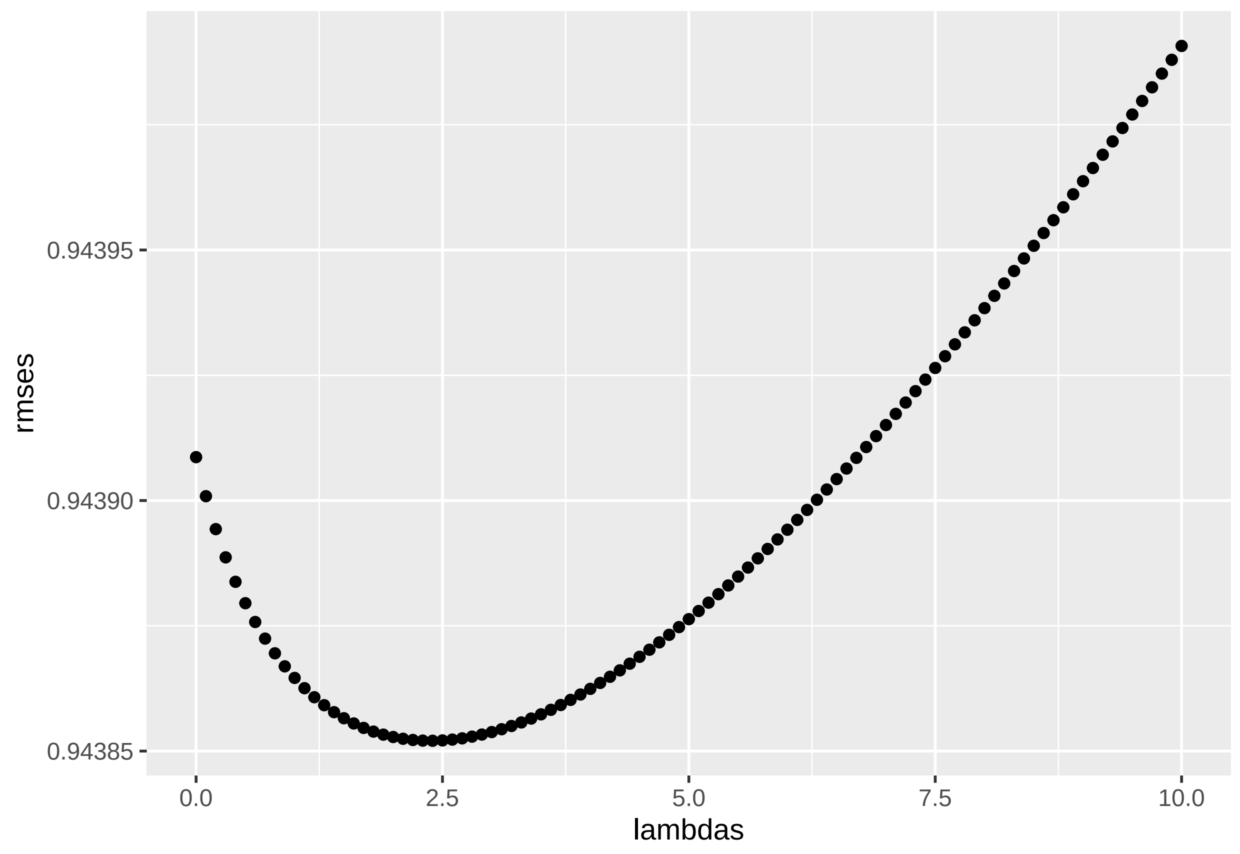
where n_i is the number of ratings made for movie i. This approach will have our desired effect: when our sample size n_i is very large, a case which will give us a stable estimate, then the penalty λ is effectively ignored since $n_i+\lambda\approx n_i$. However, when the n_i is small, then the estimate $\hat{b}_i(\lambda)$ is shrunken towards 0. The larger λ, the more we shrink.

Note however that λ is a tuning parameter and we can use cross-validation to choose it.

```r
# Generate a sequence of lambdas
lambdas <- seq(0, 10, 0.1)

# Use different values of lambda on validation dataset to generate predictions
rmses <- sapply(lambdas, function(lambda) {

   # Calculate the movie bias
   b_i <- edx %>%
      group_by(movieId) %>%
      summarize(b_i = sum(rating - mu) / (n() + lambda))

   # Generate the predicted ratings
   predicted_ratings <- validation %>%
      left_join(b_i, by='movieId') %>%
      mutate(pred = mu + b_i) %>%
      pull(pred)

   # Calculate the RMSE
   return(RMSE(validation$rating, predicted_ratings))
})
```

We can visualize how lambda affects RMSE with a plot:

```r
# plot the result of lambdas against RMSE
qplot(lambdas, rmses)
```



We can find the value of lambda which minimizes the RMSE and use the associated RMSE.

```r
# lambda value that minimizes the RMSE
min_lambda <- lambdas[which.min(rmses)]
min_lambda
```

```
## [1] 2.4
```

```r
# minimum RMSE value
rmse_reg_movie <- min(rmses)
```

Model	Predicted_RMSE
Simple/Average	1.0612018
Movie Bias Model	0.9439087
Movie + User Bias Model	0.8653488
Regularized Movie Bias Model	**0.9438521**

As we can see a regularized model provides an improvement on the equivalent non-regularized model.

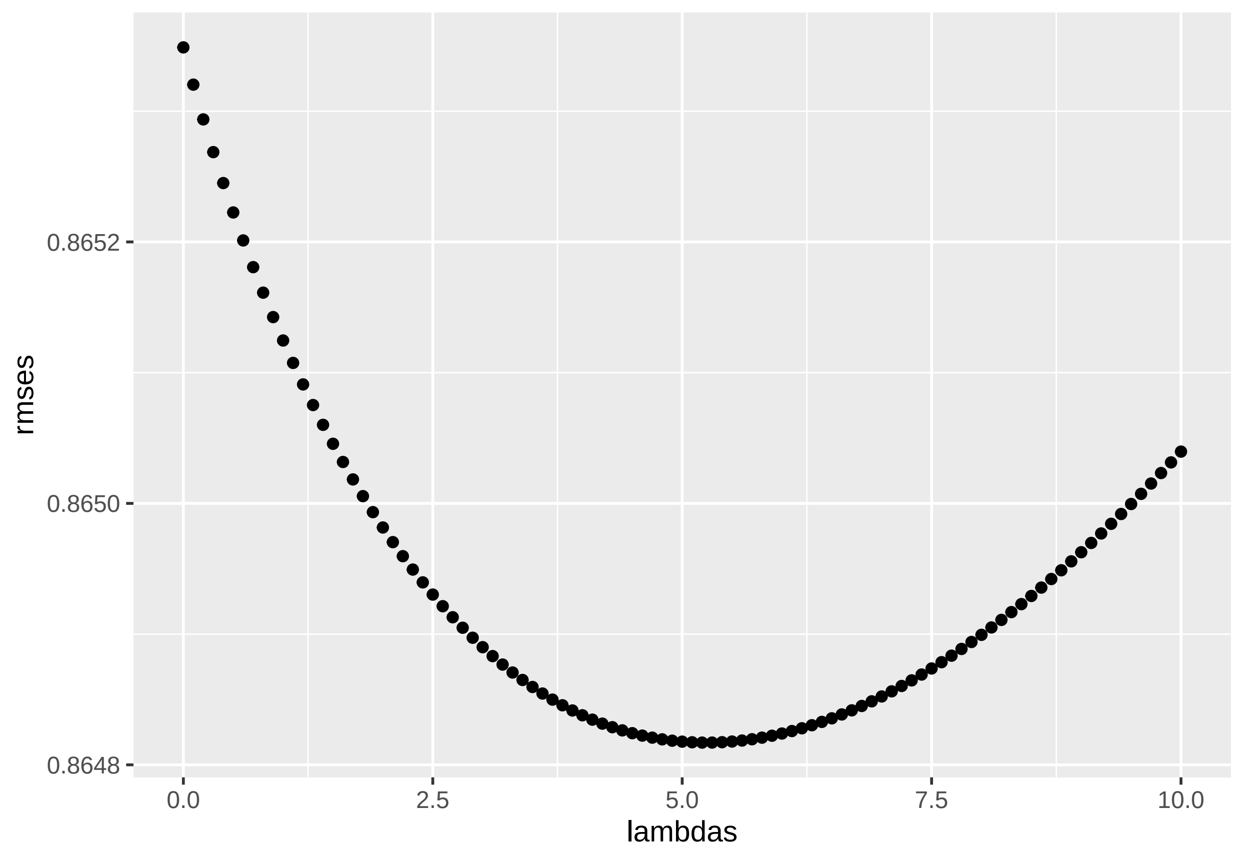
Model 5 – Regularization with Movie and User Bias

We can use regularization for the estimate of user effects also, so are adding an additional penalizing term for the user bias.

$$\frac{1}{N}\sum_{u,i}(Y_{u,i}-\mu-b_i-b_u)^2+\lambda(\sum_i b_i^2+\sum_u b_u^2)$$

Again we can use cross-validation to tune λ and find the RMSE:

```r
# Generate a sequence of lambdas
lambdas <- seq(0, 10, 0.1)

# Use different values of lambda on validation dataset to generate predictions
rmses <- sapply(lambdas, function(lambda) {

   # Calculate the average of all movies
   mu <- mean(edx$rating)

   # Calculate the movie bias
   b_i <- edx %>%
      group_by(movieId) %>%
      summarize(b_i = sum(rating - mu) / (n() + lambda))

   # Calculate the user bias
   b_u <- edx %>%
      left_join(b_i, by='movieId') %>%
      group_by(userId) %>%
      summarize(b_u = sum(rating - b_i - mu) / (n() + lambda))

   # Generate the predicted ratings
   predicted_ratings <- validation %>%
      left_join(b_i, by='movieId') %>%
      left_join(b_u, by='userId') %>%
      mutate(pred = mu + b_i + b_u) %>%
      pull(pred)

   # Calculate the RMSE
   return(RMSE(validation$rating, predicted_ratings))
})

# plot the result of lambdas against RMSE
qplot(lambdas, rmses)
```



```
# lambda value that minimizes the RMSE
min_lambda <- lambdas[which.min(rmses)]
min_lambda
```

```
## [1] 5.2
```

```
# minimum RMSE value
rmse_reg_movie_user <- min(rmses)
```

Model	Predicted_RMSE
Simple/Average	1.0612018
Movie Bias Model	0.9439087
Movie + User Bias Model	0.8653488
Regularized Movie Bias Model	0.9438521
Regularized Movie + User Bias Model	**0.8648170**

Results

Below is a summary of the models we have built and tested along with the RMSE associated with each model:

Model	**Predicted_RMSE**
Simple/Average	**1.0612018**
Movie Bias Model	**0.9439087**
Movie + User Bias Model	**0.8653488**
Regularized Movie Bias Model	**0.9438521**
Regularized Movie + User Bias Model	**0.8648170**

From this we can see that a simple model of using the average rating of all movies leads to a high RMSE which would lead to inaccurate predictions. However, we can see taking into account the affects of movie and user bias, greatly improve the model. Regularization further improves the model and reduces the RMSE below our target.

Conclusion

In this project we have designed a movie recommendation model to predict a movie rating a user may give. We started by exploring the data and could see variation in the number of ratings per movie and number of ratings submitted per user. We started with a simple model that used the average of all ratings for our predictions. However, this led to a high RMSE and the need to vastly improve the model.

We used our knowledge obtained from the data exploration to introduce a few terms to our model; movie bias and user bias. Taking these into account meant our predictions were improved and as a result RMSE reduced. Lastly, we used regularization to take into account extreme estimates of our movie and user bias. Doing so meant we were able to achieve our goal.

Although we met our target RMSE, the model may still be improved further by looking at other predictors such as genres. We can also look at Matrix Factorization and take into account that groups of movies have similar rating patterns and groups of users have similar rating patterns also. There is also the possibility to use other models such as Knn and Random Forests. All of these could potentially improve the recommendation system and further reduce RMSE.