MovieLens Project Report

Dale Chen-Song

08/06/2020

1. Introduction

Recommendation systems are an important part of many businesses nowadays. As companies collect user informations and ratings, it becomes imperative to the companies to understand and predict the users to make specific recommendations to the users based, so the companies can better sell their products. Therefore, recommendation systems are based on previous user's ratings and uses machine learning to achieve this. The goal of machine learning is to process the data using algorithms to predict.

One big example of this, comes from Netflix. Netflix set out an open challenge, called the Netflix Prize, for the best algorithm to predict user's movie ratings from a large dataset of previous ratings. The winner of the challenge would receive $1M if they could achieve an Root Mean Square Error (RMSE) of 0.8572.

In our report, we create our own recommendation system, similar to the Netflix Prize, and try to achieve the challenge's RMSE target. We divide the dataset so we have a validation set to test our final recommendation system and an edx dataset that we will use to make our recommendation system. We then further divide the edx dataset, so we can use train a set to the test set. We train by looking at bias with movies, then movies with user, and then we finally use regularize our movies and user effect.

2. Summary

While we aren't using the Netflix dataset, the dataset that we use for this project is obtained from MovieLens dataset. While the Netflix challenge uses 100 millions ratings, we use the 10 million version of the MovieLens dataset to make the computation a little easier.

The dataset contains 10 million ratings with 100,000 tag applications, 72,000 users, and 10,000 movie. We split the dataset so it is 90%-10%; the 90% is the edx dataset, which we'll use to analyze and the 10% is used for validation at the end. Both datasets contains 6 categories: the user ID, the movie ID, rating by the user, timestamp when the rating was done, title of the movie, and the genres of the movie.

```
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1                Comedy|Romance
## 2         Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
```

```
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

```
dim(edx)
```

```
## [1] 9000055       6
```

```
n_distinct(edx$userId)
```

```
## [1] 69878
```

```
n_distinct(edx$movieId)
```

```
## [1] 10677
```

```
dim(validation)
```

```
## [1] 999999      6
```

The edx dataset contains around 9,000,000 ratings, with almost 70,000 different users, and more than 10,000 different movies, while the validation dataset has 999,999 ratings.

Here, we show the top most rated movies. The most rated movie is *Pulp Fiction* (1994) with over 31,000 ratings.

```
edx %>% group_by(title) %>%
  summarize(no_ratings = n()) %>%
  arrange(desc(no_ratings))
```

```
## # A tibble: 10,676 x 2
##    title                                                     no_ratings
##    <chr>                                                          <int>
##  1 Pulp Fiction (1994)                                            31362
##  2 Forrest Gump (1994)                                            31079
##  3 Silence of the Lambs, The (1991)                               30382
##  4 Jurassic Park (1993)                                           29360
##  5 Shawshank Redemption, The (1994)                               28015
##  6 Braveheart (1995)                                              26212
##  7 Fugitive, The (1993)                                           25998
##  8 Terminator 2: Judgment Day (1991)                              25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)   25672
## 10 Apollo 13 (1995)                                               24284
## # ... with 10,666 more rows
```

The most popular movie (with more than 1000 ratings) with an average rating of 4.455 is *The Shawshank Redemption* (1994), with over 28,000 ratings.

```
popular<-edx %>% group_by(movieId, title) %>%
  filter(n() >= 1000) %>%
  summarise(count = n(), average = mean(rating)) %>%
  arrange (desc(average))

head(popular)
```

```
## # A tibble: 6 x 4
## # Groups:   movieId [6]
##   movieId title                                count average
##     <dbl> <chr>                                <int>   <dbl>
## 1     318 Shawshank Redemption, The (1994) 28015    4.46
## 2     858 Godfather, The (1972)            17747    4.42
## 3      50 Usual Suspects, The (1995)       21648    4.37
## 4     527 Schindler's List (1993)          23193    4.36
## 5     912 Casablanca (1942)                11232    4.32
## 6     904 Rear Window (1954)                7935    4.32
```

There are over a hundred films with only one rating that may skew the results.

```
edx %>% group_by(title) %>%
  summarize(no_ratings = n()) %>%
  filter(no_ratings == 1) %>%
  count() %>%
  pull()
```

```
## [1] 126
```

Due to the large size of the dataset, we can't use machine learning algorithms from caret as it would use up too many resources. Thus, the method that we use is user and movie effects with regularization.

3. Methods/Analysis

From the edx dataset, we further divide the dataset into 80%-20%, the train and test sets respectively.

The easiest model we can make is by just taking the average and using that to predict against the test set. We find that the average of the training set is 3.512. Using this mean, we can calculate an RMSE value of 1.062.

```
mu<-mean(train_set$rating)
RMSE(test_set$rating, mu)
```

```
## [1] 1.060273
```

As this is an unoptimal RMSE value, we can improve our model by adding an independent error term, the movie bias. With the addition of movie effect, our model's new RMSE value is 0.9441

```
# Add movie bias term
b_i<-train_set %>%
  group_by(movieId) %>%
```

```
  summarize(b_i = mean(rating-mu))

# Predict rating with mean and movie bias term
predicted_ratings<-test_set %>%
  left_join(b_i, by= 'movieId') %>%
  mutate(pred=mu+b_i)%>%
  pull(pred)

# RMSE for movie effect
RMSE(predicted_ratings, test_set$rating)
```

```
## [1] 0.9440821
```

We can further improve on our model by adding an additional bias, users. With this addition, the RMSE value is now 0.8669.

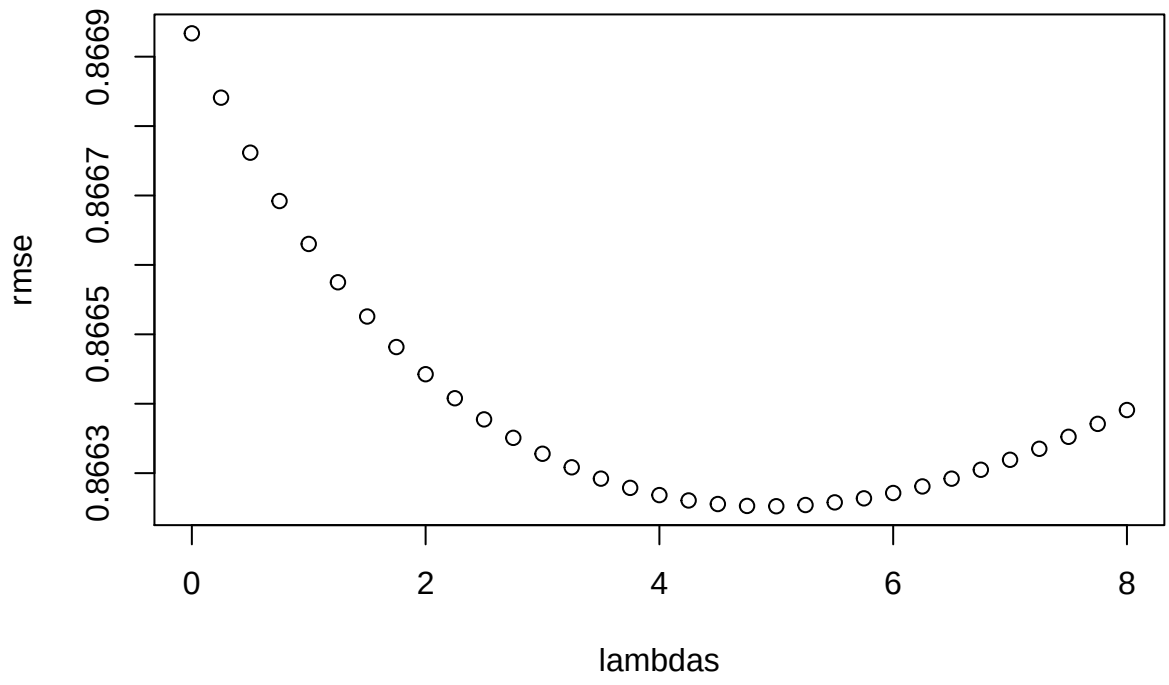
```
# Add user bias term
b_u<- train_set %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

# Predict rating with mean, movie and user bias term
predicted_ratings<- test_set %>%
  left_join(b_i, by= 'movieId') %>%
  left_join(b_u, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

# RMSE for movie and user effect
RMSE(predicted_ratings, test_set$rating)
```

```
## [1] 0.8669336
```

As there are many estimates with small sample sizes as noted in Summary, we use regularization to reduce the effects of large errors in our predictions. We apply regularization on both our biases.

```
# Figuring out best lambda
lambdas<- seq(0,8,0.25)

# Regularization function
regularization<-function(x){
  # Overall average rating of train dataset
  mu<-mean(train_set$rating)

  # Regularized movie bias term
  b_i<- train_set %>%
    group_by(movieId)%>%
    summarize(b_i = sum(rating-mu)/(n()+x))

  # Regularized user bias term
  b_u<- train_set %>%
```

```
    left_join(b_i, by= 'movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i-mu)/(n()+x))

  # Predict rating with mean, movie and user bias term
  predicted_ratings<- test_set %>%
    left_join(b_i, by= 'movieId') %>%
    left_join(b_u, by= 'userId')%>%
    mutate(pred= mu + b_i + b_u) %>%
    .$pred
  # RMSE for regularized movie and user effect
  return(RMSE(predicted_ratings, test_set$rating))}

# Obtain RMSE of lambda using regularization function
rmse<- sapply(lambdas, regularization)

# Plot RMSE vs. lambdas
plot(lambdas, rmse)
```



We see that the best lambda for minimal RMSE is 5.

```
# Figuring out the minimized lambda
lambda<-lambdas[which.min(rmse)]
lambda
```

```
## [1] 5
```

From there, we can see that the RMSE for the minimzed lambda and regularization of movie and user effect is 0.8662.

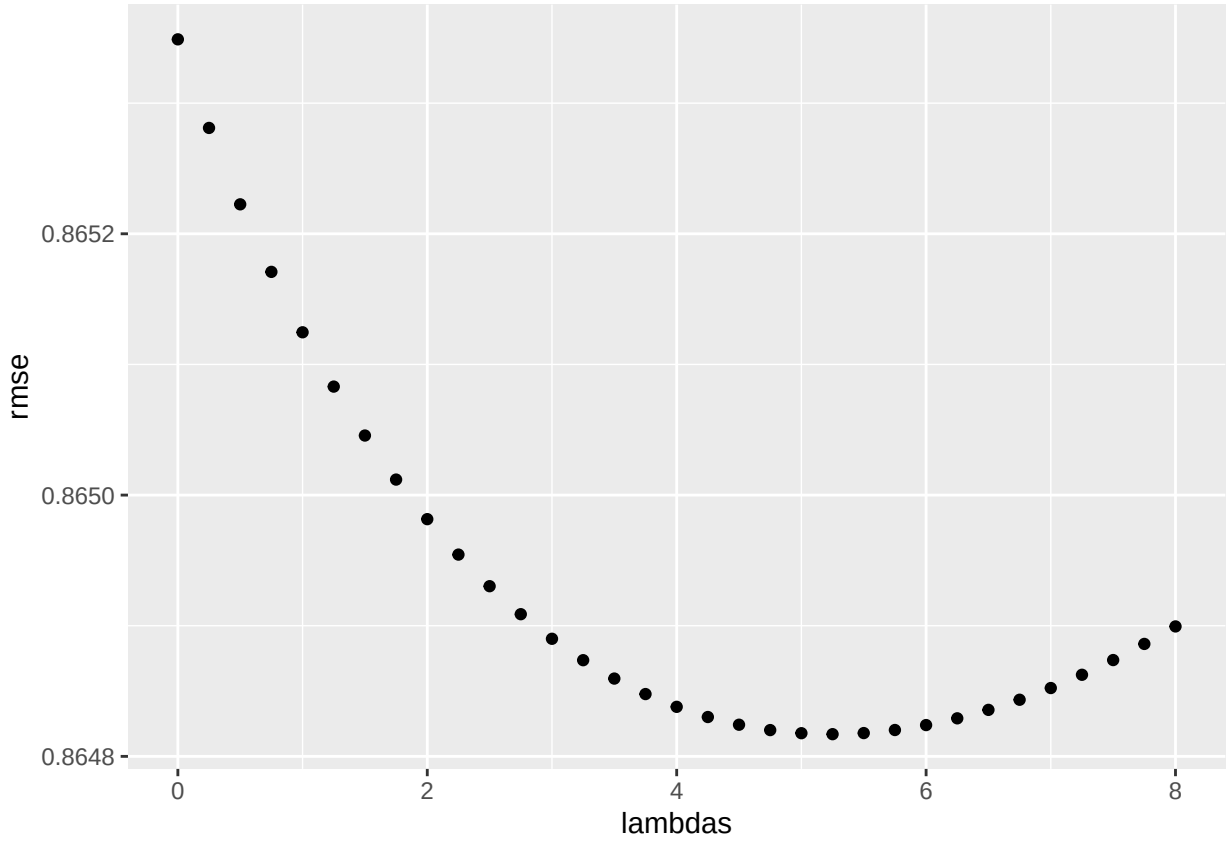
```
# Output RMSE of regularized model
regularization(lambda)
```

```
## [1] 0.8662523
```

4. Results

As we have achieved our final model, we use our model on our validation set to obtain our final RMSE.

```
# Figuring out best lambda
lambdas<- seq(0,8,0.25)

# Regularization function
regularization<-function(x){
  # Overall average rating of edx dataset
  mu<- mean(edx$rating)

  # Regularized movie bias term
  b_i<- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+x))

  # Regularized user bias term
  b_u<- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+x))

  # Predictions on validation set using mean and regularized movie and user bias term
  predicted_ratings<- validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by="userId") %>%
    mutate(pred= mu +b_i+b_u) %>%
    pull(pred)

  # Final model output RMSE
  return(RMSE(predicted_ratings, validation$rating))}

# Obtain RMSE of lambda using regularization function
rmse<- sapply(lambdas, regularization)

#plot
qplot(lambdas,rmse)
```



From then, we find the best lambda to achieve the minimal RMSE is 5.25.

```
# Figuring out the minimized lambda
lambda<-lambdas[which.min(rmse)]
lambda
```

```
## [1] 5.25
```

The RMSE result that we obtain after using the validation set is 0.8648.

```
# Output RMSE of regularized model
regularization(lambda)
```

```
## [1] 0.864817
```

5. Conclusion

While we achieved a reasonable RMSE value of 0.8648, it is not as optimized as it could be. We only looked at movie and user effect, however there are more effects that we could have used such as genre and timestamp.

The winner of the Netflix Prize, BellKor's Pragmatic Chaos, obtained a Test RMSE of 0.8567, showing that there is room for more improvement. BellKor's achieved this level of RMSE by combining multiple approaches. One example is to use timestamp to figure out users' rating, as users rated differently between Fridays vs. Mondays. However, there are limitations that occur while doing this; it would be too much engineering effort for the results that even Netflix doesn't implement this winning algorthim for their recommendation systems.