HarvardX MovieLense Project

Khalil Zarifsadr (Kyle Zarif)

2/26/2020

Overview

Recommender systems are one of the most widespread machine learning applications using by big companies for rating predictions. These systems predict the "rating" a user would give to an item to make specific recommendations. This project generates a movie recommendation system that uses 10M version of the MovieLens dataset. The code to generate the data set is provided by the instructor and is available on the attached Rmd file, and the R script.

A machine learning algorithm is trained using the inputs in one subset to predict movie ratings in the validation set. The edx set is used to develop the algorithm and is split into separate training set and test set. Movie ratings are predicted in the validation set as if they were unknown. Residual mean squared error (RMSE) is used to evaluate how close the predictions are to the true values in the validation set. The goal of this project is to fit the best model that minimizes the RMSE. The validation data is not used for training the algorithm and is only used for evaluating the RMSE of the final model.

Analysis

The first step is to explore the provided data including the dimensions of the edx and validation sets and checking the available variables. We can see that each row represents a rating given by a user to a movie. In this step the number of unique users and unique movies that were rated is also calculated as following:

```
##################
# Data Exploration
##################

# Dimensions of the edx set and test set and checking the variables
dim(edx)

## [1] 9000055       6

dim(validation)

## [1] 999999      6

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
```

```r
head(validation)
```

```
##   userId movieId rating timestamp
## 1      1     231      5 838983392
## 2      1     480      5 838983653
## 3      1     586      5 838984068
## 4      2     151      3 868246450
## 5      2     858      2 868245645
## 6      2    1544      3 868245920
##                                                     title
## 1                                     Dumb & Dumber (1994)
## 2                                     Jurassic Park (1993)
## 3                                        Home Alone (1990)
## 4                                           Rob Roy (1995)
## 5                                    Godfather, The (1972)
## 6 Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
##                                           genres
## 1                                         Comedy
## 2         Action|Adventure|Sci-Fi|Thriller
## 3                                Children|Comedy
## 4                       Action|Drama|Romance|War
## 5                                    Crime|Drama
## 6 Action|Adventure|Horror|Sci-Fi|Thriller
```

```r
# Number of unique users and movies
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId))
```

```
##   n_users n_movies
## 1   69878    10677
```

In This project, movie rating predictions will be compared to the true ratings in the validation set using residual mean square error (RMSE). RMSE is a standard way to measure the typical error of a model in predicting quantitative data. RMSE is a kind of (normalized) distance between the vector of predicted values and the vector of observed values. Typical error of more than 1 means a 1-star error for prediction which is a big error. The following function computes RMSE for vectors of ratings and the corresponding predictors. The edx data set is also partitioned into a test set with 20% of the edx data and a train set with 80% of the edx data. The goal of the project is to develop a final model that minimizes RMSE on the test set. The final model will be evaluated on the validation set.

```
####################################################################
# Defining RMSE and Dividing the edx data into training and test set
####################################################################

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

set.seed(1, sample.kind="Rounding")

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
sampler
## used

test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list =
FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
```

To make sure we didnt include users and movies in the test set that do not appear in the training set we removed these using:

```
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

To implement the first model, we start with a very simple model that considers the same rating regardless of users for all movies. The differences explained by random variability as following: Yu,i = μ + εu,i Where μ is the true rating for all movies and εu,i denotes independent errors from the same distribution which is centered at zero. The optimum estimate to minimize RMSE is the least square estimate of μ as following:

```
##################################################################
# #Building the Recommendation System using average of all ratings
##################################################################

mu_hat <- mean(train_set$rating)
mu_hat

## [1] 3.512482

# Calculating Error using the test set
naive_rmse <- RMSE(test_set$rating, mu_hat)
naive_rmse # In this case 1 star error is big

## [1] 1.059904
```

```
#Any other number makes bigger error
predictions <- rep(3, nrow(test_set))
RMSE(test_set$rating, predictions) # We see RMSE is a bigger star error

## [1] 1.177271

rmse_results <- tibble(method = "Just the average", RMSE = naive_rmse)
rmse_results

## # A tibble: 1 x 2
##   method            RMSE
##   <chr>            <dbl>
## 1 Just the average  1.06
```

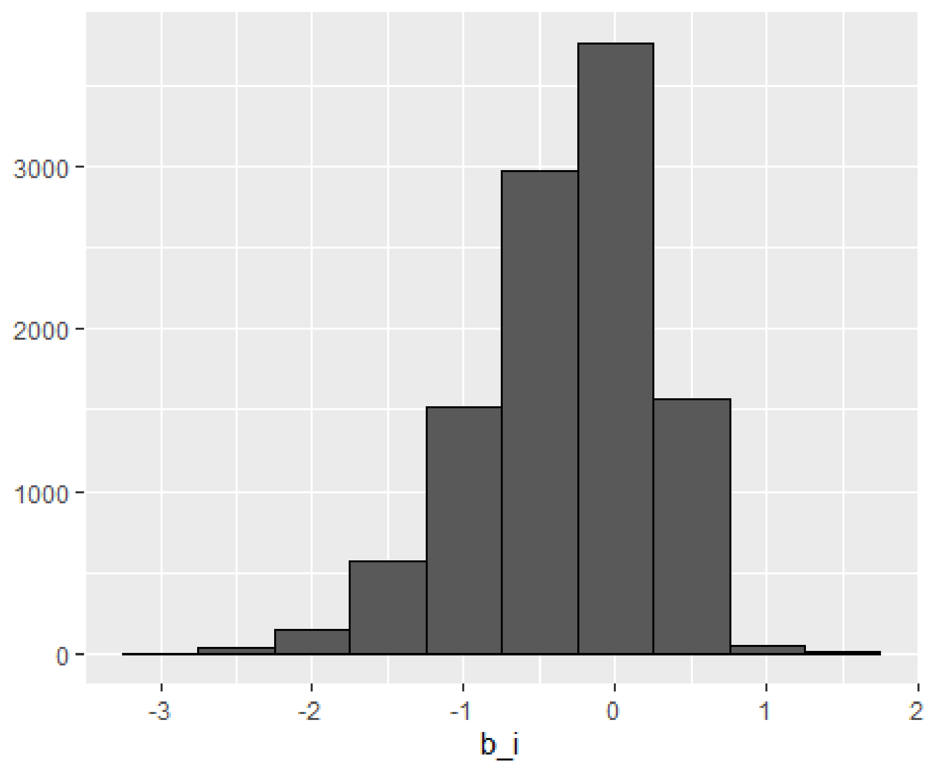
To optimize the first model, we should consider bs as effects. It is intuitive that some movies are generally rated higher compare to others. Adding the term 'bi' to represent average ranking for movie 'i' can improve the first model. 'bi' is the average of Yu,i - μ where μ is the average of the true rating for all movies on the training set. The model is as following:

$$Yu,i = \mu + bi + \varepsilon u,i$$

```
#############################
# Modeling Movie Effects 'bi'
#############################

#fit <- lm(rating ~ as.factor(userId), data = edx) #This code is slow
mu <- mean(train_set$rating)
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

# We know that mu is 3.5 and it means that a 'bi' equal to 1.5 implies a
perfect 5 star movie in the following qplot
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color =
I("black"))
```



```r
predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

model_1_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie Effect Model",
                                 RMSE = model_1_rmse ))

rmse_results %>% knitr::kable()   # we see improvement
```

Method	RMSE
Just the average	1.0599043
Movie Effect Model	0.9437429

It is very helpful to consider user effect as an extra bs to the model. Some users are very though while the others like every movie. Some users give a 4 star rating rather than a 2 to a bad movie. User effect can better predict the rating that kind or cranky users grant to movies by adding the term 'bu' and this model is as following:

$$Yu,i = \mu + bi + bu + \varepsilon u,i$$

```r
##################################################
# Considering user effect by adding the term 'bu'
##################################################
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

predicted_ratings <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

model_2_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie + User Effects Model",
                                 RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()   # we see improvement
```

Method	RMSE
Just the average	1.0599043
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8659320

It is possible to further improve the model by considering regularization. By exploring the data in the following section, we see some movies are rated only one or few times while others are rated thousands of times. Thus, it is very critical to constrain the total variability of effect sizes. When there is a small size, regularization puts a penalty term to large estimates that are formed from small sample sizes.

```
################
# Regularization
################

# 10 commom mistakes we made previously using only movie effects 'bi'
test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  mutate(residual = rating - (mu + b_i)) %>%
  arrange(desc(abs(residual))) %>%
  select(title,  residual) %>% slice(1:10) %>% knitr::kable()
```

Title	residual
From Justin to Kelly (2003)	4.154762
Time Changer (2002)	4.000000
Shawshank Redemption, The (1994)	-3.957318
Shawshank Redemption, The (1994)	-3.957318
Shawshank Redemption, The (1994)	-3.957318
Shawshank Redemption, The (1994)	-3.957318
Shawshank Redemption, The (1994)	-3.957318

Shawshank Redemption, The (1994)	-3.957318
Shawshank Redemption, The (1994)	-3.957318
Children Underground (2000)	-3.928571

```r
# data base that connects movieId to movie title
movie_titles <- movielens %>%
  select(movieId, title) %>%
  distinct()

# 10 best movies based on 'bi'
movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i) %>%
  slice(1:10) %>%
  knitr::kable()
```

Title	b_i
NA	1.487518
NA	1.487518
NA	1.487518
Shadows of Our Forgotten Ancestors (Tini zabutykh predkiv)	1.487518
NA	1.487518
NA	1.487518
NA	1.487518
NA	1.487518
NA	1.487518
More	1.404184

```r
# 10 worst movies based on 'bi'
movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  select(title, b_i) %>%
  slice(1:10) %>%
  knitr::kable()
```

Title	b_i
NA	-3.012482
NA	-3.012482
NA	-3.012482
SuperBabies: Baby Geniuses 2	-2.749982
NA	-2.667244
NA	-2.637482
Disaster Movie	-2.637482
Hip Hop Witch, Da	-2.603391

NA	-2.512482
NA	-2.512482

```
# now lets consider n (number of ratings) and see how often they rated
# 10 Best considering n
train_set %>% dplyr::count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()

## Joining, by = "movieId"
```

Title	b_i	n
NA	1.487518	1
NA	1.487518	3
NA	1.487518	2
Shadows of Our Forgotten Ancestors (Tini zabutykh predkiv)	1.487518	1
NA	1.487518	1
NA	1.487518	1
NA	1.487518	1
NA	1.487518	1
NA	1.487518	1
More	1.404184	6

```
# We see some of them are rated only by one or few users

# 10 worst considering n
train_set %>% dplyr::count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()

## Joining, by = "movieId"
```

Title	b_i	n
NA	-3.012482	1
NA	-3.012482	1
NA	-3.012482	2
SuperBabies: Baby Geniuses 2	-2.749982	40
NA	-2.667244	168

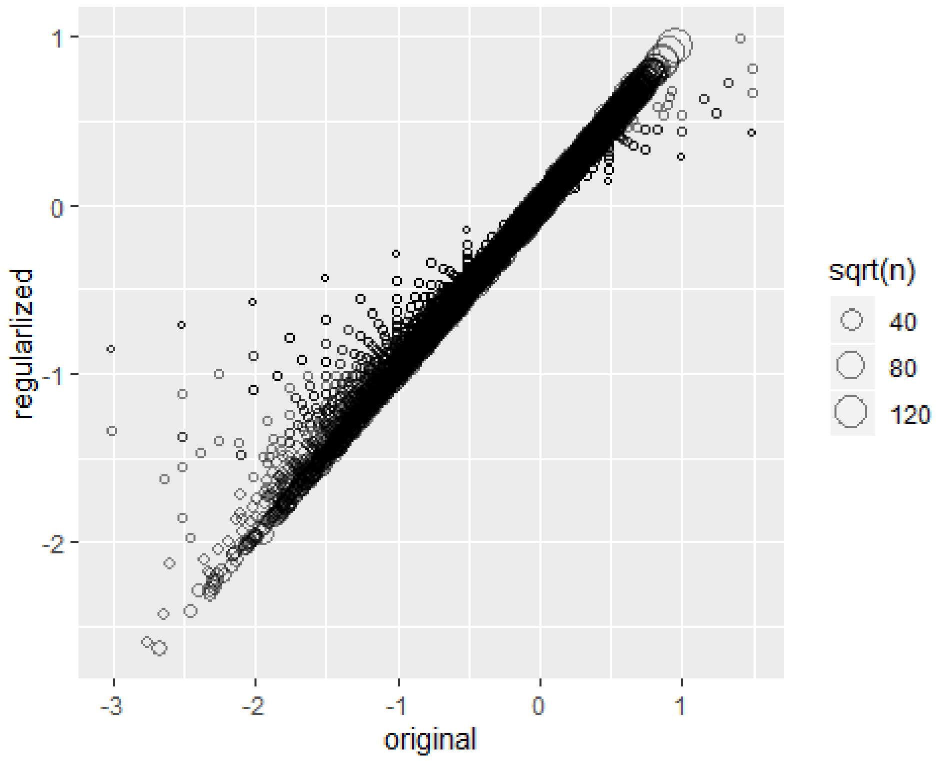
NA	-2.637482	4
Disaster Movie	-2.637482	28
Hip Hop Witch, Da	-2.603391	11
NA	-2.512482	1
NA	-2.512482	1

These are noisy estimates that we shoud not trust. Regulirization starts here to penalize large estimates that come from small sample.

Lambda is a tuning parameter for regularization studies. This parameter controls the total variability of the movie effect by minimizing an equation that adds a penalty term. When the sample size is big the penalty term is ignored and is ignored. For small sample sizes of 'bi', lambda is shrunken to zero. In this section, the plot shows how estimates shrink.

```r
# considering Lambda
lambda <- 2.5
mu <- mean(train_set$rating)
movie_reg_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

# plot of regularazied estmates vs lease square estimate
tibble(original = movie_avgs$b_i,
       regularlized = movie_reg_avgs$b_i,
       n = movie_reg_avgs$n_i) %>%
  ggplot(aes(original, regularlized, size=sqrt(n))) +
  geom_point(shape=1, alpha=0.5)
```



```
# When n is small, then the estimate b_i is shrunken toward zero. The Larger
Lambda, the more we shrink.

# Top 10 movies considering PENALIZED estimates
train_set %>%
  dplyr::count(movieId) %>%
  left_join(movie_reg_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()

## Joining, by = "movieId"
```

Title	b_i	n
More	0.9911891	6
Shawshank Redemption, The	0.9447301	22363
Godfather, The	0.9048453	14107
Usual Suspects, The	0.8568137	17315
Schindler’s List	0.8514631	18567
NA	0.8113734	3
Rear Window	0.8087002	6325
Casablanca	0.8046208	9027

Double Indemnity	0.7972822	1711
Third Man, The	0.7961995	2420

```
# Top Worst considering PENALIZED estimates
train_set %>%
  dplyr::count(movieId) %>%
  left_join(movie_reg_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()

## Joining, by = "movieId"
```

Title	b_i	n
NA	-2.628135	168
SuperBabies: Baby Geniuses 2	-2.588218	40
Disaster Movie	-2.421295	28
NA	-2.406821	109
Glitter	-2.300597	282
Barney's Great Adventure	-2.287959	168
Carnosaur 3: Primal Species	-2.282927	51
Gigli	-2.268819	249
Pokemon 4 Ever (a.k.a. Pokémon 4: The Movie)	-2.249836	168
NA	-2.226802	128

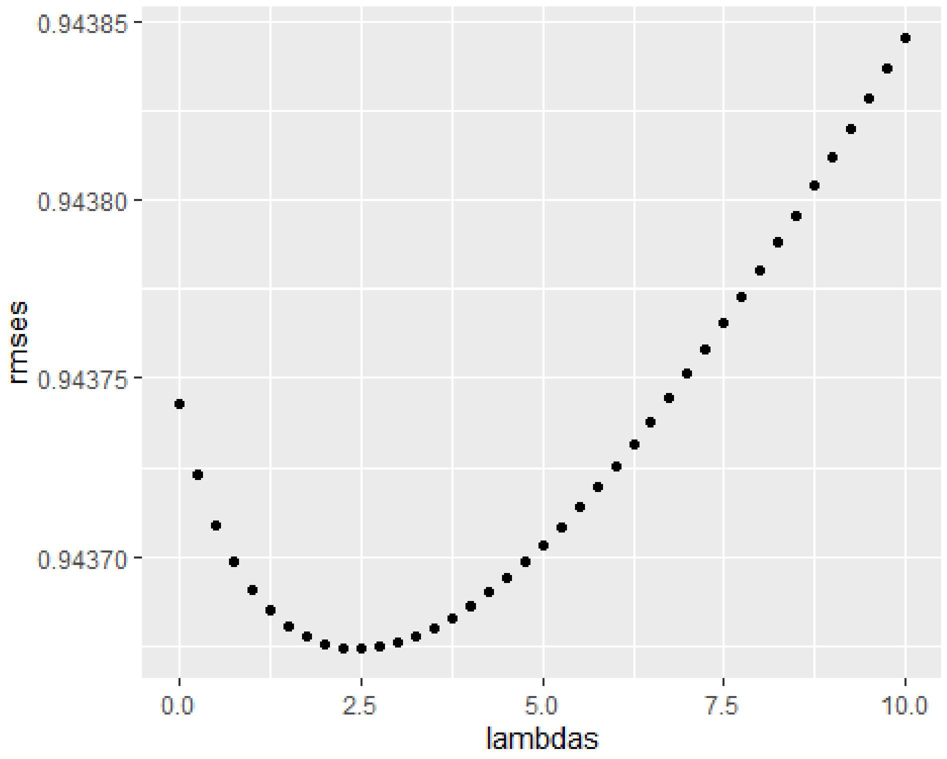
```
# check improvement
predicted_ratings <- test_set %>%
  left_join(movie_reg_avgs, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  .$pred

model_3_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Regularized Movie Effect Model",
                                 RMSE = model_3_rmse ))
rmse_results %>% knitr::kable()
```

Method	RMSE
Just the average	1.0599043
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8659320
Regularized Movie Effect Model	0.9436745

Lambda is a tuning parameter and we can use cross-validation to choose it.

```r
lambdas <- seq(0, 10, 0.25)
mu <- mean(train_set$rating)
just_the_sum <- train_set %>%
  group_by(movieId) %>%
  summarize(s = sum(rating - mu), n_i = n())
rmses <- sapply(lambdas, function(l){
  predicted_ratings <- test_set %>%
    left_join(just_the_sum, by='movieId') %>%
    mutate(b_i = s/(n_i+l)) %>%
    mutate(pred = mu + b_i) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})
qplot(lambdas, rmses)
```

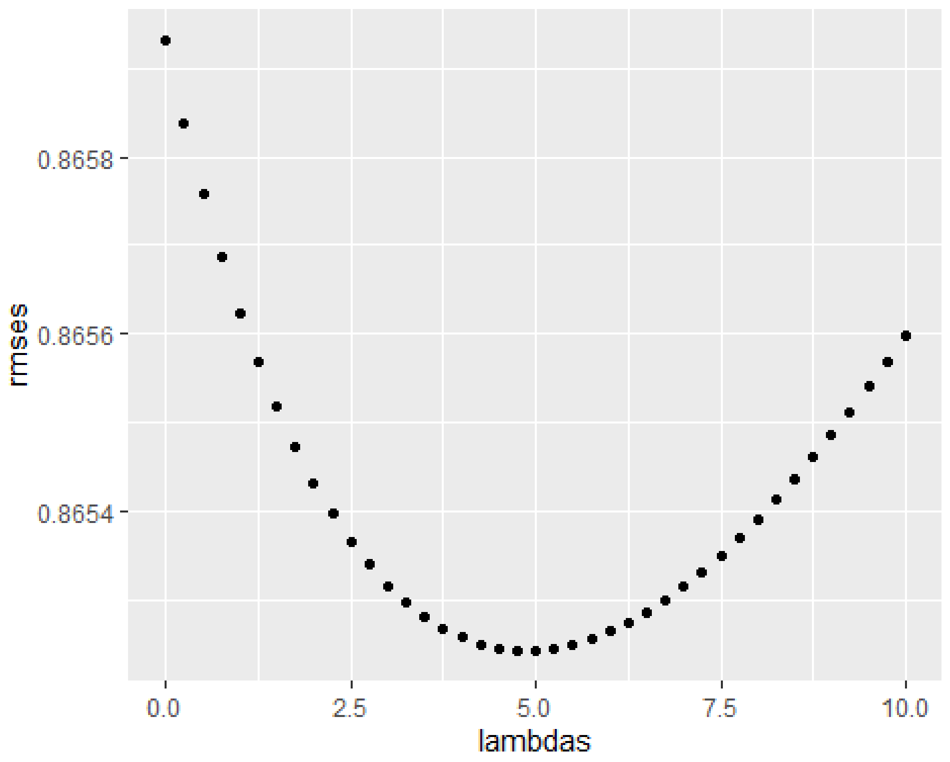


```r
lambdas[which.min(rmses)]
```

```
## [1] 2.5
```

```r
# Regularization considering user effects
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- train_set %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})

qplot(lambdas, rmses)
```



```
# for the full model the optimum Lambda is:
lambda <- lambdas[which.min(rmses)]
lambda

## [1] 4.75

# Comparing RMSEs
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Regularized Movie + User Effect
Model",
                                 RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

Method	RMSE
Just the average	1.0599043
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8659320
Regularized Movie Effect Model	0.9436745
Regularized Movie + User Effect Model	0.8652421

We see that that regularized movie model that considered the User Effect gives a smaller RMSE.

Results

In this section the final movie rating predictions will be compared to the true ratings in the validation set using RMSE. The model that minimized RMSE is the regularized model that considered the movie effect. The model used the tuning parameter lambda equal to 4.75 on the test set. Now for the final step we fit the final model on the validation set.

```
#############################################################################
#
# Fitting Model on the Validation set using the lambda that minimuzed the
RMSE
#############################################################################
#

lambda <- 4.75
mu <- mean(edx$rating)
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))
b_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))
predicted_ratings <-
  validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

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

## [1] 0.8648201
```

Conclusion

The first proposed model of movie recommendation systems considers the same rating using the average of all ratings without considering any biases and the final calculated RMSE was 1.05 which is a big number for this project. By considering movie effect ‘bi’ which is the average of Yu,i - μ for movie ‘i’, the RMSE improved to 0.9437. By adding the user effect ‘bu’ to the model as the second effect RMSE improved to 0.8659. The RMSE hiked to 0.8652 by considering regularization and controlling the variability of the data by adding the tuning parameter Lambda. Lambda equal to 4.75 minimized the RMSE. The final movie prediction was compared to the true rating in the validation set and resulted the RMSE of 0.8648 which is an accepted RMSE for this project. The final model is regularized and is not over-fitted. For future work it is recommended to evaluate and test models by adding additional effects like the released year of movies, genres and other possible effects. Also, matrix factorization and studying the Principal Component Analysis (PCA) can reveal other important patterns that can be used for model optimization.