Report MovieLens Project

Ayman Tuffaha

28/09/2021

About this document (R Markdown Document)

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Introduction

At this report, I have used the MovieLens 10M dataset: https://grouplens.org/datasets/movielens/10m/ which consists of 10 million ratings. 100,000 tag applications applied to 10,000 movies by 72,000 users.

The predictions submitted are scored against the right grades in terms of root mean square error (RMSE), and the goal is to reduce this error value to the minimum based on the different type of biases presented in the movie reviews.

This report contains five sections: About this document (R Markdown Document), Introduction, Build the data and Data exploration, Methods, and Conclusion and results.

The below line of codes, and algorithm written to train a machine learning on using the inputs in one script to predict movie ratings in the validation set.

This will calculate RMSE based on created and predicted movie ratings.

Build the data and Data exploration

```
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-
10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-
10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating",
"timestamp"))
movies <- str_split_fixed(readLines(unzip(dl, "ml-
10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
colnames(movies)
```

```
## [1] "movieId" "title"   "genres"

movies <- as.data.frame(movies) %>% mutate(movieId =
as.numeric(movieId),
                                           title = as.character(title),
                                           genres =
as.character(genres))
movielens <- left_join(ratings, movies, by = "movieId")
movielens

##           userId movieId rating timestamp
##        1:      1     122      5 838985046
##        2:      1     185      5 838983525
##        3:      1     231      5 838983392
##        4:      1     292      5 838983421
##        5:      1     316      5 838983392
##       ---
## 10000050:  71567    2107      1 912580553
## 10000051:  71567    2126      2 912649143
## 10000052:  71567    2294      5 912577968
## 10000053:  71567    2338      2 912578016
## 10000054:  71567    2384      2 912578173
##                                                title
##        1:                                Boomerang (1992)
##        2:                                  Net, The (1995)
##        3:                             Dumb & Dumber (1994)
##        4:                                  Outbreak (1995)
##        5:                                  Stargate (1994)
##       ---
## 10000050:         Halloween H20: 20 Years Later (1998)
## 10000051:                               Snake Eyes (1998)
## 10000052:                                    Antz (1998)
## 10000053: I Still Know What You Did Last Summer (1998)
## 10000054:                  Babe: Pig in the City (1998)
##                                               genres
##        1:                                Comedy|Romance
##        2:                          Action|Crime|Thriller
##        3:                                        Comedy
##        4:                  Action|Drama|Sci-Fi|Thriller
##        5:                       Action|Adventure|Sci-Fi
##       ---
## 10000050:                                Horror|Thriller
## 10000051:                 Action|Crime|Mystery|Thriller
## 10000052: Adventure|Animation|Children|Comedy|Fantasy
## 10000053:                       Horror|Mystery|Thriller
## 10000054:                                Children|Comedy

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

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

test_index <- createDataPartition(y = movielens$rating, times = 1, p =
0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
temp

##          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
##      ---
## 1000003:  71566     235      5 830341062
## 1000004:  71566     273      3 830341118
## 1000005:  71566     434      3 830340953
## 1000006:  71567     480      4 912580688
## 1000007:  71567     898      4 912649403
##                                                    title
##       1:                                Dumb & Dumber (1994)
##       2:                                Jurassic Park (1993)
##       3:                                   Home Alone (1990)
##       4:                                      Rob Roy (1995)
##       5:                                Godfather, The (1972)
##      ---
## 1000003:                                        Ed Wood (1994)
## 1000004: Frankenstein (Mary Shelley's Frankenstein) (1994)
## 1000005:                                  Cliffhanger (1993)
## 1000006:                                Jurassic Park (1993)
## 1000007:                     Philadelphia Story, The (1940)
##                                      genres
##       1:                             Comedy
##       2: Action|Adventure|Sci-Fi|Thriller
##       3:                   Children|Comedy
##       4:         Action|Drama|Romance|War
##       5:                       Crime|Drama
##      ---
## 1000003:                     Comedy|Drama
## 1000004:                     Drama|Horror
## 1000005:        Action|Adventure|Thriller
## 1000006: Action|Adventure|Sci-Fi|Thriller
## 1000007:                   Comedy|Romance

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

```
## # A tibble: 10,668 x 2
##    title
n_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 10,658 more rows
```

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

```
## [1] 118
```

```
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
glimpse(validation)
```

```
## Rows: 999,999
## Columns: 6
## $ userId    <int> 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5,
5, 5, 5, ~
## $ movieId   <dbl> 231, 480, 586, 151, 858, 1544, 590, 4995, 34, 432,
434, 85, ~
## $ rating    <dbl> 5.0, 5.0, 5.0, 3.0, 2.0, 3.0, 3.5, 4.5, 5.0, 3.0,
3.0, 3.0, ~
## $ timestamp <int> 838983392, 838983653, 838984068, 868246450,
868245645, 86824~
## $ title     <chr> "Dumb & Dumber (1994)", "Jurassic Park (1993)",
"Home Alone ~
```

```
## $ genres    <chr> "Comedy", "Action|Adventure|Sci-Fi|Thriller",
"Children|Come~
```

Methods

We will use a linear model due to the huge amount of data and to be applicable to run the code.

The dataset is split 90-10 on train and test sets respectively. This is completed in the first steps of the script. The training set (edx) has 9,000,055 entries with 6 columns. Similarly, the test set (validation) has 999,999 entries and 6 columns. The column information is shown below for the validation dataset.

The simplest model is to use the average across every user and every movie as all of our predicted ratings. This model follows the below simple equation,

where $Y_{u,i}$ is the predicted rating of user u and movie i and μ is the average rating across all entries. This is computed as 3.512 (`mean(edx$rating)`).

```
mu_1 <- mean(edx$rating)
RMSE(validation$rating, mu_1)

## [1] 1.061202
```

In order to improve our model, we add an independent error term $b_{i_u}sers_movies$ that expresses rating differences for users and movies. We will add the user bias term later, but for now we add the movie bias term b_i. This term averages the rankings for any movie i because some are liked or hated more than others. The new model is:

```
# add movie bias term, b_i_users_movies
b_i_users_movies <- edx %>%
  group_by(movieId) %>%
  summarize(b_i_users_movies = mean(rating - mu_1))
# predict all unknown ratings with mu and b_i_users_movies
predicted_ratings <- validation %>%
  left_join(b_i_users_movies, by='movieId') %>%
  mutate(pred = mu_1 + b_i_users_movies) %>%
  pull(pred)
# calculate RMSE of movie ranking effect
RMSE(validation$rating, predicted_ratings)

## [1] 0.9439087
```

Now we introduce the user bias term b_u in order to further improve our model. This term minimizes the effect of extreme ratings made by users that love or hate every

movie. Each user u is given a bias term that sways their predicted movies. Our updated model is:

```
# add user bias term, b_user
b_user <- edx %>%
  left_join(b_i_users_movies, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_user = mean(rating - mu_1 - b_i_users_movies))
# predict new ratings with movie and user bias
predicted_ratings <- validation %>%
  left_join(b_i_users_movies, by='movieId') %>%
  left_join(b_user, by='userId') %>%
  mutate(pred = mu_1 + b_i_users_movies + b_user) %>%
  pull(pred)
# calculate RMSE of movie and user bias effect
RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.8653488
```

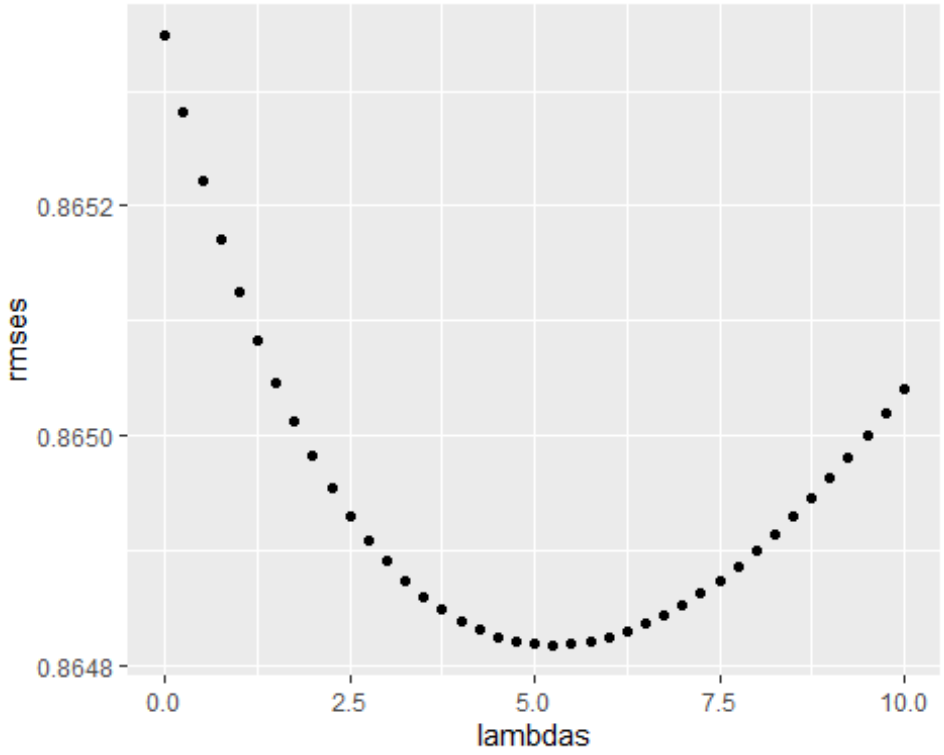
last but not least, we use regularization to minimize the effect of high errors in our predictions.

Regularization penalizes incorrect estimates on small sample sizes. For instance, our b_i term accounts for the average deviation on all ratings of a movie, whether there is 1 or 100 ratings to the movie. We use regularization to reduce the dramatic effect that a exceptionally extreme rating will have on our b_i term. This method is also applied to the user bias term b_u to reduce large anomalies in the ratings of users.

Regularization achieves the same goal as confidence intervals when you are only able to predict a single number, not an interval. Our new model is:

where the first term is our previous least squares equation and the last term is the penalty with large bias terms. Minimizing the biases using a single λ is the goal to our model shown above. We test `lamda <- seq(from=0, to=10, by=0.25)` and plot the results below:

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




```
min(rmses)
```

```
## [1] 0.8648175
```

We see that the minimizing λ term is

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

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

Conclusion and results

Final model with regularized movie and user effects

```
# choose minimized lambda
lam_1 <- lambdas[which.min(rmses)]
# compute regularize movie bias term
b_i_users_movies <- edx %>%
  group_by(movieId) %>%
  summarize(b_i_users_movies = sum(rating - mu_1)/(n()+lam_1))
# compute regularize user bias term
b_user <- edx %>%
  left_join(b_i_users_movies, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_user = sum(rating - b_i_users_movies - mu_1)/(n()+lam_1))
# compute predictions on validation set based on these above terms
predicted_ratings <- validation %>%
  left_join(b_i_users_movies, by = "movieId") %>%
```

```
  left_join(b_user, by = "userId") %>%
  mutate(pred = mu_1 + b_i_users_movies + b_user) %>%
  pull(pred)
# output RMSE of our final model
RMSE(predicted_ratings, validation$rating)

## [1] 0.8648175
```

We notice here an incremental improvements to the RMSE as we supplant our model with bias terms and regularization..

Method	RMSE
Average	1.06120
Movie effect	0.94391
Movie and user effects	0.86535
Regularized movie and user effect	**0.86481**

This model will give us better utilization of computing processing.