Movie Lens Report

Ravi Singh (RS)

09/29/2020

Introduction

Data Science and Analysis is the future. Programming Languages like R are the backbone and Machine Learning (ML) Techniques is how the world will evolve in this highly-technological modern world. It would be used in the food we select, the TV shows we watch and also the Cars we drive. It can work according to our Bias or against it, in a business world. The main aim of ML Techniques is to analyse and process data into a useful and meaning form by providing intuitive solutions. Effective machine learning algorithms have attracted a lot of attention in recent years: for example, Oil and Gas companies, Tech companies and also Netflix and Amazon, they all have invested heavily into this powerhouse.

Building off the Hulu challenge premise and, more specifically, predicting movie ratings for customers in a dataset, is the taken up problem. Training a linear regression model to predict movie ratings and calculate the Root Mean Square Error (RMSE) of the predicted versus actual ratings, is the aim of the analysis.

In this study, there are four major parts: Firstly, the problem is presented in the introduction section, the summary describes the large dataset and generates preliminary inquiries, the model is established in the methods section and implements in with the accompanying .R file, and finally the results are present is the conclusion section.

Summary

Using the given Data Set MovieLens 10M dataset which comprises of 100,000 applications (tag) applied to 10,000 movies by 72,000 users along with 10 million ratings. Leading into a varying numbers of ratings for movies, with the most rated movie *Pulp Fiction* (1994) comprising over 31,000 ratings and about 100 titles from a single rating.

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

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## Warning: `...` is not empty.
##
## We detected these problematic arguments:
## * `needs_dots`
##
```

```
## These dots only exist to allow future extensions and should be empty.
## Did you misspecify an argument?

## # A tibble: 1 x 2
##   title n_ratings
##   <chr>     <int>
## 1 <NA>    9000061

# Movie count Rated a single time
edx %>% group_by(title) %>%
  summarize(n_ratings = n()) %>%
  filter(n_ratings==1) %>%
  count() %>% pull()

## `summarise()` ungrouping output (override with `.groups` argument)

## [1] 0
```

On observing the size of the dataset, a built-in ML algorithms using R packages example *caret* would require abundant simulation power in a desktop for running all the data in a timely manner. Thus, the goal is achieved using a linear regression model developed within the section named method. The RMSE function is used utilizing the DescTools package as the measure of accuracy.

The dataset is split, it is split between train and test sets respectively (90-10). Completed in the first steps of the script. The training sets (edx) comprises of 9000055 entries and 6 columns. Similarly, the test set (validation) has 999999 entries with 6 columns. The column information is generated and shown in the validation dataset.

```
glimpse(validation)

## Rows: 999,993
## Columns: 6
## $ userId    <int> 1, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6,
...
## $ movieId   <dbl> 588, 1210, 1544, 151, 1288, 5299, 380, 435, 480, 477,
508...
## $ rating    <dbl> 5.0, 4.0, 3.0, 4.5, 3.0, 3.0, 3.0, 3.0, 5.0, 3.0, 3.0,
4....
## $ timestamp <int> 838983339, 868245644, 868245920, 1133571026, 1133571035,
...
## $ title     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
N...
## $ genres    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
N...
```

Methods

The average across each user and each movie is the method used in all the predicted ratings. This model is in the form of a simple equation.

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

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

## [1] 1.060651
```

To improve the ML model, addition of an error term $M_{u,i}$ could be useful as it expresses the differences in ratings between users and movies. The addition of user bias term will be done later, but for now we add the movie bias term M_i. The term averages the rankings for any movie i because some are liked or hated more than others. The new model is:

```
# Adding movie bias term, M_i
M_i <- edx %>%
  group_by(movieId) %>%
  summarize(M_i = mean(rating - mu))

## `summarise()` ungrouping output (override with `.groups` argument)

# Unknown Rating Prediction using mu and M_i
predicted_ratings <- validation %>%
  left_join(M_i, by='movieId') %>%
  mutate(pred = mu + M_i) %>%
  pull(pred)
# calculating RMSE of movie ranking effect
RMSE(validation$rating, predicted_ratings)

## [1] 0.9437046
```

The user bias term M_u in added to further improve the regression model. The term reduces the effect of impactful ratings made by users that extremely like or dislike every movie. Each user u is provided with a bias term that sways the predicted movies. Thus the updated model is given as follows:

```
# add user bias term, b_u
M_u <- edx %>%
  left_join(M_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(M_u = mean(rating - mu - M_i))

## `summarise()` ungrouping output (override with `.groups` argument)

# predict new ratings with movie and user bias
predicted_ratings <- validation %>%
  left_join(M_i, by='movieId') %>%
```

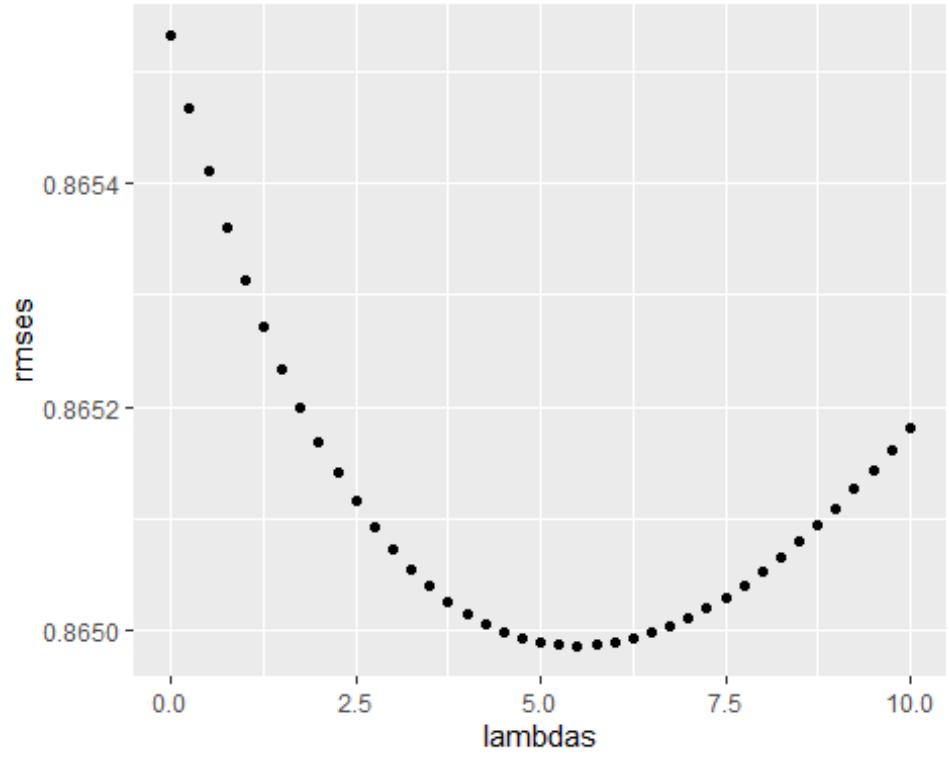
```
  left_join(M_u, by='userId') %>%
  mutate(pred = mu + M_i + M_u) %>%
  pull(pred)
# calculate RMSE of movie and user bias effect
RMSE(predicted_ratings, validation$rating)

## [1] 0.8655329
```

Lastly, Regularization are employed so as to reduce the effect of large errors for our predictions. Regularization incorrect estimates on small dataset sample sizes. Upon looking at the M_i term it accounts for the average deviation on all ratings of a movie, whether there is 1 or 100 ratings within the movie. We use regularization to reduce the dramatic effect that are exceptionally extreme rating affecting the M_i term. Similarly this procedure is applied on the user bias term M_u in order to reduce large anomalies towards the ratings of users.

Confidence interval and Regualtions achieve the same goals when predicting a single number and not an interval. The model is given as:

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

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



We see that the minimizing λ term is

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

```
## [1] 5.5
```

Results

For completeness, the final model is executed below:

```
# minimized lambda is chosen
lam <- lambdas[which.min(rmses)]
# the movie bias term is compute and regularize
M_i <- edx %>%
  group_by(movieId) %>%
  summarize(M_i = sum(rating - mu)/(n()+lam))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# generate & regularize user bias term
M_u <- edx %>%
  left_join(M_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(M_u = sum(rating - M_i - mu)/(n()+lam))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# generate the predictions upon validation set based on these above terms
predicted_ratings <- validation %>%
  left_join(M_i, by = "movieId") %>%
  left_join(M_u, by = "userId") %>%
  mutate(pred = mu + M_i + M_u) %>%
  pull(pred)
# output RMSE of our final model
RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.8649857
```

Incremental improvements are observed to the RMSE upon supplementing 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

The model developed is significant and more effective than the machine learning algorithms of R programming language