HarvardX Data Science Capstone: MovieLens Project

Medea

5/3/2021

Contents

1 Introduction

In our modern era business success often depends if we are able to recommend correct product to the end-user. The good example of this is streaming services like Netflix. To keep users engaged we must provide them relative content.

In this project we pursued two simple objectives:

1. Try to analyze what were the key factors that contributed to the high or low rating for the specific user, which is the the heart of movie recommendation
2. Make RMSE < 0.86490 (We have built our movie rating predictation model by expanded what we have learned in Prof. Rafael A Irizarry's class about Netflix challenge, Assumptions here reader is familiar terms like RMSE & regularization, so we won't need to redefine them. We would highly recommend to check his "Introduction to Data Science".)

Data we used in the analysis comes from GroupLens, 10M version, which makes our work simpler but is sufficient to draw vital conclusions. We performed data wrangling: split data for training/testing/validation reasons, compare different biases and achieve our goals.

2 Analysis

After download data we have reviewed columns of the movielens, we decided to extract year from timestamp as separate column, release year may have effect rating, old movies sometimes seems naive, and in sci-fi effects seems to be outdated. For analysis data was split into edx (90%) & validation(10%) sets . Former for training and latter for the final validation.

We observed columns/dimensions and distinct users, movies, genres, year in the edx/validation dataset

```r
colnames(edx)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
## [7] "year"
```

```r
dim(edx)
```

```
## [1] 9000055       7
```

```r
dim(validation)
```

```
## [1] 999999      7
```

```r
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId),
            n_genres=n_distinct(genres),
            n_year=n_distinct(year),
            )
```

```
##   n_users n_movies n_genres n_year
## 1   69878    10677      797     15
```

As we see not all users rated all movies, this may related to genre preferences, movies quality in users eyes, date movie released. We needed to experiment with data, to determine different biases. So questions we've tried to answer:

1. What RMSE we will get if we use naive approach, e.i. if we follow our intuition and give movies rating as average of whole ratings
2. What if we only consider movie effect?
3. What if we only consider user effect?
4. What if we only consider year effect?
5. What if we only consider genres effect?
6. What if we use different combination of those?
7. Should we consider using of regularization?

To create reliable model we divide edx dataset into the train(90%) and test(10%) sets. We also make sure userId and movieId in test_set are also in train set

First let's develop simple model and predict the same rating for all movies regardless of user. Here all the differences explained by random variation and model would look like this:

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

Which is translated into following code:

```r
#Naive rmse model
mu <- mean(train_set$rating)
naive_rmse <- RMSE(test_set$rating, mu)
naive_rmse
```

```
## [1] 1.061135
```

```
rmse_results <- tibble(method = "Average", RMSE = naive_rmse)
```

Once we have a starting line model we can start to augment it. For example let's consider year effect only. It can be depicted using following formula:

$$Y_{u,y} = \mu + b_y + \varepsilon_{u,y}$$

where term b_y to represent average ranking for year so $\hat{b_y}$ is mean of $Y_{u,y} - \hat{\mu}$

```
year_avgs <- train_set %>%
  group_by(year) %>%
  summarize(b_y = mean(rating - mu))

predicted_ratings <- test_set %>%
  left_join(year_avgs, by='year') %>%
  mutate(pred = mu + b_y) %>%
  pull(pred)

y_rmse <-RMSE(predicted_ratings, test_set$rating)
rmse_results <- rmse_results %>% add_row(method = "Year", RMSE = y_rmse)
y_rmse
```

```
## [1] 1.059381
```

The result we get is better than naive one. Although we've proceed with checking other possible biases, but for simplicity reasons here we are skip the discussion of the most scenarios and focus on the three: Movie/User, Movie/User/Year/Genres and Movie/User/Year/Genres/Regularization combinations.

If we take both user and movie in account model can be described by formula

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

where term b_u is user specific effect. so $\hat{b_u}$ is mean of $Y_{u,i} - \hat{\mu} - \hat{b_i}$ We can use below code to calculate result

```
#Predict using user & movie effect
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) %>%
  pull(pred)

i_u_rmse <-RMSE(predicted_ratings, test_set$rating)
rmse_results <- rmse_results %>% add_row(method = "Movie/User", RMSE = i_u_rmse)
i_u_rmse
```

```
## [1] 0.8659736
```

The prediction is improved again, but we still have room for improvement.

Following the similar logic as in "Movie/User" model "Movie/User/Year/Genres" combination can be expressed using following formula:

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

```
#Predict using user & movie & year & genres effect
genres_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(year_avgs, by='year') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u - b_y))

predicted_ratings <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(year_avgs, by='year') %>%
  left_join(genres_avgs, by='genres') %>%
  mutate(pred = mu + b_i + b_u + b_y + b_g) %>%
  pull(pred)

i_u_y_g_rmse<-RMSE(predicted_ratings, test_set$rating)
rmse_results <- rmse_results %>% add_row(method = "Movie/User/Year/Genres",
                                          RMSE = i_u_y_g_rmse)
i_u_y_g_rmse
```

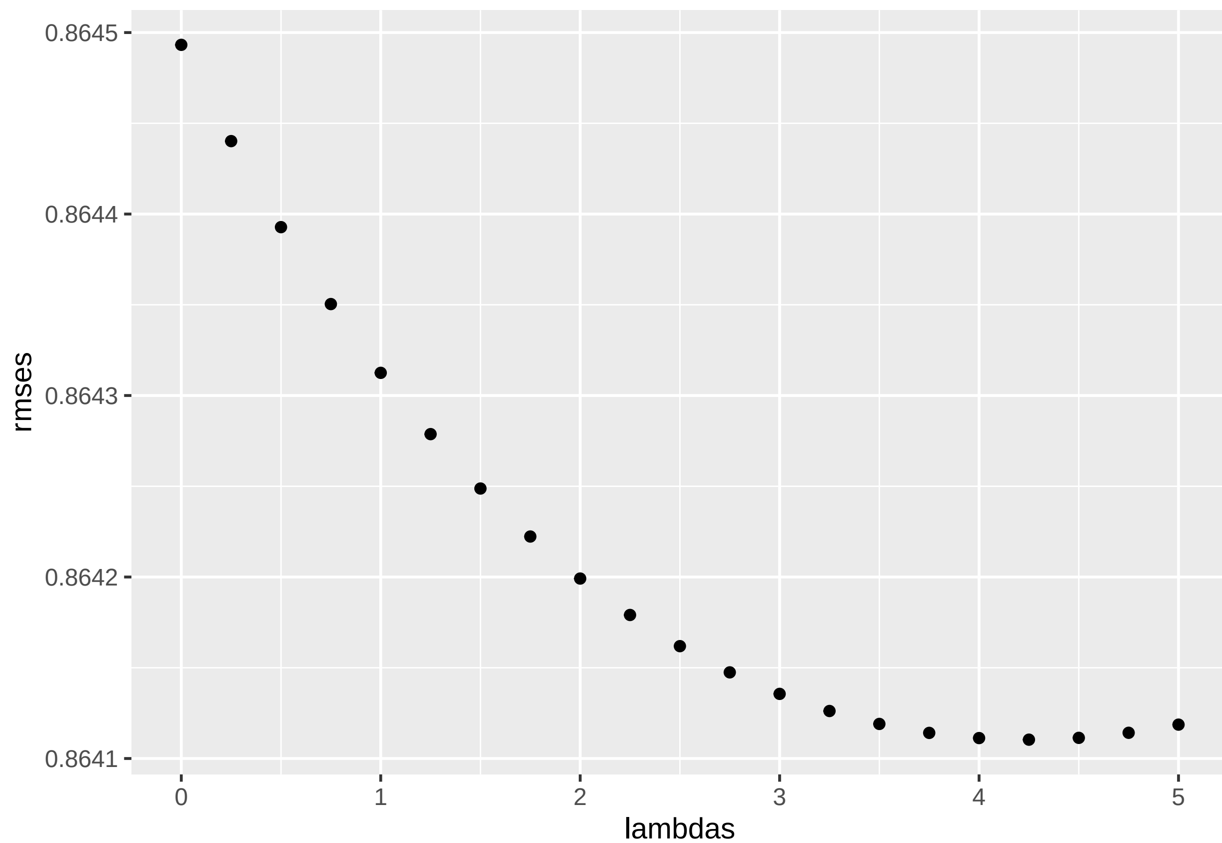
```
## [1] 0.865588
```

Result is the best so far but we still have not reached our second objective. e.i. RMSE < 0.86490. From the edx dataset is easy to see that some movies are receiving only a handful reviews and their rating could mess up with recommendation. To mitigate this effect regularization should be performed and suitable lambda which minimizes RMSE be chosen.

```
mu <- mean(train_set$rating)
lambdas <- seq(0, 5, 0.25)
rmses <- sapply(lambdas, function(lambda) {

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu) / (n() + lambda))


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

  b_y <- train_set %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    group_by(year) %>%
    summarize(b_y = mean(rating - mu - b_i  - b_u))
```

```r
  b_g <- train_set %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_y, by='year') %>%
    group_by(genres) %>%
    summarize(b_g = mean(rating - mu - b_i - b_u - b_y))

  # Compute the predicted ratings on test_set dataset

  predicted_ratings <- test_set %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_y, by='year') %>%
    left_join(b_g, by='genres') %>%
    mutate(pred = mu + b_i + b_u + b_y + b_g) %>%
    pull(pred)

  # Predict the RMSE on the validation set

  return (RMSE(predicted_ratings,test_set$rating))
})
min_lambda <- lambdas[which.min(rmses)]
i_u_y_g_cv_rmse<-min(rmses)
rmse_results <- rmse_results %>% add_row(method = "Movie/User/Year/Genres cross-validation",
                                         RMSE = i_u_y_g_cv_rmse)
```

Let's plot our lambdas to observe dynamic



3 Results

After observing results we had from our training set we conclude that the most valuable bias combination is : user/movie/year/genre in combination with data regularization, the most insignificant effect is Year, but we still decide to keep it. Lambda value used:

```
min_lambda <- lambdas[which.min(rmses)]
```

The results on train set

```
rmse_results %>% knitr::kable()
```

method	RMSE
Average	1.0611350
Year	1.0593807
Movie	0.9441568
User	0.9795916
Year	1.0593807
Movie/User	0.8659736
Movie/User/Year	0.8659598
Movie/User/Genres	0.8656018
Movie/User/Year/Genres	0.8655880
Movie/User/Year/Genres cross-validation	0.8641104

Once we tried different combinations of biases and created our final model we test it against validation set.

```
## [1] 0.864426
```

So we have achieved our second objective, although it is a little bit worse than what we get in the training data, but still in the range we need.

4 Conclusion

Neither of bias alone would have been sufficient enough to fulfill our requirements. Movie and User, Genres effects play important roles, year bias not that important and regularization helped us achive our ultimate goal for RMSE.

Next step in analyzes is to split "genres" column into separate columns and see if some genres combination contributes rating more than others and or it is more beneficial treat them as whole. What about time affect. Also in the current global pandemic situation it will be interesting to see if pandemic year affected movie ratings more.