MovieLens report

Wenyan Ge

June 3, 2019

Overview

- **About the goal** The goal of this report is to create a **movie recommendation system (or a prediction model)** using the MovieLens 10M Dataset above. We use RMSE to evaluate the model we built. We should try to minimize the RMSE of our model to create a better model.
- **About the data set** We use MovieLens 10M movie ratings as the data set. It's a stable benchmark dataset. 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. Released 1/2009.
- **About the prediction models** We used 4 methods to build the prediction models. They are:

1. Predict by all movies' average model
2. Predict by all movies' by_group average model
3. Movie effects model
4. Movie effects + User effects model

Create Test and Validation Sets (Data wrangling)

The code following is provided by the course instruction to create test and validation sets of the MovieLens data set. The training set is named as "edx", and the test set is named as "validation".

```
# Create Test and Validation Sets

# Install and load the required libraries
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.1.1       v purrr   0.3.2
## v tibble  2.1.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

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

# Unzip "ratings.dat" and add column names then saves it as "ratings"
ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                      col.names = c("userId", "movieId", "rating", "timestamp"))

# Unzip "movies.dat" and add column names then saves it as "movies"
movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

# Modify "movies" as a data.frame change columns' variables into proper class
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

# Join "ratings" and "movies" into one file
movielens <- left_join(ratings, movies, by = "movieId")



# Validation set will be 10% of MovieLens data
set.seed(1) # if using R 3.6.0: set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
     semi_join(edx, by = "movieId") %>%
     semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```r
edx <- rbind(edx, removed)

# Remove temp variables
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Basic Data Exploration of the Training Set (edx)

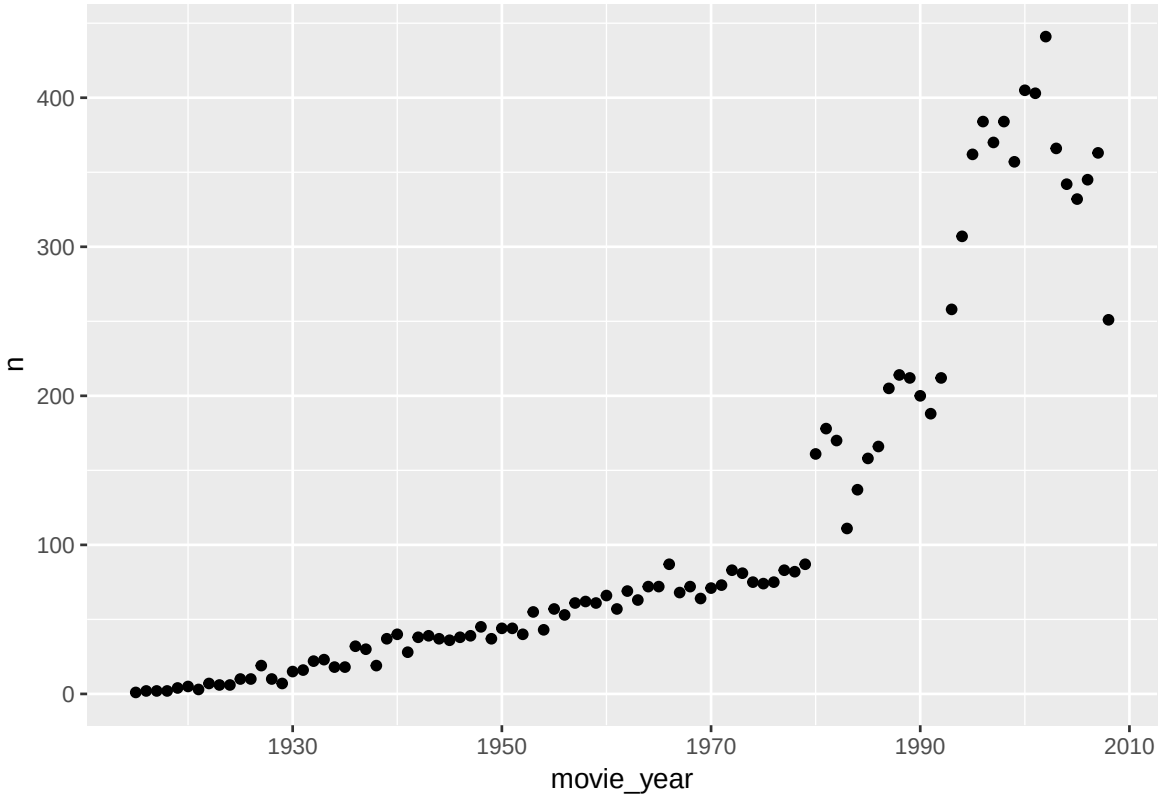
- 10677 unique movies and 69878 unique users `n_distinct(edx$movieId)` `n_distinct(edx$userId)`
- Most rated movies are

```
edx %>% group_by(movieId, title) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                  count
##      <dbl> <chr>                                                  <int>
##  1     296 Pulp Fiction (1994)                                    31362
##  2     356 Forrest Gump (1994)                                    31079
##  3     593 Silence of the Lambs, The (1991)                       30382
##  4     480 Jurassic Park (1993)                                   29360
##  5     318 Shawshank Redemption, The (1994)                       28015
##  6     110 Braveheart (1995)                                      26212
##  7     457 Fugitive, The (1993)                                   25998
##  8     589 Terminator 2: Judgment Day (1991)                      25984
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (19~ 25672
## 10     150 Apollo 13 (1995)                                       24284
## # ... with 10,667 more rows
```

- movie release year distribution

```
library(stringr)
# Extract movie year from title as a new column named movie year
movie_years <- edx %>% mutate(movie_year = as.numeric(gsub("*.+\\((\\d{4})\\)$","\\1", edx$title)))
  group_by(movieId) %>%
  summarise(title = first(title), movie_year = first(movie_year))

# Count movies number by movie year
movie_years_summary <- movie_years %>%
  group_by(movie_year) %>%
  summarise(n = n()) %>%
  arrange(desc(n))

# Plot movie year against a number
movie_years_summary %>%
  ggplot(aes(movie_year,n)) +
  geom_point()
```



Before 1980, movie numbers are under 100/year. After about 1995, movie numbers increased significantly to about 350 ~ 400/year.

Modeling approach

Model 1: Predict by all movies' average model

Our first prediction model simply uses the overall average of the training data. RMSE of the model is about **1.06**, which performance is not good.

Here is the code:

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

#predict by all movies' average

#average rating of all movies
mu_hat_1 <- mean(edx$rating)
rmse_1 <- RMSE(validation$rating, mu_hat_1)

#rmse_1 is about 1.061202
rmse_1
```

```
## [1] 1.061202
```

```
#store the result
rmse_results <- tibble(method = "Just the average", RMSE = rmse_1)
```

Model 2: Predict by all movies' by_group average model

Like Model 1, we tried another simple average model. In this model, we first grouped movies by movieId, then calculate the average. As a result, RMSE is **1.10**, even worse than model 1.

We assume to use the grouped movies average, we will do better than model 1. Because this model grouped movies, which give a movie an average rating. Then we use the average of movies to predict an unknown movie. Model 2 makes more sense than model 1, which just calculating the overall movies' average.

However, in fact, as model 1 calculated overall movies' average, we have to test overall movies' ratings. thus model 1 will do better than model 2.

Here is the code:

```
#predict by all movies' by_group average
mu_hat_2 <- edx %>% group_by(movieId) %>% summarize(mean_by_group = mean(rating)) %>% summarize(m = mea
rmse_2 <- RMSE(validation$rating, mu_hat_2)

#rmse_2 is about 1.108485 which is worse than rmse_1
#I take mu_hat_2 as the TRUE rating average of all movies, but its performance is worse
#because we have to predict NOT once per movie, but many times per movie. So the rmse_1 works better.
rmse_2
```

```
## [1] 1.108485
```

```
#store the result
rmse_results <- bind_rows(rmse_results,
                          tibble(method = "Grouped average",
                                 RMSE = rmse_2))
```

model 3: Movie effects model

As we knew, there are some popular movies that almost everyone like, we can definitely improve our RMSE by predicting these popular movies higher ratings. The same is true to "bad movies", which have a generally bad reputation. This model calculates how good a movie usually is as **Movie effect**.

We can use this equation, which $Y_{u,i}$ is the outcome (rating) of movie $\boldsymbol{i}$ rated by user $\boldsymbol{u}$, and μ is the overall movies' average, b_i for Movie effect of movie i :

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

We can see our RMSE is quite improved, is now **0.94**. Here is the code to calculate the Movie effects model:

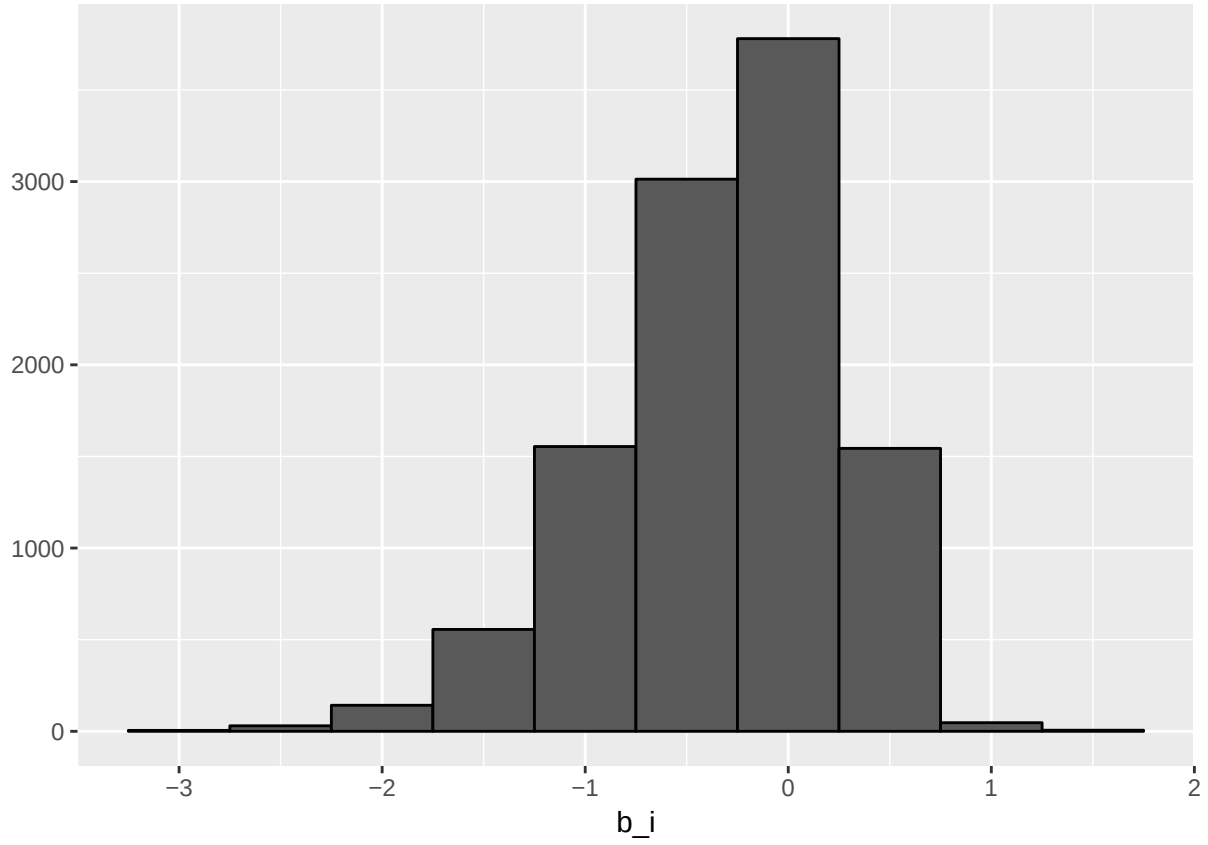
```
#Movie effects
#Since some movies are generally interesting and getting a higher rating, we should put this into the m

#calculate the overall movie rating average
mu_3 <- mean(edx$rating)

#calculate the movie effect as b_i
movie_avgs_3 <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu_3))

#Use b_i to calculate the predicted rating
```

```
predicted_ratings_3 <- validation %>%
  select(movieId) %>%
  left_join(movie_avgs_3, by='movieId') %>%
  mutate(predict_rating = b_i + mu_3) %>%
  pull(predict_rating)

#calculate the rmse
rmse_3 <- RMSE(predicted_ratings_3, validation$rating)

#store rmse into results
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie Effect Model",
                                     RMSE = rmse_3))
rmse_results
```

```
## # A tibble: 3 x 2
##   method              RMSE
##   <chr>              <dbl>
## 1 Just the average   1.06
## 2 Grouped average    1.11
## 3 Movie Effect Model 0.944
```

We can use this peice of code to get an idea of movie effect. A score of 0 is an overall average movie, left side shows ratings below average and right side shows ratings above.

```
movie_avgs_3 %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"))
```



model 4: Movie effects + User effects model

Similar to Movie effects, some users tend to give critical ratings, while some users give all movies 5, the highest rating. We capture this effect as the **user effects**(b_u), which can be used to improve the prediction.

Now our model becomes

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

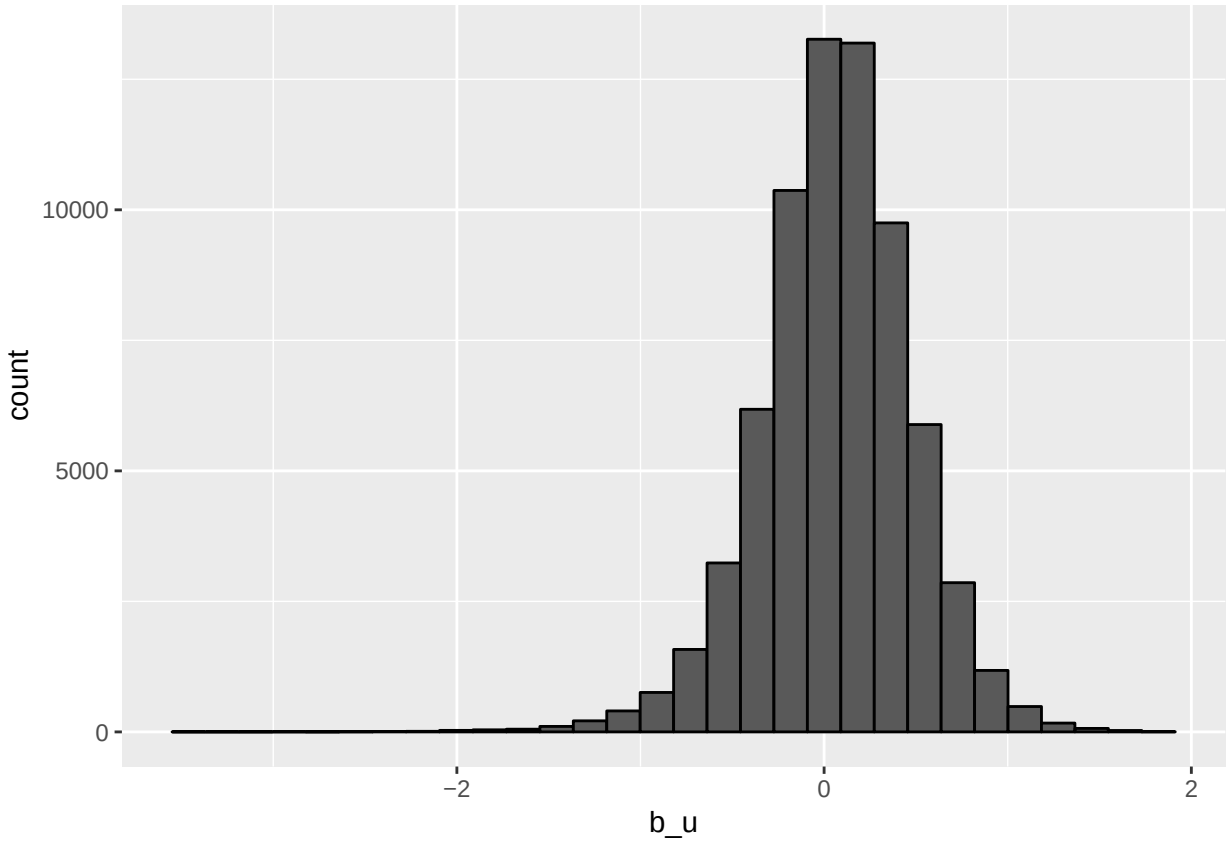
Not surprised, user effects helped us to improve the RMSE, which is now **0.86**. Here is the code to calculate the user effects:

```
#User effects
#Similar to the movie effects, users have their own criteria, we consider the user effects in this mode

#Calculate the user effect (b_u)
#since Y = mean + b_i + b_u
#b_u = Y - mean - b_i
user_avgs_4 <- edx %>%
  left_join(movie_avgs_3, by = "movieId") %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu_3 - b_i))

predicted_ratings_4 <- validation %>%
  left_join(movie_avgs_3, by = "movieId") %>%
  left_join(user_avgs_4, by = "userId") %>%
  mutate(prediction_4 = mu_3 + b_i + b_u) %>%
  pull(prediction_4)



#calculate the rmse
rmse_4 <- RMSE(predicted_ratings_4, validation$rating)

#store rmse into results
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie Effect + User Effect",
                                 RMSE = rmse_4))
rmse_results
```

```
## # A tibble: 4 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Just the average           1.06
## 2 Grouped average            1.11
## 3 Movie Effect Model         0.944
## 4 Movie Effect + User Effect 0.865
```

We can use this peice of code to get an idea of user effects. We can see the distribution of user effects are quite normalized.

```
user_avgs_4 %>% ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```



Results

Here is a summary of our models' RMSEs.

```
rmse_results
```

```
## # A tibble: 4 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Just the average           1.06
## 2 Grouped average            1.11
## 3 Movie Effect Model         0.944
## 4 Movie Effect + User Effect 0.865
```

Conclusion

The models used here is quite simple but capture the main aspects of the data. Some data such as genres and timestamp are not used in the models, which can be considered for a further challenge.

I would like to thank edX and professor Rafael A. Irizarry to offer me a great chance to learn from the Data Science Course. And also like to thank all the online students and TAs, who helped me understand the course better.

References

Rafael A. Irizarry https://rafalab.github.io/dsbook/large-datasets.html#recommendation-systems

edX HarvardX: PH125.9x Data Science: Capstone https://courses.edx.org/courses/course-v1:HarvardX+PH125.9x+2T2018/course/