Data Science Capstone: MovieLens Project

Shannon Hakkal

April 3, 2020

Introduction

The goal of this project was to build an effective recommendation system that predicts likely ratings for movie watchers based on a dataset of millions of past movie ratings called MovieLens10M. This task was approached by implementing and examining the results of four different supervised machine-learning algorithms on a subset of the dataset. Each algorithm was improved upon successively in order to minimize the Root Mean Squared Error (RMSE) of the predictions. The benchmark for success was set at an RMSE of 0.86490 or below.

The models developed included a Just the Average Model, an Average Plus Movie Effect Model, an Average Plus Movie and User Effects Model, and a Regularized Average Plus Movie and User Effects Model. Each model reduced the RMSE from the previous one, and the final model, Regularized Average Plus Movie and User Effects, was determined to be the best model overall. When tested on the validation dataset at the end of the project, the Regularized Average Plus Movie and User Effects Model produced an RMSE of 0.8648177, which is below the target of 0.86490.

Methods

To complete this project, we accessed and imported the MovieLens10M dataset from the GroupLens website: https://grouplens.org/datasets/movielens/10m/. We then used R to split the enormous dataset into two subsets called "edx" and "validation." The edx set was used to train and test each of the four algorithms we proposed. The validation set was only used to test the final, optimized model at the end of the analysis.

Materials

The materials used in this project are listed here.

- MovieLens10M dataset
- R and RStudio

Procedure

1. Below is the code we used to import the MovieLens10M dataset into RStudio and set up the "edx" and "validation" sets. The edx set (90% of MovieLens10M) was used to train and test the proposed recommendation system models, and the validation set (10% of MovieLens10M) was only used to test the final model with the lowest RMSE.

```
# Step 1: Set up the movielens10M data into "edx" and "validation" sets in
order to test the final recommendation system model with the lowest RMSE.

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

# MovieLens10M dataset:
 # https://grouplens.org/datasets/movielens/10m/
 # http://files.grouplens.org/datasets/movielens/ml-10m.zip

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")
 movies <- as.data.frame(movies) %>% mutate(movieId =
as.numeric(levels(movieId))[movieId],
                                            title = as.character(title),
                                            genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens10M data
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)
 edx <- rbind(edx, removed)

# Remove no longer needed objects.
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

2. The next step was to further split the edx dataset into training and test sets in order to implement and evaluate the proposed algorithms. The code for this step is below.

```
# Step 2: Split the edx data into train and test sets in order to explore
different training models.

library(caret)
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list =
FALSE)
train_set <- edx[-test_index,]
```

```
temp <- edx[test_index,]

# Same process used earlier with edx and validation sets:
test_set <- temp %>%
semi_join(train_set, by = "movieId") %>%
semi_join(train_set, by = "userId")

removed <- anti_join(temp, test_set)
train_set <- rbind(train_set, removed)
rm(test_index, temp, removed)
```

3. To evaluate the models developed here, we will apply them to the training set and then compare the resulting predictions with the test set to determine the root mean squared error (RMSE) of each model. The RMSE is calculated with the following formula, where $y_{u,i}$ is defined as the rating for movie i by user u, and our prediction is denoted by $\hat{y}_{u,i}$.

   $$\text{RMSE} = \sqrt{\boxed{1/N\sum_{u,i}(\hat{y}_{u,I} - y_{u,i})^2}}$$

   with N equal to the number of user/movie combinations and the sum $\boxed{\sum}$ adding up all of these combinations.

   The RMSE represents the typical error when predicting a movie rating with a particular model. If the RMSE > 1, it means that the typical error is larger than one star, which is a poor prediction.

   Below is the code we used to build a function to compute the RMSE for each model.

```
# Step 3: Build a function that computes the RMSE for vectors of ratings and
their corresponding predictors.

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

4. Now we proceed to the training and testing of four different models for recommendation systems. The first model is called Just the Average Model, and the code is displayed below.

   First Model: Just the Average

   The simplest possible recommendation system predicts the same rating for all movies regardless of any other variable. This model can be represented like this:

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

   with $\varepsilon_{u,i}$ representing independent errors sampled from the same distribution centered at 0, and μ representing the “true” rating for all movies. The estimate that minimizes the

RMSE is the least squares estimate of μ (represented by μ_hat). Here, this is simply the mean of all ratings.

```
# Just the Average Model:

# Compute the estimate of µ:
mu_hat <- mean(train_set$rating)
mu_hat
[1] 3.512456

# Compute the RMSE:
naive_rmse <- RMSE(test_set$rating, mu_hat)
naive_rmse
[1] 1.060054

# Start a table to keep track of models and RMSEs:
rmse_results <- tibble(Method = "Just the Average Model", RMSE = naive_rmse)
rmse_results

# A tibble: 1 x 2
  Method                        RMSE
  <chr>                         <dbl>
1 Just the Average Model        1.06
```

The RMSE produced by using the Just the Average Model (1.06) is far above our target of less than 0.86490. In the next model, we will add a parameter for movie effect or "bias."

Second Model: Average Plus Movie Effect

Intuitively, it makes sense that more well-known movies are generally given higher ratings than others. It might be worthwhile to augment the Just the Average Model by adding the term b_i to represent the average ranking for movie i:

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

Here, the least squares estimate of b_i is the average of $Y_{u,I} - \mu\_hat$ for each movie i.

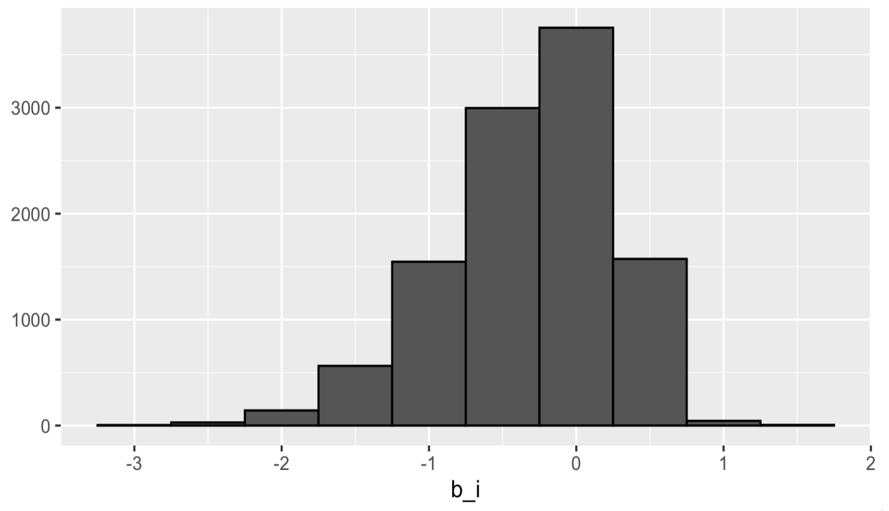
```
# Average Plus Movie Effect Model:

# Compute the estimate of b_i: ("hat" notation for estimates dropped from
here on)
mu <- mean(train_set$rating)
movie_avgs <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = mean(rating - mu))
```

One can see by this histogram (Figure 1) that there is in fact a movie effect on the data. The estimates of b_i vary substantially depending on the movie being rated.

```
# Plot histogram of b_i:
```

```
qplot(b_i, data = movie_avgs, bins = 10, color = I("black"))
```

Figure 1: b_i represents the average of each individual rating minus the average movie rating



We therefore decided to include the movie effect b_i in this second model.

```
# Compute the RMSE:
predicted_ratings <- test_set %>%
left_join(movie_avgs, by='movieId') %>%
mutate(pred = mu + b_i) %>%
pull(pred)
MovieEffect_rmse <- RMSE(predicted_ratings, test_set$rating)

MovieEffect_rmse
[1] 0.9429615

# Add to earlier table of models and RMSEs:
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Average Plus Movie Effect Model",
                                 RMSE = MovieEffect_rmse))

rmse_results %>% knitr::kable()

|Method                          |      RMSE|
|:-------------------------------|---------:|
|Just the Average Model          | 1.0600537|
|Average Plus Movie Effect Model | 0.9429615|
```

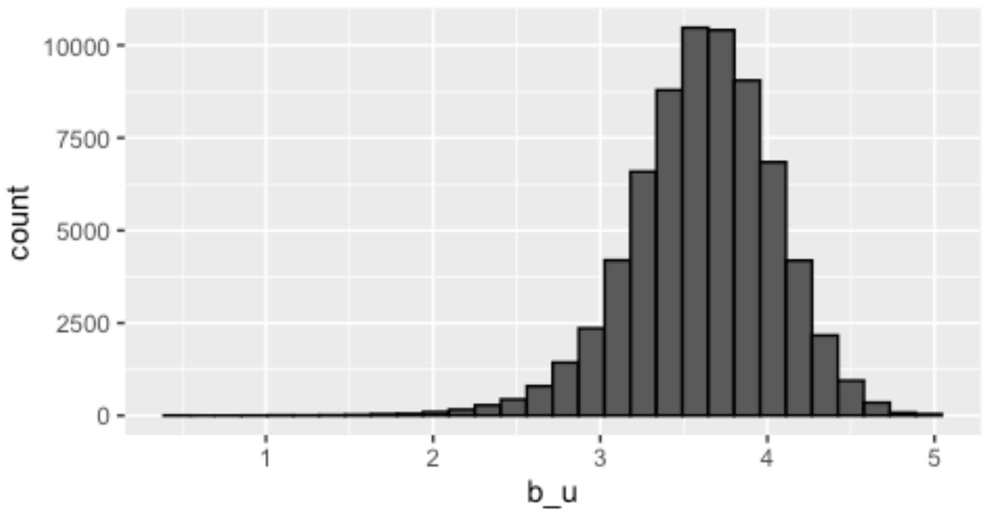
The resulting RMSE from the Average Plus Movie Effect Model (0.943) is still higher than our goal of 0.86490 or below. In the next model, we will add an additional parameter for user effect.

Third Model: Average Plus Movie and User Effects

One can see by the plot below (Figure 2) that there is also a user effect on the data. Some users are very generous with high ratings, while other users are not. This histogram shows the average ratings for users who have rated more than 100 films.

```
# Plot histogram of user effect:

train_set %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```

Figure 2: b_u represents the average ratings of users who rated >100 movies



If we add this user effect b_u to the Average Plus Movie Effect model, it will look like this:

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

```
# Average Plus Movie and User Effects Model:

# Compute the estimate of b_u:
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

# Compute the RMSE:
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)
```

```
M_U_rmse <- RMSE(predicted_ratings, test_set$rating)

M_U_rmse

[1] 0.8646843

# Add to earlier table of models and RMSEs:
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Average Plus Movie and User Effects
Model", RMSE = M_U_rmse))

rmse_results %>% knitr::kable()

|Method                                    |      RMSE|

|:-----------------------------------------|---------:|

|Just the average Model                    | 1.0600537|

|Average Plus Movie Effect Model           | 0.9429615|

|Average Plus Movie and User Effects Model | 0.8646843|
```

The RMSE resulting from the Average Plus Movie and User Effects Model (0.86468) is below our target of 0.86490. In the next model, we will try to minimize the RMSE even further through the use of regularization.

Fourth Model: Regularized Average Plus Movie and User Effects

As noted in the second model (Average Plus Movie Effect), it is clear that well-known movies are rated more often (and more highly) than obscure movies. These little-known films were mostly rated by very few users, sometimes just one. These small sample sizes lead to errors in the prediction estimates because there is greater uncertainty, so larger estimates of b_i, positive or negative, are more likely to occur. These large errors, in turn, increase the RMSE of the model, making it less reliable.

In order to account for the errors caused by small sample sizes for obscure films, we tuned the Average Plus Movie and User Effects Model through regularization. Regularization constrains the variability of the movie and user effects by penalizing large estimates from small sample sizes.

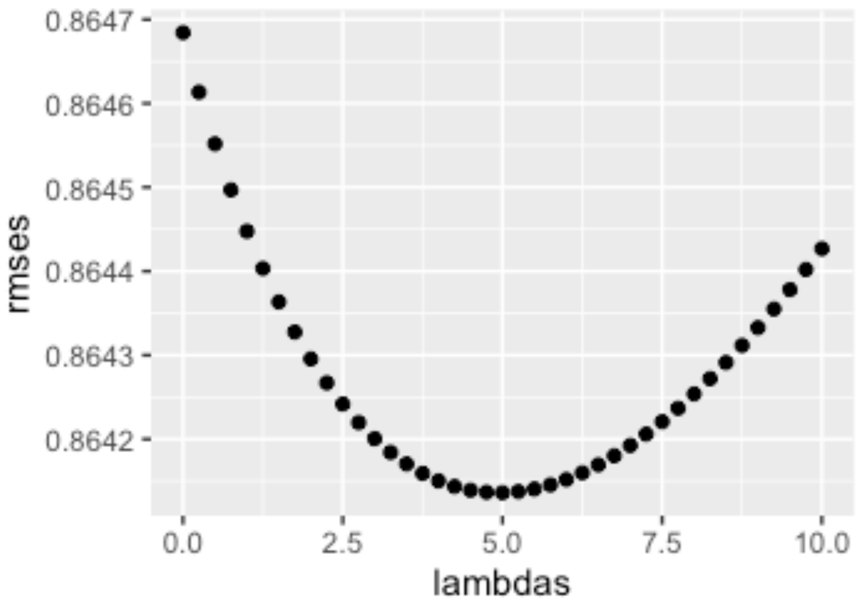
A regularization model with penalty terms for movie and user effects would look like this:

$$1/N\sum_{u,i}(y_{u,I} - \mu - b_i - b_u)\hat{}2 + \lambda(\sum_i b\hat{}2_i + \sum_u b\hat{}2_u)$$

The lambda (λ) in the model is a parameter that can be tuned to minimize the RMSE. We used cross-validation to determine the best lambda for this regularized model.

```
# Regularized Average Plus Movie and User Effects Model:

# Cross-validation to determine the best lambda for Regularized Average +
Movie and User Effects Model:

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) %>%
pull(pred)
return(RMSE(predicted_ratings, test_set$rating))
})

# Plot the RMSEs vs. lambdas to see which lambda minimizes the RMSE:
qplot(lambdas, rmses)
```

Figure 3: Plot of RMSEs vs. possible lambdas for Regularization



The plot reveals that the optimal lambda that minimizes the RMSE is 5.

```
# Verify the optimal lambda from the plot:
lambda <- lambdas[which.min(rmses)]
lambda
[1] 5

# Pull predictions based on the Regularized Average + Movie and User Effects
Model using the optimal value of lambda:

lambda <- 5

Reg_rmses <- sapply(lambda, 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) %>%
        pull(pred)
 return(RMSE(predicted_ratings, test_set$rating))
})

Reg_rmses
[1] 0.8641362

# Add to earlier table of models and RMSEs:
rmse_results <- bind_rows(rmse_results,
                tibble(Method="Regularized Average Plus Movie and User
Effects Model", RMSE = Reg_rmses))

rmse_results %>% knitr::kable()

|Method                                                |      RMSE|
|:-----------------------------------------------------|---------:|
|Just the average Model                                | 1.0600537|
|Average Plus Movie Effect Model                       | 0.9429615|
|Average Plus Movie and User Effects Model             | 0.8646843|
|Regularized Average Plus Movie and User Effects Model | 0.8641362|

# Regularization improved the previous models to an RMSE of 0.8641362.
```

We can see that regularization improved the previous models to an RMSE of 0.8641362.

Results

After training and analyzing four different models of recommendation systems, we determined that the best and final model for prediction is the **Regularized Average Plus Movie and User Effects Model** shown here.

$$1/N\sum_{u,i}(y_{u,I} - \mu - b_i - b_u)^2 + \lambda(\sum_i b^{2}_i + \sum_u b^{2}_u)$$

Finally, we returned to the edx and validation sets created at the beginning of this project and tested the final model for our recommendation system using those original datasets.

```
# Test the final model (Regularized Average + Movie and User Effects) with
the edx and the validation sets.

lambda <- 5

Reg_rmses_edx <- sapply(lambda, function(l){

    mu <- mean(edx$rating)

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

    b_u <- edx %>%
        left_join(b_i, by="movieId") %>%
        group_by(userId) %>%
        summarize(b_u = sum(rating - b_i - mu)/(n()+l))

    predicted_ratings <- validation %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_u, by = "userId") %>%
        mutate(pred = mu + b_i + b_u) %>%
        pull(pred)
  return(RMSE(predicted_ratings, validation$rating))
})

Reg_rmses_edx
[1] 0.8648177
```

The recommendation system that we built using **the Regularized Average Plus Movie and User Effects Model** made predictions on the validation set with an RMSE of 0.8648177. The goal of achieving an RMSE of less than 0.86490 was reached and exceeded using this algorithm.

Conclusion

The goal of this project was to design an efficient recommendation system that predicts likely film ratings based on a dataset of millions of past movie ratings called MovieLens10M. We approached this task by implementing and examining the results of four different machine-learning algorithms on a subset of the MovieLens10M data. We adjusted each algorithm with successive improvements in order to minimize the Root Mean Squared Error (RMSE) of the predictions. Our ultimate goal was to achieve an RMSE of less than 0.86490.

Of the four models we developed, the Regularized Average Plus Movie and User Effects Model was determined to be the best model overall. When tested on the validation dataset at the end of the analysis, the Regularized Average Plus Movie and User Effects Model produced an RMSE of 0.8648177, which is below the target of 0.86490. The results indicated that the recommendation system we developed using this final model was effective.

Even though the overall goal was achieved in this recommendation system project, it is still possible to improve upon our final model. The accuracy of the predictions could be increased through the utilization of other machine-learning techniques, both supervised and unsupervised, such as K-Means Clustering, K-Nearest Neighbors, Principle Component Analysis, or Random Forest. Applying different types of algorithms to this data could reveal further insights to enhance the recommendation system.