MovieLens Project

Patricia McSharry

6/6/2019

Patricia McSharry, HarvardX: PH125.9x Data Science Certificate

Executive Summary

The course provided code to download the MovieLens database and create a train set, edx, and a test set, validation. This code was inserted into this rmd and executed.

The goal of the project is to create a machine learning algorithm to obtain a residual mean squared error (RMSE) of <= 0.87750. I went through four models, each improving on the RMSE result, and put the results in a tibble, and present the results at the end to show the improvement in reducing the RMSE. The final model is for movie, user and year difference (year rate - year movie rate) effects and achieved an RMSE of 0.8649038.

This code was developed in R Studio Version 1.2.1335 and R version 3.5.2.

Machine Learning Algorithm Development Procuedure and Details

Base model: Used the average movie rating of the training set to predict the ratings in the test set and calculated the resulting RMSE of the test set. The RMSE was 1.0612018.

Movie Effect Model: To account for the fact that some movies are typically rated higher than other movies, Owrote an algorithm to account for the movie effect. The algorithm was run on the edx train set. The results were used to predict the ratings in the test set and the resulting RMSE was calculated. The RMSE went down to 0.9439087.

Movie and User Effects Model: Revised the algorithm to account for the user effect, or user bias, and incorporated it with the movie effect algorithm. The revised algorithm was trained on the edx train set and the results were used to predict the ratings of the validation set. The resulting RMSE calculation went down to 0.8653488. Although the RMSE is lower than the goal of 0.87750, I wanted to get it even lower.

Movie, User and Year Difference Effects Model: Older movies are often rated differently than more recent movies. To account for this "year difference effect", code was written to calculate the difference between the year the movie was made and the year the movie was reviewed. In this alogorithm this is referred to as the "year difference" effect. Code was written to account for year difference effect and it was combined with the previous movie and user effect algorithm. The revised algorithm was trained on the edx train set and the results were used to predict the ratings of the validation set. The resulting RMSE calculation went down to 0.8649038

Run the code to set up edx and validation dataframes, provided as part of course material.

The code is available in the rmd file, echo=FALSE to unclutter the PDF.

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

```
## Warning: package 'tidyverse' was built under R version 3.5.3
```

```
## -- Attaching packages ------------------------------------ tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.1     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
```

```
## v readr   1.3.1     v forcats 0.4.0
## Warning: package 'ggplot2' was built under R version 3.5.3
## Warning: package 'tibble' was built under R version 3.5.3
## Warning: package 'tidyr' was built under R version 3.5.3
## Warning: package 'readr' was built under R version 3.5.3
## Warning: package 'purrr' was built under R version 3.5.3
## Warning: package 'dplyr' was built under R version 3.5.3
## Warning: package 'stringr' was built under R version 3.5.3
## Warning: package 'forcats' was built under R version 3.5.3
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## Loading required package: caret
## Warning: package 'caret' was built under R version 3.5.3
## Loading required package: lattice
##
## Attaching package: 'caret'
## The following object is masked from 'package:purrr':
##
##     lift
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

Machine Learning Algorithm Models Code and Results Table

```
# Patricia McSharry - Movielens project

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

```
## Loading required package: lubridate
## Warning: package 'lubridate' was built under R version 3.5.3
##
## Attaching package: 'lubridate'
## The following object is masked from 'package:base':
##
##     date
```

```
# Function to calculate residual mean squared error (RMSE)
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2))
  }

# Base model - use the average movie rating of train set to
# predict ratings in test set
```

```r
# average rating of train set
mu <- mean(edx$rating)

# calculate RMSE
basic_rmse <- RMSE(validation$rating,mu)
# Store the results in the rmse_results table
rmse_results <- data_frame(model = "Basic Model", RMSE = basic_rmse)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```r
# Movie effect model.
# Group the train set by movieId
# Train the edx set using the movie effect algorithm
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_movie = mean(rating - mu))

# Using results of training, predict the ratings in the test set
movie_predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_movie)

# Calculate the RMSE
model_movie_effect <- RMSE(movie_predicted_ratings, validation$rating)
# Store the results in the rmse_results table
rmse_results <- bind_rows(rmse_results,
  data_frame(model="Movie Effect Model",
  RMSE = model_movie_effect))

# Movie and User Effects Model
# group the edx train set by userID and incorporate the movie effect
# Train the edx set using the movie and userid algorithm
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_user = mean(rating - mu - b_movie))

# Using results of training, predict the ratings in the test set
user_movie_predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_movie + b_user) %>%
  pull(pred)

# Calculate the RMSE
model_user_movie_rmse <- RMSE(user_movie_predicted_ratings, validation$rating)
# Store the results in the rmse_results table
rmse_results <- bind_rows(rmse_results,
  data_frame(model="Movie and User Effects Model",
  RMSE = model_user_movie_rmse))

# Movie, User and Year Difference Effects Model
```

```r
# Modify the edx and validation sets to add "year" to hold the year the movie was
# made and to add "yeardiff" to calculate the difference between the year the movie
# was reviewed and the year it was made.


# modify edx to add year and yeardiff
edx <- edx %>% mutate(year = as.numeric(substring(edx$title,nchar(edx$title)-4,nchar(edx$title)-1)))

edx <- edx %>% mutate(yeardiff =
as.numeric(substring(as_datetime(edx$timestamp),1,4))-year)

# modify validation to add year and yeardiff
validation <- validation %>% mutate(year = as.numeric(substring(validation$title,nchar(validation$title)

validation <- validation %>% mutate(yeardiff =
as.numeric(substring(as_datetime(validation$timestamp),1,4))-year)

# group the edx train set by yeardiff and incorporate the movie and user effects
# Train the edx set using the movie and userid algorithm
year_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(yeardiff) %>%
  summarize(b_year = mean(rating - mu - b_movie - b_user))

# Using results of training, predict the ratings in the test set
year_user_movie_predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(year_avgs, by='yeardiff') %>%
  mutate(pred = mu + b_user + b_movie + b_year) %>%
  pull(pred)

# Calculate the RMSE

model_year_user_movie_rmse <- RMSE(year_user_movie_predicted_ratings, validation$rating)
# Store the results in the rmse_results table
rmse_results <- bind_rows(rmse_results,
  data_frame(model="Movie, User and Year Difference Effects Model",
  RMSE = model_year_user_movie_rmse))
# print out the table
rmse_results %>% knitr::kable()
```

model	RMSE
Basic Model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effects Model	0.8653488
Movie, User and Year Difference Effects Model	0.8649038

Conclusion

The algorithm successful meets the goal of RMSE <= 0.87750, with a final RMSE of 0.8649038. An iterative method to develop this machine learning algorithm was the best approach for this solution. Each successive iteration of the algorithm improved the performance. The final model was the Movie, User and Year Difference Effects Model with an RMSE of 0.8649038.