Movie lens Rating Prediction Project

Sandeep Kumar

23/02/2022

Introduction

Recommendation systems use ratings that users have given to items to make specific recommendations.Items for which a high rating is predicted for a given user are then recommended to that user.

Recommendation systems are one of the most used models in machine learning algorithms.For this project we will focus on create a movie recommendation system using the 10M version of MovieLens dataset, collected by GroupLens Research.

Goal of the project

The goal in this project is to train a machine learning algorithm that predicts user ratings using the inputs of a provided subset (edx dataset provided by the staff) to predict movie ratings in a provided validation set.

The function that computes the RMSE for vectors of ratings and their corresponding predictors will be the following:

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

Finally, the best resulting model will be used to predict the movie ratings.

Dataset

The MovieLens dataset is automatically downloaded

- [MovieLens 10M dataset] https://grouplens.org/datasets/movielens/10m/
- [MovieLens 10M dataset - zip file] http://files.grouplens.org/datasets/movielens/ml-10m.zip

```
#############################################################
# Create edx set, validation set, and submission file
#############################################################
# Note: this process could take a couple of minutes for loading required package: tidyverse and package
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")
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
ratings <- read.table(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)
```

```r
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")
```

```r
# The Validation subset will be 10% of the MovieLens data.
set.seed(1)
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 subset:
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)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

The MovieLens dataset will be splitted into 2 subsets that will be the “edx”, a training subset to train the algorithm, and “validation” a subset to test the movie ratings.

Algorithm development is to be carried out on the “edx” subset only, as “validation” subset will be used to test the final algorithm.

Modelling Approach

Compute the RMSE, defined as follows:

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

The RMSE is our measure of model accuracy. We can interpret the RMSE similarly to a standard deviation: it is the typical error we make when predicting a movie rating. If its result is larger than 1, it means that our typical error is larger than one star, which is not a good result.

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

I. Average rating model

The averge rating model predicts the same rating for all movies, so we compute the dataset’s mean rating. We know that the estimate that minimize the RMSE is the least square estimate the average of all ratings: The expected rating of the underlying data set is between 3 and 4.

```r
mean_rating <- mean(edx$rating)
mean_rating
```

```
## [1] 3.512464
```

If we predict all unknown ratings with mean_rating, we obtain the first naive RMSE:

```
naive_rmse <- RMSE(validation$rating, mean_rating)
naive_rmse
```

```
## [1] 1.060651
```

Here, we represent results table with the first RMSE:

```
rmse_results <- data_frame(method = "Average movie rating model", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

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

method	RMSE
Average movie rating model	1.060651

The above RMSE give us our baseline RMSE to compare with next modelling approaches.

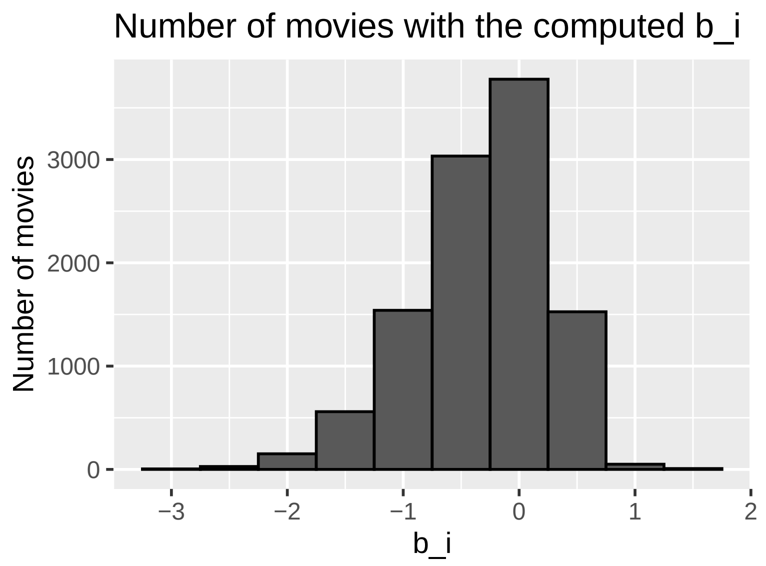
II. Movie effect model

We know that some movies are just generally rated higher than others. Higher ratings are mostly linked to popular movies among users and the opposite is true for unpopular movies. We compute the estimated deviation of each movies' mean rating from the total mean of all movies μ. The resulting variable is called "b" (as bias) for each movie "i" b_i, that represents average ranking for movie i:

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

The histogram is left skewed, implying that more movies have negative effects

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mean_rating))
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"),
ylab = "Number of movies", main = "Number of movies with the computed b_i")
```



This is called the penalty term movie effect.

```
predicted_ratings <- mean_rating +  validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
model_s_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie effect model",
                                     RMSE = model_s_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0606506
Movie effect model	0.9437046

So we have predicted movie rating based on the fact that movies are rated differently by adding the computed b_i to $mean_rating$.

We can see an improvement but this model does not consider the individual user rating effect.

III. Regularized effect model

The use of the regularization permits to penalize the aspects of movies with very few ratings and in some users that only rated a very small number of movies. We should find the value of lambda (that is a tuning parameter) that will minimize the RMSE.

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){

  mean_rating <- mean(edx$rating)

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

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

  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mean_rating + b_i + b_u) %>%
    pull(pred)

  return(RMSE(predicted_ratings, validation$rating))
})
```

For the full model, the optimal lambda is:

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

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

For the full model, the optimal lambda is: 5.25

The new results will be:

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized effect model",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0606506
Movie effect model	0.9437046
Regularized effect model	0.8649857

Results

The RMSE values of all the 3 represented models are the following:

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

method	RMSE
Average movie rating model	1.0606506
Movie effect model	0.9437046

method	RMSE
Regularized effect model	0.8649857

We therefore found the lowest value of RMSE that is 0.8648170.

Conclusion

The regularized model including the effect of user is characterized by the lower RMSE value and is hence the optimal model to use for the present project. The optimal model characterised by the lowest RMSE value (0.8648170) lower than the initial evaluation criteria (0.8775) given by the goal of the present project.