Recommendation System Report - MovieLens

Romina Jaramillo

7/4/2021

Overview

Machine learning is applied in predictive analysis. The historical data is used to predict the future outcomes. To illustrate, given one x value, you can predict any value. One of the goals of machine learning is to process data and generate useful information to personalize the information given to every user. In 2006, Netflix implemented a contest to optimize their recommendation algorithm in a 10%. The contestants had to learn as much as possible about machine learning and algorithms to solve that big problem. The linear model had to be trained to generate predicted movies scores for the users and calculate the Root Mean Square Error (RMSE) of the predicted ratings versus the actual scores. The MovieLens project, as the Netflix prize, create a recommendation system. But, In this case the Edx libraries and data set will be used in the training of the algorithms and movie ratings prediction.

```r
summary(edx)
```

```
##      userId         movieId          rating        timestamp        
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08  
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08  
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09  
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09  
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09  
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09  
##     title              genres         
##  Length:9000055     Length:9000055    
##  Class :character   Class :character  
##  Mode  :character   Mode  :character  
##                                       
##                                       
##                                       
```

```r
dim(edx)
```

```
## [1] 9000055       6
```

```r
class(edx)
```

```
## [1] "data.table" "data.frame"
```

```r
head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

```r
nrow(edx)
```

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

```r
# Most rated movies
edx %>% group_by(title) %>% summarize(n_ratings = n()) %>% arrange(desc(n_ratings))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 10,676 x 2
##    title                                                        n_ratings
##    <chr>                                                            <int>
##  1 Pulp Fiction (1994)                                              31362
##  2 Forrest Gump (1994)                                              31079
##  3 Silence of the Lambs, The (1991)                                 30382
##  4 Jurassic Park (1993)                                             29360
##  5 Shawshank Redemption, The (1994)                                 28015
##  6 Braveheart (1995)                                                26212
##  7 Fugitive, The (1993)                                             25998
##  8 Terminator 2: Judgment Day (1991)                                25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)     25672
## 10 Apollo 13 (1995)                                                 24284
## # ... with 10,666 more rows
```

Based on the data size that this data set involves. This report is going to use a linear model to solve the problem. The RMSE is the error amount that is between two values. For example, it compares a predicted value and a known value.

After overview the data, we could say that the train set is conformed by 9000055 rows and 6 columns. The Validation set is 10% of Movie Lens data and the other 90% is train set. In the same way, the validation train has 999,999 occurrences and 6 columns as shown.

```r
# printing the validation and training data
glimpse(validation)
```

```
## Rows: 999,999
## Columns: 6
## $ userId    <int> 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, ...
## $ movieId   <dbl> 231, 480, 586, 151, 858, 1544, 590, 4995, 34, 432, 434, 8...
## $ rating    <dbl> 5.0, 5.0, 5.0, 3.0, 2.0, 3.0, 3.5, 4.5, 5.0, 3.0, 3.0, 3....
## $ timestamp <int> 838983392, 838983653, 838984068, 868246450, 868245645, 86...
## $ title     <chr> "Dumb & Dumber (1994)", "Jurassic Park (1993)", "Home Alo...
## $ genres    <chr> "Comedy", "Action|Adventure|Sci-Fi|Thriller", "Children|C...
```

```
glimpse(edx)
```

```
## Rows: 9,000,055
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 42...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 83...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)",...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|...
```

Analysis and Methods

The perfect model for this project is the lineal regression model. As we know, linear regression is an approach for modeling the relationships between scalar dependent variables (y) and one or more independent variables (x). This project uses simple techniques like linear regression model with regularized movie and user effects using lambda for validation. The model used is the following:

$$A_{x,y} = \mu, \tag{1}$$

The predicted rating is represented by $A_{x,y}$ where x represents user and y movie. The predicted rating is equal to the mean or average rating between all the registries or entries.

```
# calculate the average of all ratings of the edx dataset
media <- mean(edx$rating)
RMSE(validation$rating, media)
```

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

Before fine-tuning our model, we must analyze and understand the situation. It is observed that the most rating grades in the users were 4 and 3.

```
#summary of each rating count
edx %>% group_by(rating) %>% summarize(count = n()) %>% top_n(8, count) %>%
arrange
```

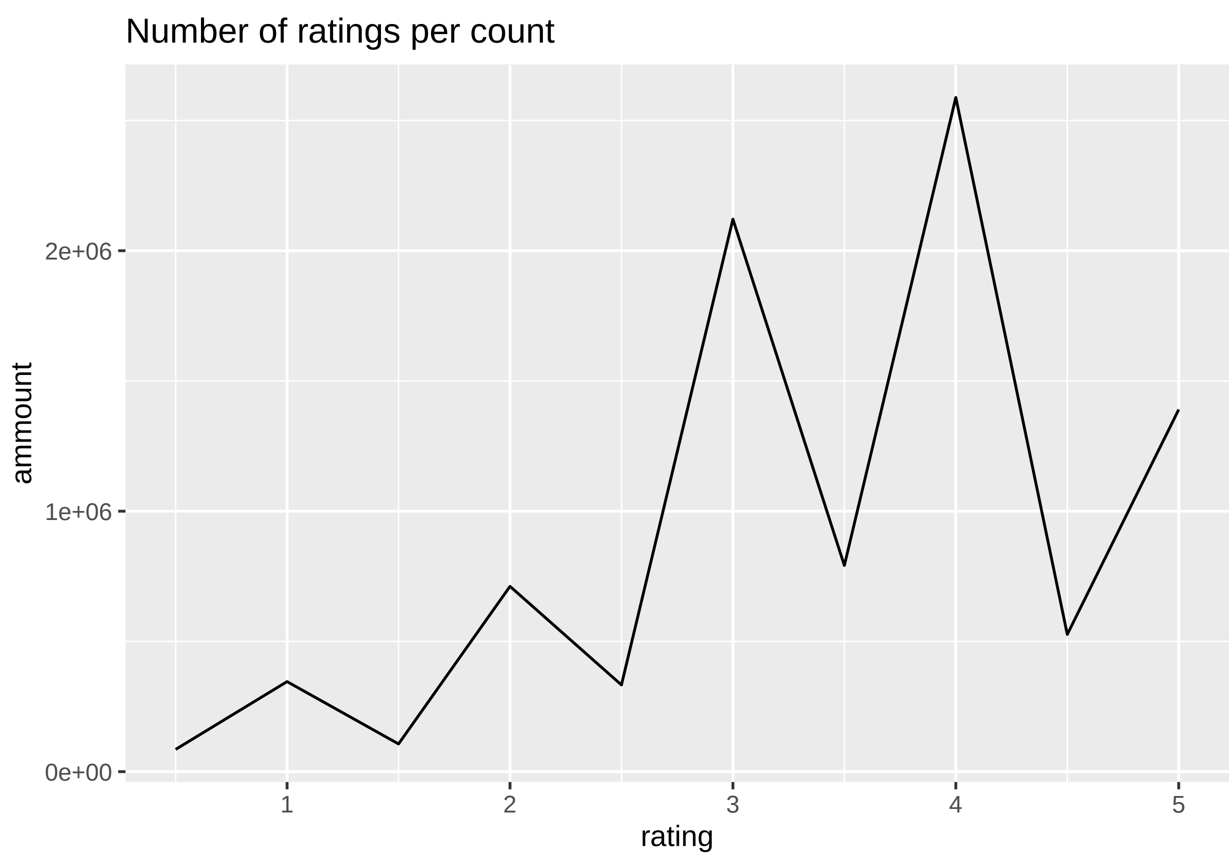
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 8 x 2
##   rating   count
##    <dbl>   <int>
```

```
## 1    1    345679
## 2    2    711422
## 3    2.5  333010
## 4    3   2121240
## 5    3.5  791624
## 6    4   2588430
## 7    4.5  526736
## 8    5   1390114
```

```r
# Rating count plot
edx %>%
group_by(rating) %>%
summarize(count = n()) %>%
ggplot(aes(x = rating, y = count)) +
geom_line()+
labs(x="rating", y="ammount") +
ggtitle("Number of ratings per count")
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```




```r
# movies with the major number of ratings
top_movies <- edx %>% group_by(title) %>%
summarize(count=n()) %>% top_n(10,count) %>%
arrange(desc(count))
```

```
## 'summarise()' ungrouping output (override with '.groups' argument)
```

```r
head(top_movies)
```

```
## # A tibble: 6 x 2
##   title                             count
##   <chr>                             <int>
## 1 Pulp Fiction (1994)               31362
## 2 Forrest Gump (1994)               31079
## 3 Silence of the Lambs, The (1991) 30382
## 4 Jurassic Park (1993)              29360
## 5 Shawshank Redemption, The (1994) 28015
## 6 Braveheart (1995)                 26212
```

The movies with the highest number of ratings are in the top genres categories.

Fine-tuning the model

The bias is a tendency or disproportionate weight in favor or against one thing. For this reason, we are going to add the bias for user and movies (rating differences) to improve our model

$$A_{x,y} = \mu + b_x + b_y, \tag{2}$$

```r
# calculate the error (bias) of movies on the training dataset
bx <- edx %>% group_by(movieId) %>% summarize(bx = mean(rating - media))
```

```
## 'summarise()' ungrouping output (override with '.groups' argument)
```

```r
bx
```

```
## # A tibble: 10,677 x 2
##    movieId      bx
##      <dbl>   <dbl>
##  1       1  0.415
##  2       2 -0.307
##  3       3 -0.365
##  4       4 -0.648
##  5       5 -0.444
##  6       6  0.303
##  7       7 -0.154
##  8       8 -0.378
##  9       9 -0.515
## 10      10 -0.0866
## # ... with 10,667 more rows
```

```r
# predicted ratings
predicted_ratings_bx <- media + validation %>%
left_join(bx, by='movieId') %>% .$bx
```

```r
# calculate the error (bias) of users on the training dataset
by <- edx %>% left_join(bx, by='movieId') %>%
  group_by(userId) %>%
  summarize(by = mean(rating - media - bx))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

Now we are going to predict ratings with movie and user bias. Then, calculate RMSE of movies and users bias effect

```r
# getting new ratings taking care of user and movie errors
predicted_R <- validation %>% left_join(bx, by='movieId') %>%
  left_join(by, by='userId') %>%
  mutate(pred = media + bx + by) %>% pull(pred)

#The root mean square error (RMSE) models for movies and users
rmse_movie <- RMSE(validation$rating,predicted_ratings_bx)
rmse_movie
```

```
## [1] 0.9439087
```

```r
rmse_movie_user <- RMSE(validation$rating, predicted_R)
rmse_movie_user
```

```
## [1] 0.8653488
```

Results

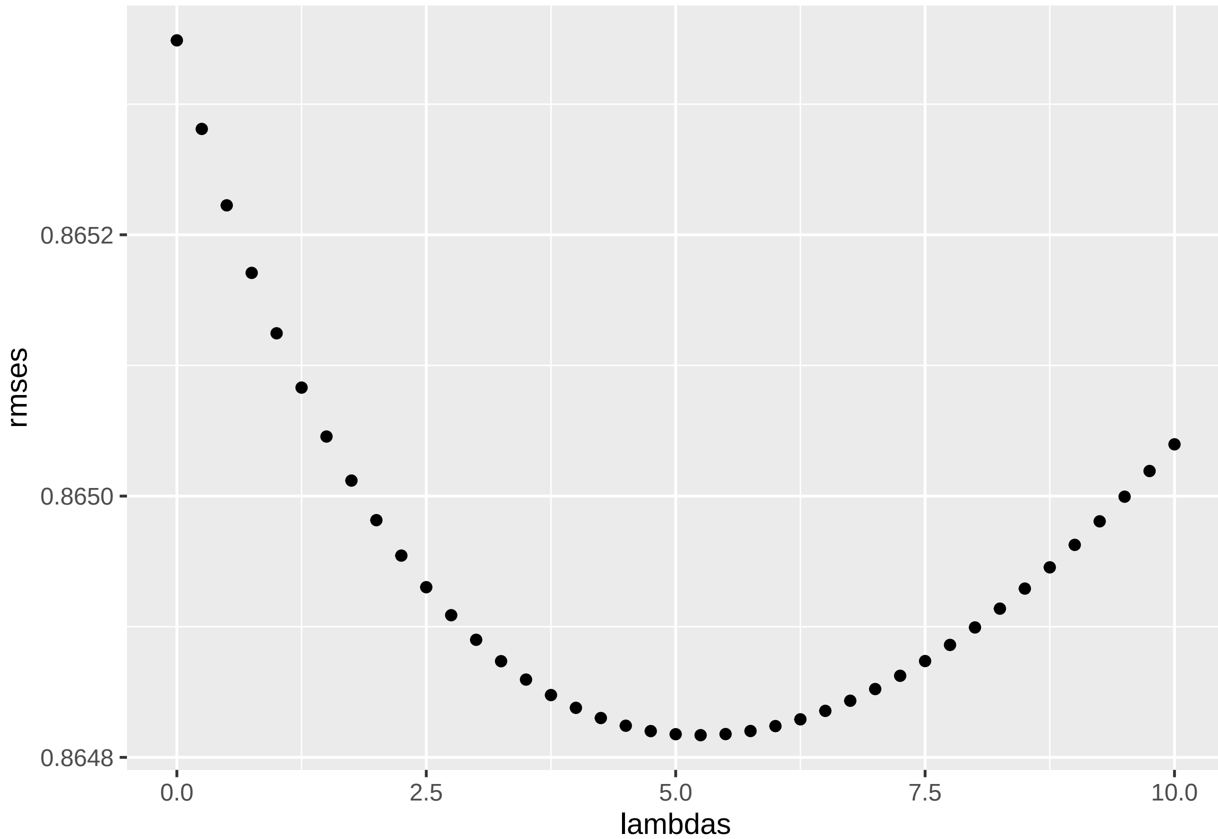
With the objective to reduce the effect of large errors in our predictions, we applied regularization. Regularization is used to allow models to usefully model such data without over fitting. It penalizes inappropriate estimates on sample sizes. To illustrate, the bias user and movie (bx and by) accounts for the average deviation, if there are 1 or 100 ratings to the movie. Regularization let us reduce the impact that an extreme rating or anomalies in the rating of users could cause. The equation could be represented in R like this:

```r
# determine best lambda from a sequence
lambdas <- seq(from=0, to=10, by=0.25)

# determine best lambda from a sequence
rmses <- sapply(lambdas, function(l){ media_reg <- mean(edx$rating)
bx_reg <- edx %>% group_by(movieId) %>% summarize(bx_reg = sum(rating - media)/(n()+l))

by_reg <- edx %>% left_join(bx_reg, by="movieId")%>%
  group_by(userId) %>% summarize(by_reg = sum(rating - bx_reg - media_reg)/(n()+l))

predicted_ratings_bias <- validation %>%
  left_join(bx_reg, by = "movieId") %>%
  left_join(by_reg, by = "userId") %>%
  mutate(pred = media_reg + bx_reg + by_reg) %>% .$pred

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

```
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
qplot(lambdas, rmses)
```




```
# optimal lambda
lambdas[which.min(rmses)]
```

```
## [1] 5.25
```

```
# output RMSE of our final model
rmse_final <- min(rmses)
rmse_final
```

```
## [1] 0.864817
```

The regularization descends the RMSE's value to 0.86481

Conclusion

To conclude, the algorithm is more efficient than other algorithms from R packages. This simple model let us predict movie ratings without consuming a big amount of resources from the computer.

```
rmse_results <- data.frame(methods=c("movie effect",
                                     "movie and user effects" ,"Regularized movie and user effect"), val

rmse_results
```

```
##                             methods    values
## 1                      movie effect 0.9439087
## 2            movie and user effects 0.8653488
## 3 Regularized movie and user effect 0.8648170
```

The RMSE showed that using linear regression with regularization of users and movies is a proper recommended system.

Vocabulary

1. RMSE (Root Mean Square Error): Value used to evaluate the closeness of the predictions to the true values in the validation set.
2. Bias: Statistical bias is a term that refers to any type of error or distortion that is found with the use of statistical analyses.