Movie Rating Prediction

Julien Rroquelaure

2020/12/16

Introduction

The starting point of the exercise is the *Movielens 10M* dataset:

https://grouplens.org/datasets/movielens/10m/

It contains roughly 10 millions movie ratings. These ratings are done on $\approx 10,000$ movies by $\approx 72,000$ users.

The data is prepared beforehand as follows:

- 90% as training data called **edx**, with every user and every movie being represented.
- 10% as a test set called **validation**.

The dataset has six variables :

- *userId*
- *movieId*
- *rating*
- *timestamp*
- *title*
- *genres*

Our goal is to predict the ratings of the validation set.

In order to measure the accuracy of our prediction, we will compute the **RMSE** (Root Mean Square Error). The RMSE is akin to a distance between our predictions and the actual ratings.

Analysis

1. Test set average

The datasets are very big. **edx** is a 9000055×6 table and **validation** is 999999×6. Early attempts with machine learning algorithms were very slow so we tried another method.

We start with the easiest prediction we can make about the test set. We predict that every rating in the test set is the average rating of the training set.

```
mu <- mean(edx$rating)
mu
```

```
## [1] 3.512465
```

```
RMSE(mu, validation$rating)
```

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

We call μ the mean rating of **edx** and we get $\mu = 3.512465$. With this value as prediction for the whole validation set, we get a RMSE of 1.061202. This value will be our baseline and we aim to improve from here.

2. Movie and user bias

Intuitively, we expect that a rating will depend on the intrinsic quality of a movie, the *movie bias* b_m, and the personal scale of a user, the *user bias* b_u.

So the second step of our analysis is to subtract μ from the test data, and average first by individual movie, and then by individual user.

```
edx_movie_bias <- edx %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating - mu))
edx_user_bias <- edx %>%
  left_join(edx_movie_bias, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_m))
edx_pred <- validation %>%
  left_join(edx_movie_bias, by="movieId") %>%
  left_join(edx_user_bias, by="userId") %>%
  mutate(pred = mu + b_m + b_u)
RMSE(edx_pred$pred, validation$rating)
```

After accounting for both bias, we got a RMSE of 0.8653488.

```
head(edx_pred)
```

```
##    userId movieId rating timestamp
## 1:      1     231      5 838983392
## 2:      1     480      5 838983653
## 3:      1     586      5 838984068
## 4:      2     151      3 868246450
## 5:      2     858      2 868245645
## 6:      2    1544      3 868245920
##                                                 title
## 1:                                Dumb & Dumber (1994)
## 2:                                Jurassic Park (1993)
## 3:                                   Home Alone (1990)
## 4:                                      Rob Roy (1995)
## 5:                               Godfather, The (1972)
## 6: Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
##                                   genres         b_m        b_u     pred
## 1:                                Comedy -0.57734404  1.6792347 4.614356
## 2:         Action|Adventure|Sci-Fi|Thriller  0.15105660  1.6792347 5.342757
## 3:                         Children|Comedy -0.45681303  1.6792347 4.734887
## 4:                 Action|Drama|Romance|War  0.01759325 -0.2364086 3.293650
## 5:                             Crime|Drama  0.90290078 -0.2364086 4.178957
## 6: Action|Adventure|Horror|Sci-Fi|Thriller -0.56718682 -0.2364086 2.708870
```

By printing our predictions, we notice that some predictions are below 0.5 or above 5. We then correct this by bounding our predictions.

```
edx_pred_bound <- edx_pred %>%
  mutate(pred = ifelse(pred>5, 5, pred)) %>%
```

```r
  mutate(pred = ifelse(pred<.5, .5, pred))
RMSE(edx_pred_bound$pred, validation$rating)
```

```r
sum(edx_pred_bound$pred <.5)
```

```
## [1] 0
```

```r
sum(edx_pred_bound$pred >5)
```

```
## [1] 0
```

We have a slight improvement of our RMSE but there is still room for improvement.

3. Regularization

Let's analyze where the biggest contributions to our RMSE come from.

```r
edx_difference <- edx_pred_bound %>%
  mutate(diff = abs(pred - validation$rating)) %>%
  arrange(desc(diff))
```

```r
edx_difference %>%
  filter(diff>2)
```

```
##        userId movieId rating  timestamp
##     1:   3843    2804    0.5 1189713153
##     2:  14159     912    0.5 1219762673
##     3:  37651     858    0.5 1202076481
##     4:  40373    2804    0.5 1069734711
##     5:  69802    5952    0.5 1073161740
##    ---
## 28960:  19540    2959    2.0  943540191
## 28961:  56855    3114    2.0  978323450
## 28962:  25723   37729    2.0 1224571487
## 28963:  58264     235    2.5 1128299974
## 28964:  10758     170    1.0  889718687
##                                            title
##     1:                    Christmas Story, A (1983)
##     2:                            Casablanca (1942)
##     3:                        Godfather, The (1972)
##     4:                    Christmas Story, A (1983)
##     5: Lord of the Rings: The Two Towers, The (2002)
##    ---
## 28960:                            Fight Club (1999)
## 28961:                           Toy Story 2 (1999)
## 28962:                          Corpse Bride (2005)
## 28963:                               Ed Wood (1994)
## 28964:                               Hackers (1995)
##                                      genres       b_m       b_u
##     1:                          Children|Comedy  0.5793507  0.9914838
##     2:                            Drama|Romance  0.8079586  0.9726404
##     3:                              Crime|Drama  0.9029008  1.0178302
##     4:                          Children|Comedy  0.5793507  1.0798614
##     5:                Action|Adventure|Fantasy  0.6069349  1.0883292
##    ---
```

```
## 28960:                 Action|Crime|Drama|Thriller  0.6773597 -0.1896857
## 28961:  Adventure|Animation|Children|Comedy|Fantasy  0.3569514  0.1306917
## 28962:     Animation|Comedy|Fantasy|Musical|Romance  0.1026503  0.3849775
## 28963:                                 Comedy|Drama  0.1510074  0.8366177
## 28964:             Action|Adventure|Crime|Thriller -0.3249990 -0.1874564
##            pred     diff
##     1: 5.000000 4.500000
##     2: 5.000000 4.500000
##     3: 5.000000 4.500000
##     4: 5.000000 4.500000
##     5: 5.000000 4.500000
##    ---
## 28960: 4.000139 2.000139
## 28961: 4.000108 2.000108
## 28962: 4.000093 2.000093
## 28963: 4.500090 2.000090
## 28964: 3.000010 2.000010
```

Furthermore, we notice that around $30,000$ predictions are off by more than 2 points. So we are going to take a slice of the worst predictions and look at it.

```
edx_diff_slice <- edx_difference %>%
  slice(1:30000)
edx_difference %>%
  group_by(userId) %>%
  summarize(n=n(), meandiff=mean(diff)) %>%
  right_join(edx_diff_slice, by="userId") %>%
  select(userId, n, meandiff, pred, diff) %>%
  arrange(desc(meandiff))
```

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

```
## # A tibble: 30,000 x 5
##    userId     n meandiff  pred  diff
##     <int> <int>    <dbl> <dbl> <dbl>
##  1  12217     1     3.80  4.30  3.80
##  2  23333     1     3.67  4.67  3.67
##  3  15162     1     3.65  4.15  3.65
##  4  43789     2     3.60  4.90  4.40
##  5  43789     2     3.60  4.81  2.81
##  6   1007     1     3.53  4.53  3.53
##  7  16550     1     3.52  4.52  3.52
##  8  46456     1     3.38  3.88  3.38
##  9  21285     1     3.30  3.80  3.30
## 10  45890     1     3.27  3.77  3.27
## # ... with 29,990 more rows
```

```
edx_difference %>%
  group_by(movieId) %>%
  summarize(n=n(), meandiff=mean(diff)) %>%
  right_join(edx_diff_slice, by="movieId") %>%
  select(movieId, n, meandiff, pred, diff) %>%
  arrange(desc(meandiff))
```

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

```
## # A tibble: 30,000 x 5
```

```
##    movieId     n meandiff  pred  diff
##      <dbl> <int>    <dbl> <dbl> <dbl>
##  1   31692     1     3.79  4.29  3.79
##  2    5921     1     3.00  3.50  3.00
##  3     672     1     2.95  3.45  2.95
##  4    7253     1     2.94  3.44  2.94
##  5   25945     1     2.93  3.43  2.93
##  6    8733     1     2.93  3.43  2.93
##  7   61931     1     2.85  3.35  2.85
##  8    3193     1     2.74  3.74  2.74
##  9    4348     1     2.70  3.70  2.70
## 10    6150     1     2.69  3.69  2.69
## # ... with 29,990 more rows
```

It seems that the most problematic users and movies are those with a small number of ratings. This is logical since the less we have data, the more variable an observation is in general.

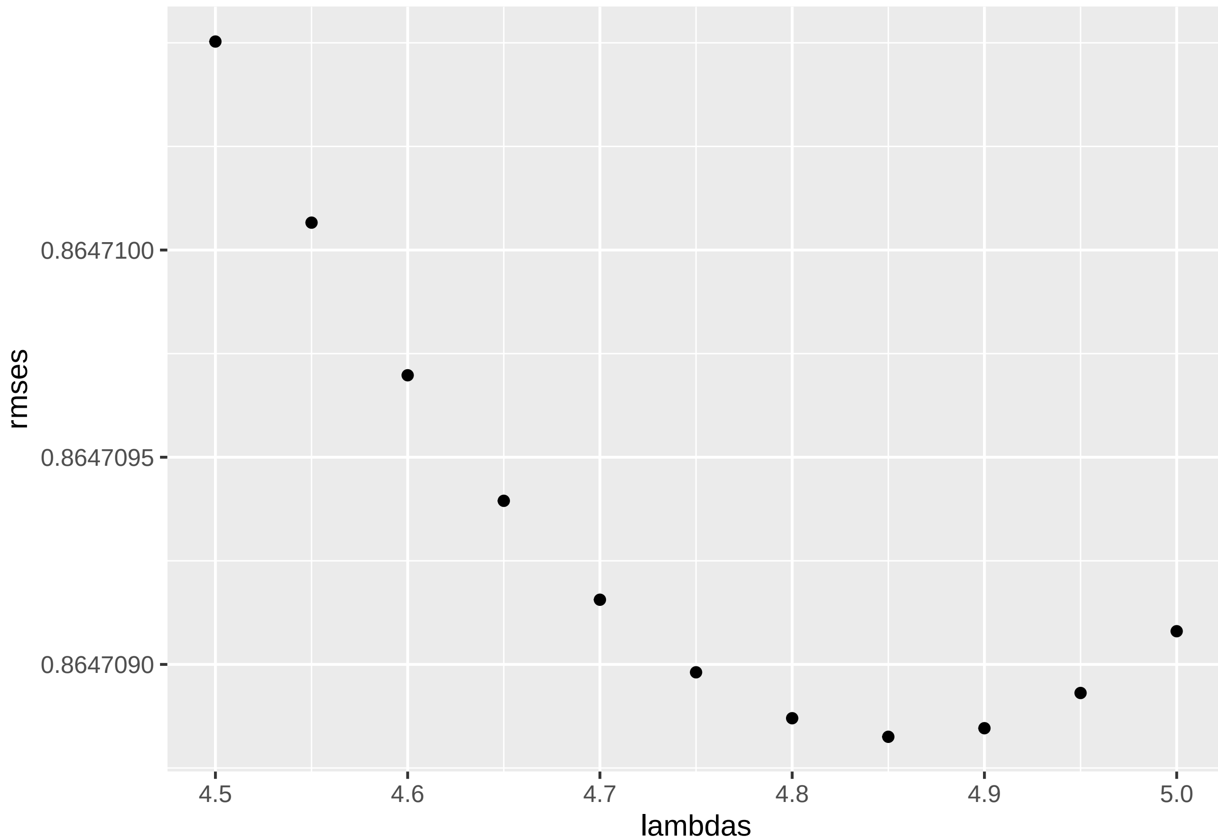
We are then going to use regularization to dampen the rare users and movies, and by bringing them closer to the average rating.

Additionally, we will be varying our parameter λ in order to get an optimal RMSE.

```r
lambdas <- seq(3, 7, .25)
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

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

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

  edx_pred_regs <- validation %>%
    left_join(edx_movie_bias_regs, by = "movieId") %>%
    left_join(edx_user_bias_regs, by = "userId") %>%
    mutate(pred = mu + b_i + b_u)

  edx_pred_regs_bound <- edx_pred_regs %>%
    mutate(pred = ifelse(pred>5, 5, pred)) %>%
    mutate(pred = ifelse(pred<.5, .5, pred))

  return(RMSE(edx_pred_regs_bound$pred, validation$rating))
})
```

We then plot our RMSES function of λ

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



After a further refinement, around $4.5 < \lambda < 5$, we settle with a value of $\lambda = 4.85$

Results

We run our final algorithm, with $\lambda = 4.85$

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

```r
RMSE_final <- min(rmses)
RMSE_final
```

```
## [1] 0.8647088
```

We end up with a RMSE = 0.8647088, which is below our target of 0.86490

Conclusion

For the analysis, we were limited by our computing power, so we didn't use an advanced machine learning algorithm.

Nevertheless, we were able to get a satisfying prediction with the idea that a movie has an intrinsic quality measured by the success among the users. We also guessed that every user has their own scale to rate movies, which is consistent with the behavior we see in our everyday life.

The regularization allowed us to diminish the variance surrounding rare unpredictable users and movies.

One can wonder if a more sophisticated algorithm will improve the predictions, and we would be the trade-off between computing power and precision.

Another improvement can be sought in the unused variables. In particular, *timestamp*: does the rating conventions change in time? And *genres*: are there clusters of genres that appeal to certain users?