MovieLens Project

Shehbaz Singh

11/6/2020

Introduction/Overview/Executive Summary

The goal of the project is to use a subset of the MovieLens data set to build a movie recommendation algorithm with a RMSE below 0.86490. The MovieLens dataset has 10 million entries with 7 variables, and the subset of the dataset we are given is a training set known as edx and a test set known as validation. The 7 variables in the MovieLens set are a unique ID for each movie known as movieId, the title, the year it was released, the genres associated with the movie, the unique ID for each user known as userId, a rating between 0 and 5, and a timestamp of the the date and time the movie was reviewed. The edx set uses 90% of the MovieLens and the remaining 10% of the dataset is validation. Interestingly, both edx and validation have 6 variables instead of the 7 that the MovieLens dataset has because the release year variable was added onto the end of the movie title.

To make a movie recommendation algorithm, variables such as the movie being reviewed or the time the movie was reviewed were taken into account. The first variable analyzed was the movie, then the user, then the genres and finally the week the review was written. To prevent movies, users, genres, or weeks with low review counts from heavily skewing the effects, they are all regularized. Both the normal and regularized effects will be shown, along with their effects on the RMSE.

Methods/Analysis

Before showing the movie effect, there is a simpler and easier alogrithm that can be made to estimate the rating, guessing a single value for all the ratings, regardless of the movie being reviewed or the user reviewing the movie. To minimize the RMSE as much as possible, the average value of all the movie ratings will be used, The average value of the ratings, which will be called mu from here on is 3.5124652, and the RMSE of this algorithm is 1.0612018.

From experience we all know that some movies are more enjoyable to watch than others, and that some movies are even widely regarded as great movies or flops, meaning that some movies have a higher base rating than others. This will be referred to as the movie effect, also known as movie bias or bi in this report. To calculate the movie bias, the movies are grouped together by their movieId, have mu subtracted from their rating and then are averaged. The RMSE of this algorithm is 0.9439087, which is an improvement over just using mu but could still be improved upon.

To see how well the movie effect algorithm worked, a good place to start is to see which movies have the biggest and lowest movie effect, shown below.

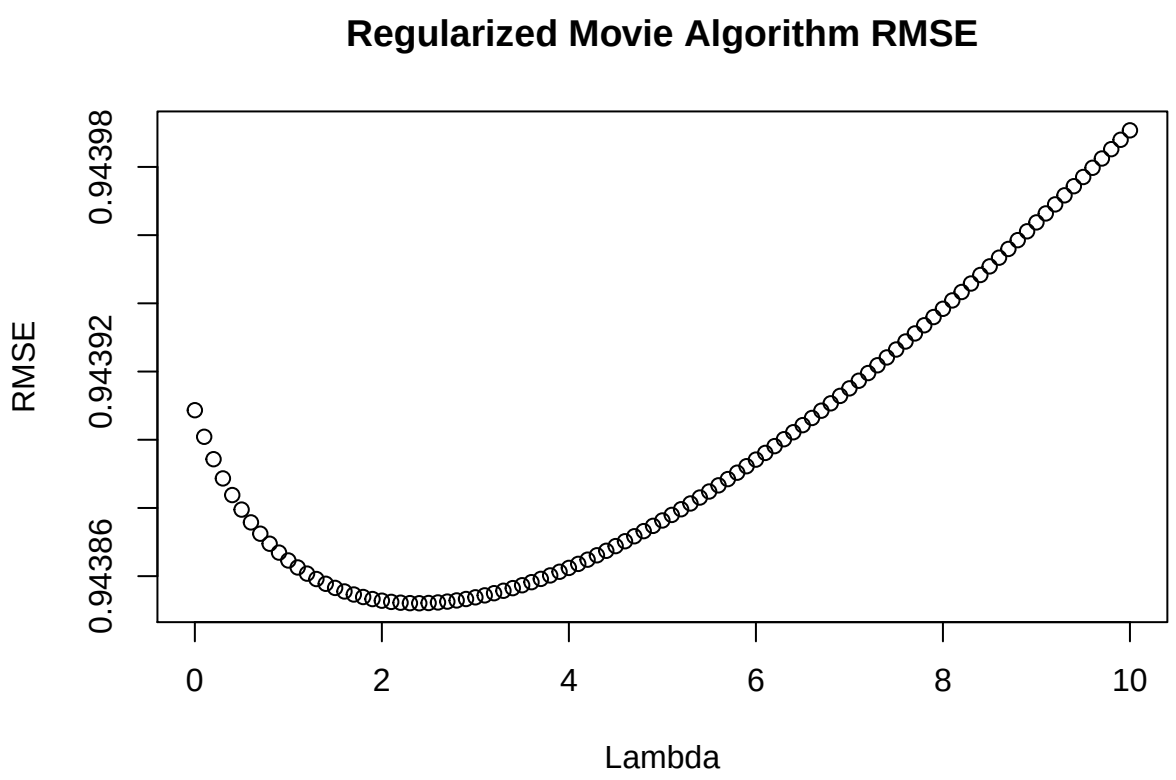
```
##                                                                 title
##  1:                                     Hellhounds on My Trail (1999)
##  2:                                                     More (1998)
##  3:  Valerie and Her Week of Wonders (Valerie a tÃ½den divu) (1970)
##  4:                                   Kansas City Confidential (1952)
##  5:                                 Shawshank Redemption, The (1994)
##  6:                       Red Desert, The (Deserto rosso, Il) (1964)
##  7:                                           Godfather, The (1972)
```

```
##  8: Man Who Planted Trees, The (Homme qui plantait des arbres, L') (1987)
##  9:                                                Usual Suspects, The (1995)
## 10:                                                   Schindler's List (1993)
##            bi     n
##  1: 1.4875348     1
##  2: 1.2018205     7
##  3: 0.9875348     1
##  4: 0.9875348     1
##  5: 0.9426660 28015
##  6: 0.9042015     6
##  7: 0.9029008 17747
##  8: 0.8875348     5
##  9: 0.8533885 21648
## 10: 0.8510281 23193
```

```
##                                                                    title
##  1:                                                   Schindler's List (1993)
##  2:                                                Usual Suspects, The (1995)
##  3: Man Who Planted Trees, The (Homme qui plantait des arbres, L') (1987)
##  4:                                                    Godfather, The (1972)
##  5:                              Red Desert, The (Deserto rosso, Il) (1964)
##  6:                                         Shawshank Redemption, The (1994)
##  7:        Valerie and Her Week of Wonders (Valerie a tÃ½den divu) (1970)
##  8:                                       Kansas City Confidential (1952)
##  9:                                                            More (1998)
## 10:                                         Hellhounds on My Trail (1999)
##            bi     n
##  1: 0.8510281 23193
##  2: 0.8533885 21648
##  3: 0.8875348     5
##  4: 0.9029008 17747
##  5: 0.9042015     6
##  6: 0.9426660 28015
##  7: 0.9875348     1
##  8: 0.9875348     1
##  9: 1.2018205     7
## 10: 1.4875348     1
```

A lot of these movies are unheard of. The reason these movies have such big positive and negative movie bias is because they have a very low number of reviews, labeled n in the diagram above. This is important because it implies that when the mean of the rating of these movies is taken that it doesn't bring them closer to zero as it does with movies with lots of reviews. In other words, the more a movie is reviewed, the less likely it's mean rating is to be as extreme. To fix this, the movie effect must be regularized, which means adding a number, known as lambda, to the denominator of the mean, meaning that it will be the sum divided by the total amount plus lambda instead of just the sum divided by the total amount.

A simple example of using regularization is using lambda = 3. Doing this results in an RMSE of 0.9438538. However, lambda is an optimization parameter which means it can be changed to reduce the RMSE. To find the lambda that lowers the RMSE the most, a graph with lambda going from 0 to 10 by increments of 0.1 will be shown on the x-axis with the possible RMSEs shown on the y-axis. Each point on the plot will represent the RMSE obtained by the algorithm when it was using said value of lambda. Such a plot is shown below, and from it we find that the optimal value of lambda is 2.4 which gives us a RMSE of 0.9438521.



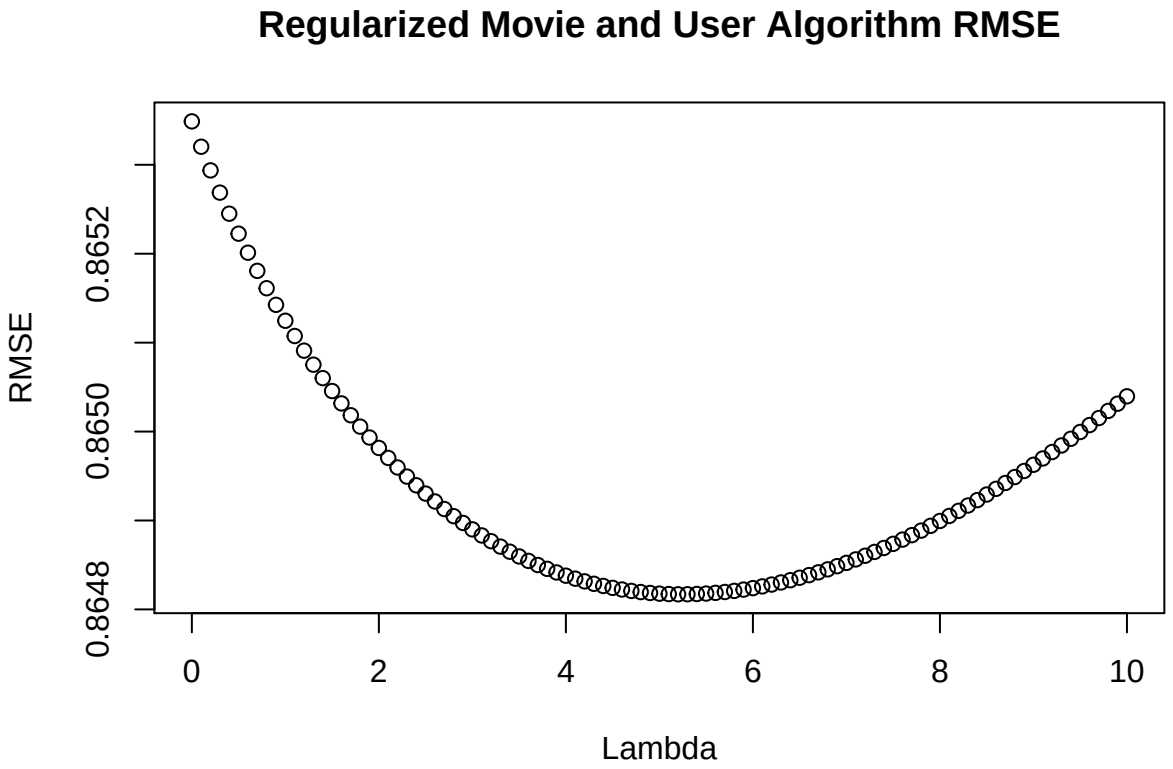
Going back to what was stated earlier, regularization of the movie effect should fix the movies that have the biggest bi. This can be seen below.

```
##                                           title        bi     n
##  1:              Shawshank Redemption, The (1994) 0.9425852 28015
##  2:                         Godfather, The (1972) 0.9027787 17747
##  3:                                More (1998) 0.8949727     7
##  4:                     Usual Suspects, The (1995) 0.8532939 21648
##  5:                        Schindler's List (1993) 0.8509400 23193
##  6:                             Casablanca (1942) 0.8077860 11232
##  7:                           Rear Window (1954) 0.8059426  7935
##  8: Sunset Blvd. (a.k.a. Sunset Boulevard) (1950) 0.8027550  2922
##  9:                           Third Man, The (1949) 0.7983147  2967
## 10:                        Double Indemnity (1944) 0.7974633  2154
```

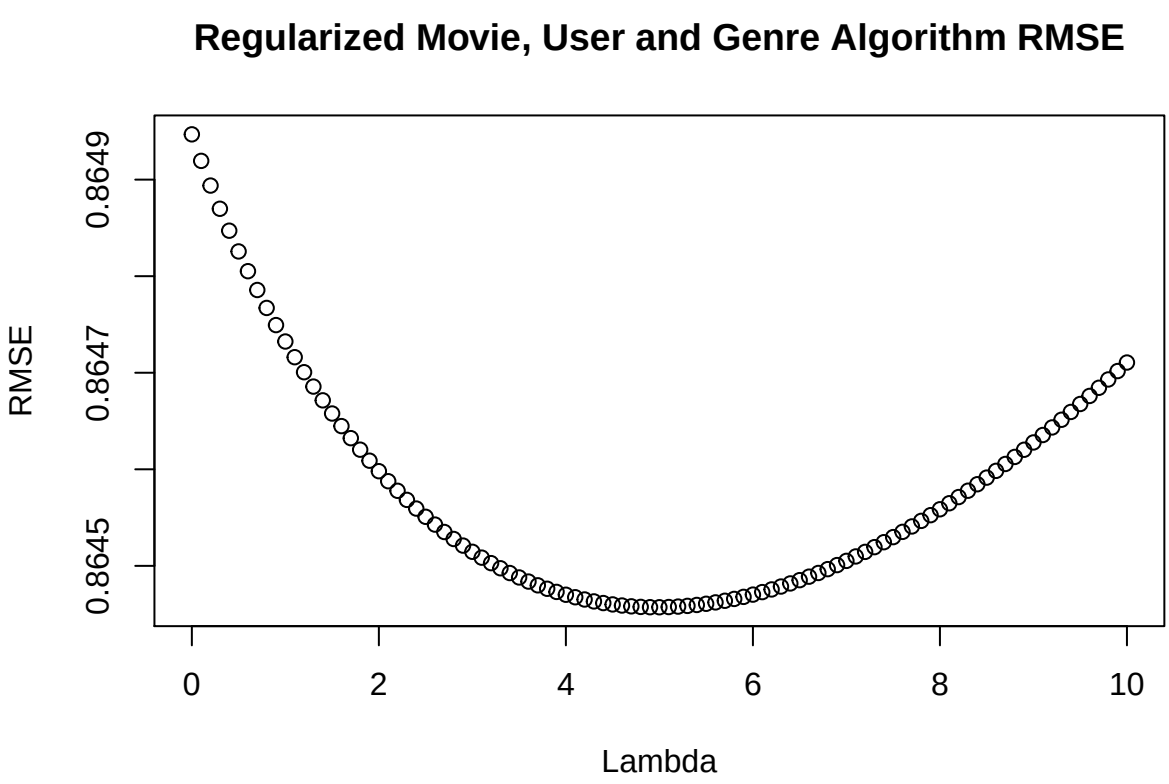
```
##                                                title        bi   n
##  1:                   SuperBabies: Baby Geniuses 2 (2004) -2.606131  56
##  2:                           From Justin to Kelly (2003) -2.579347 199
##  3:                                Disaster Movie (2008) -2.467991  32
##  4:                              PokÃ©mon Heroes (2003) -2.440515 137
##  5:                    Carnosaur 3: Primal Species (1996) -2.341586  68
##  6:                                       Glitter (2001) -2.320521 339
##  7: Pokemon 4 Ever (a.k.a. PokÃ©mon 4: The Movie) (2002) -2.306839 202
##  8:                                       Gigli (2003) -2.301527 313
##  9:                         Barney's Great Adventure (1998) -2.298445 208
## 10:                Faces of Death: Fact or Fiction? (1999) -2.222235  58
```

These movies are a lot more popular and known, though there are some movies in the tables that still have a low number of reviews. But overall, this seems more believable and in line with what a person would expect of a movie with generally higher or lower ratings.

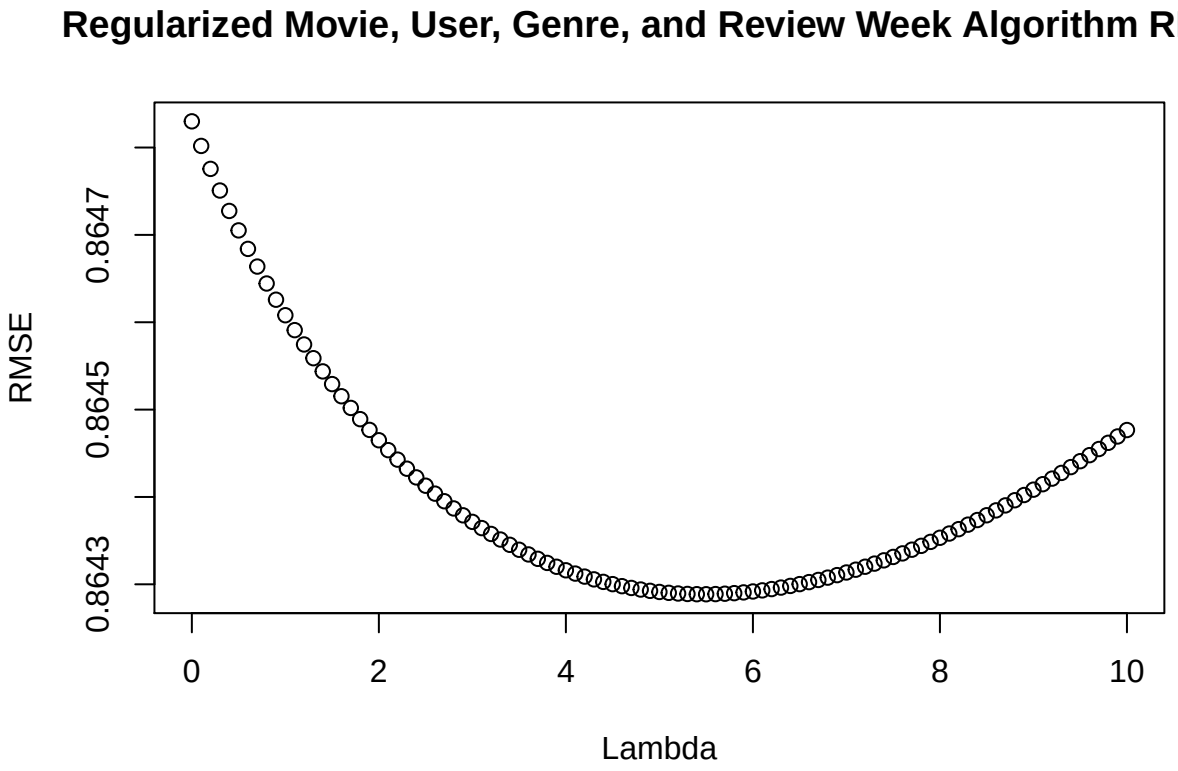
On top of using the effect of each movie, we can see how users reviewed other movies to try and optimize the code. To do this we group the movies by userId and subtract the mu and bi before taking the mean. This gives us a RMSE of 0.8653488. This number is lower than the RMSE of the movie effect, but is not lower than the regularized movie effect RMSE. To fix this, regularization can be done on the user effect as well. Similar to before, the optimal lambda can be found using a plot with the same lambdas and the RMSEs. From this can be seen that the lambda that reduces the RMSE the most is 5.2, which brings the RMSE down to 0.864817.



Another thing that could be used to increase the accuracy of the algorithm is finding out if there is a genre bias and if it can reduce the RMSE. The genre bias was added to the algorithm in a comparable way to how user effects were added to the algorithm. This resulted in an RMSE of 0.8649469. Once again, higher than the previous regularized algorithm but lower than the non-regularized version. The same values of lambdas are used to optimize the algorithm, with it being shown below. The graph below shows that the best lambda is 5 which brings the RMSE down to 0.8644572.



Something else that could be used to reduce the error of the algorithm is seeing if when a movie is reviewed has an effect on it's rating. To do this however, the timestamp variable in the dataset must be changed to a date. From here, it was changed from a date to the first Monday of the week to allow for more data points per entry. Because of this it might be more appropriate to think of it as the week a user reviewed a movie rather than the date or time a movie was reviewed. Adding the review week effect onto the previous non-regularized algorithm returns an RMSE of 0.86483. Just like before, this has a lower RMSE than the non-regularized algorithms but a higher RMSE than the regularized ones. This effect can also be regularized the same way the rest were and shown on a graph as well. The graph shows that the optimal lambda for reducing RMSE is 5.5, and that the lowers the RMSE to 0.8642886.



Results

```
##                                                                 method      RMSE
## 1                                                    Just the Average 1.0612018
## 2                                                    Movies Algorithm 0.9439087
## 3                                  Regularized Movie Algorithm, l = 3 0.9438538
## 4                                Regularized Movie Alogrithm, l =  2.4 0.9438521
## 5                                          Movies and Users Algorithm 0.8653488
## 6                         Regularized Movie and User Algorithm, l =  5.2 0.8648170
## 7                                  Movies, Users, and Genres Algorithm 0.8649469
## 8                  Regularized Movie, User, and Genre Algorithm, l =  5 0.8644572
## 9                      Movies, Users, Genres, and Review Week Algorithm 0.8648300
## 10 Regularizaed Movie, User, Genre  and Review Week Effect, l =  5.5 0.8642886
```

The table above lists the RMSE of the algorithm as more variables were added and then regularized. From it, it can be seen that having the algorithm that regularized the movie, the user, and the genre of the movie was enough to achieve the goal of a RMSE below 0.86490. The additional variable and regularization of it were done to further guarantee that the RMSE of the algorithm was safely below the desired 0.86490. The table also shows that as expected, adding more variables to the algorithm helps improve it's accuracy, and that regularization further improves the accuracy. It is interesting to note that adding one more variable would have a similar RMSE to regularizing the previous amount of variables.

However, a big problem in the model arose as more effects were being added and then calculated for regularization, and it was the time it took the algorithm to run. Each variable was adding a lot more time to the calculations than the previous one had added, meaning that it was becoming more time consuming to run the algorithm as more effects were being added on. If the effect wasn't regularized, it wouldn't add

much more time. However, when they were regularized, it appeared to nearly double the time it took for the algorithm to make it's predictions. It wouldn't take very long with one or two effects, but became noticable once three or more are added. More effects could have been added to reduce the RMSE but concerns over how long it would take for the algorithm to make a prediction were why they weren't added.

Conclusion

Regularizing the movie, user, genre and review week effects were all that were needed to bring the RMSE below 0.86490. The regularization of the effects takes the longest, but tends to result in a bigger decrease in RMSE than adding one more variable to take into effect in the algorithm. Computation time was the limiting factor, where it would take a long time to add more variables to analyze and see if they were running and being analyzed. An example of a problem would be not realizing a pipe was missing between two functions in the last part of an algorithm where the effects were being regularized which resulted in a long wait time to not get a result, and then poking around in the code to see what the possible problem was. Often times this was due to missing something in the code or writing a variable wrong, which became easier and easier as the project went on. My own coding skill was a limitation too, with more optimal coding I feel as if I could have added in release year to the algorithm. It was originally added as the third variable to be analyzed but it took longer to run than the final algorithm with the regularized movie, user, genre and review week effect. It also introduced a problem that led to me realizing that movie titles were not unique and had to tie the release year to movieId instead of title if I intended to use and attach it to another data set. Additionally, making a function out of the average rating per review date might have been more optimal than just using review weeks, but was not something I knew or remembered to do. A lot more information could have been taken from the timestamp variable, such as the season the review was written, the day it was written, or if a day it was written was a holiday, which could have all provided interesting insight and possibly increased the accuracy of the algorithm. In future works, more time will be spent on cleaning up datasets to make sure just the necessities will be in them, such as release year or genres to movieId in datasets pertaining to those rather than keeping all if not all of the dataset. Checking what each variable in the dataset represents will also be done at the start so that there aren't confusions or wrongful assumptions about what each means.