MovieLens Project

Nathan Patterson

October 7, 2019

Contents

Introduction **1**

Analysis **1**

Results **5**

Conclusion **6**

Introduction

Creating movie recommendation systems has become a standard initiation challenge for data science newcomers. The first HardvardX Data Science Capstone project relies on the MovieLens 10M data set for this challenge. The 10M data set contains ratings and related data submitted to the movie recommender service MovieLens; totaling ~10M ratings of ~10k movies submit by ~71k users.

This report details the development of a weight based approach for creating a movie recommendation system based on the 10M data set. The data set is initially processed to separate genre and date information, allowing for calculation of desired metrics and for data visualization. Weights are calculated, their effects visualized, and are added to the equation for predicting ratings. Finally, the RMSE values are calculated and compared for different weight-based predictions on the validation data set.

Analysis

Working with the Initial Data set

Using code supplied through the HarvardX project page, the DataLens 10M data set is downloaded and separated into two data frames; the 'edx' data frame, which is intended for training and testing, and the 'validation' data frame, which is for validation and final RMSE calculation. Following the initial import, the two data frames supply the same data columns, but different entries, as seen from examining the structure of the edx data frame:

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 8389..
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)"..
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Dram"..
```

The imported data set illustrates a common problem with large data sets, columns that contain multiple values which are independently important. Specifically, (1) genres are all combined into a single column and (2) movie titles and release years are combined in a single column.

The current analysis first addresses problem (1), separating the genre information into separate encoded columns. The function 'OneHotGenres' in the accompanying R script separates the entries in the combined column and fills in columns with 1's (if the movie lists a genre) and 0's (if the movie is not included in a genre). The columns with this data are then added to the original 'edx' data frame and used to initialize the 'edxOneHot' data frame. The original genres column is kept as part of this new data frame so that a comparison of the impact of creating weights for individual genres and the impact of using combined genres can be made.

Release years are appended to the end of movie titles which can also contain numbers (problem 2), however, the release year is specified within parentheses at the end of each entry. This means that by using regex and string extraction combinations, the four digit year within parentheses can be extracted. The strings with years surrounded by parentheses are initially extracted from the title-year string and then a second string extraction obtains only the four digit year and converts it to a numeric value. After separating genre and release year information from combined columns, the new 'edxOneHot' data frame is created and has the following structure:

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    9000055 obs. of  28 variables:
##  $ userId             : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId            : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating             : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp          : int  838985046 838983525 838983421 838983392 8389..
##  $ title              : chr  "Boomerang (1992)" "Net, The (1995)" "Outbr"..
##  $ genres             : chr  "Comedy|Romance" "Action|Crime|Thriller" "A"..
##  $ Action             : int  0 1 1 1 1 0 0 0 0 1 ...
##  $ Adventure          : int  0 0 0 1 1 0 0 1 1 0 ...
##  $ Animation          : int  0 0 0 0 0 0 0 0 1 0 ...
##  $ Children           : int  0 0 0 0 0 1 0 1 1 0 ...
##  $ Comedy             : int  1 0 0 0 0 1 1 0 0 1 ...
##  $ Crime              : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ Documentary        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Drama              : int  0 0 1 0 1 0 1 0 1 0 ...
##  $ Fantasy            : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ Film-Noir          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Horror             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Musical            : int  0 0 0 0 0 0 0 0 1 0 ...
##  $ Mystery            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Romance            : int  1 0 0 0 0 0 1 1 0 0 ...
##  $ Sci-Fi             : int  0 0 1 1 1 0 0 0 0 0 ...
##  $ Thriller           : int  0 1 1 0 0 0 0 0 0 0 ...
##  $ War                : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ Western            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ (no genres listed) : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ releaseYear        : num  1992 1995 1995 1994 1994 ...
##  $ movieRatingsCount  : int  2178 13469 14447 17030 14550 4831 31079 3612..
##  $ movieRatingsPerYear: num  121 896 962 1063 908 ...
```

Creating Weights for Prediction

The weight-based approach detailed in the course text ('Introduction to Data Science', R.A. Irizarry, 2019) serves as the basis for the current approach. Specifically, the rating average ($/mu$), Movie-based (b_i), and User-based (b_u) prediction weights are initially calculated and used as follows for rating predictions:

$$Prediction = \mu + b_i + b_u$$

Following the approach in the course text, the Movie-based (b_i) and User-based (b_u) weights are calculated using the following equations:

$$b_i = \sum_{u,i}^{n} (rating_{u,i} - \mu)/(\lambda + n_i)$$

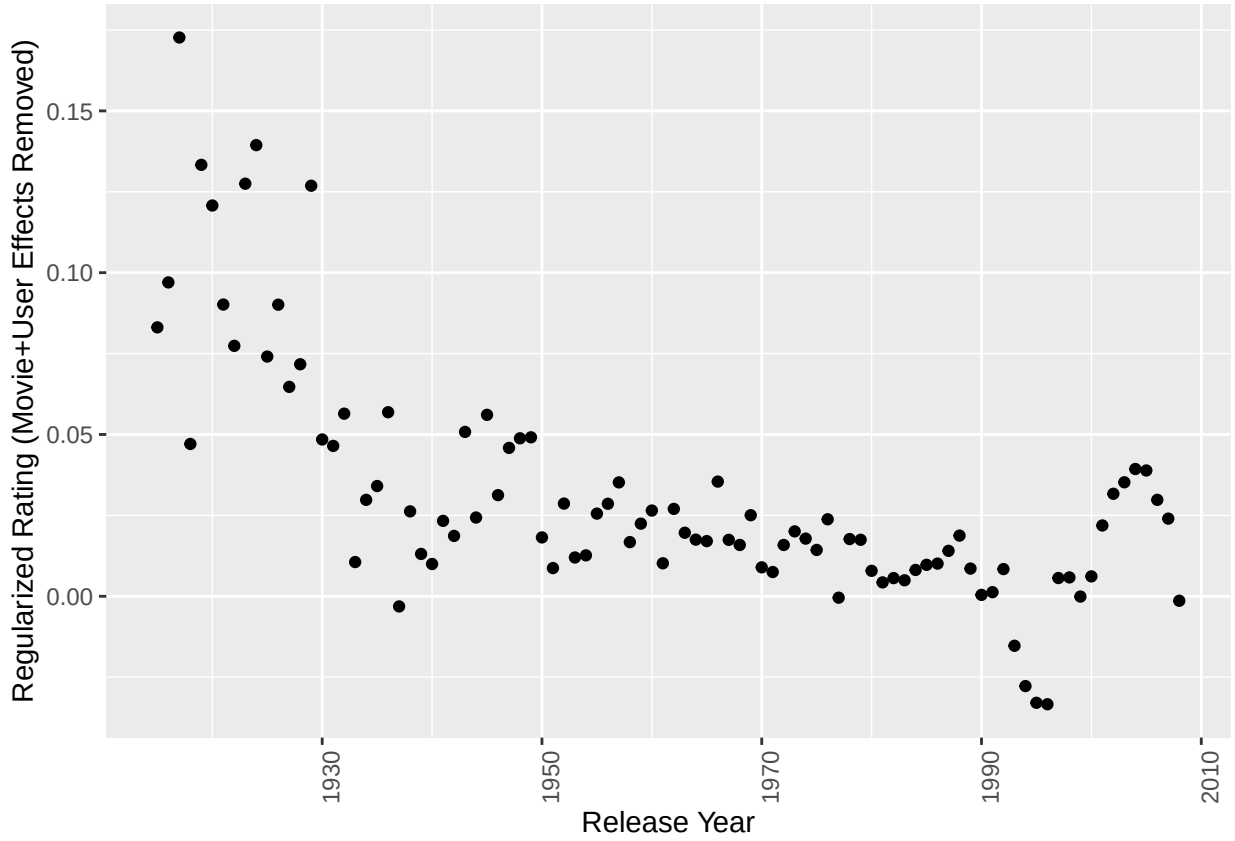
$$b_u = \sum_{i,u}^{n} (rating_{i,u} - b_i - \mu)/(\lambda + n_u)$$

Determining additional weights

Additional Year- and Genre- weights were chosen after examining the effect of each variable on the remaining score distributions. To visualize their effects, regularized rating by year (b_y) is calculated using the following equation:

$$b_y = \sum_{y}^{n} (rating_y - b_i - b_u - \mu)/(\lambda + n_y)$$

The relationship between the regularized rating by year can then be visualized.



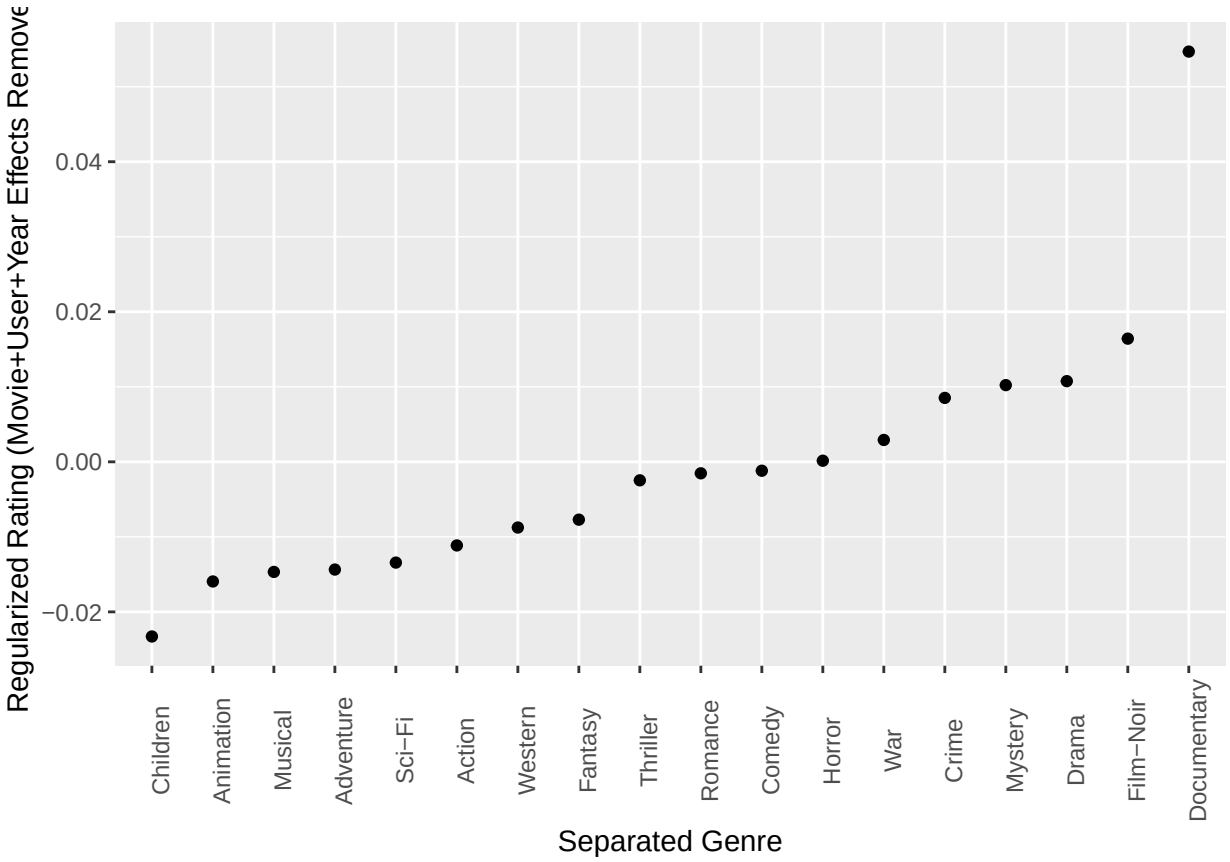
While the relationship maxes out with older movies at ~0.15, the obvious time-based trend still suggests it is worthwhile to include this weight in the prediction equation, which becomes:

$$Prediction = \mu + b_i + b_u + b_y$$

As a Sci-Fi movie fan that dislikes romantic dramas, it seems obvious that there should be important rating-relevant data within the genre/rating distributions. While extracting movie preferences (which varies by user) may require an approach that involves dimensionality reduction, two relatively simple weights are calculated. To determine the weights for different genres, the following equation is used:

$$b_g = \sum_{g}^{n}(rating_g - b_i - b_u - b_y - \mu)/(\lambda + n_g)$$

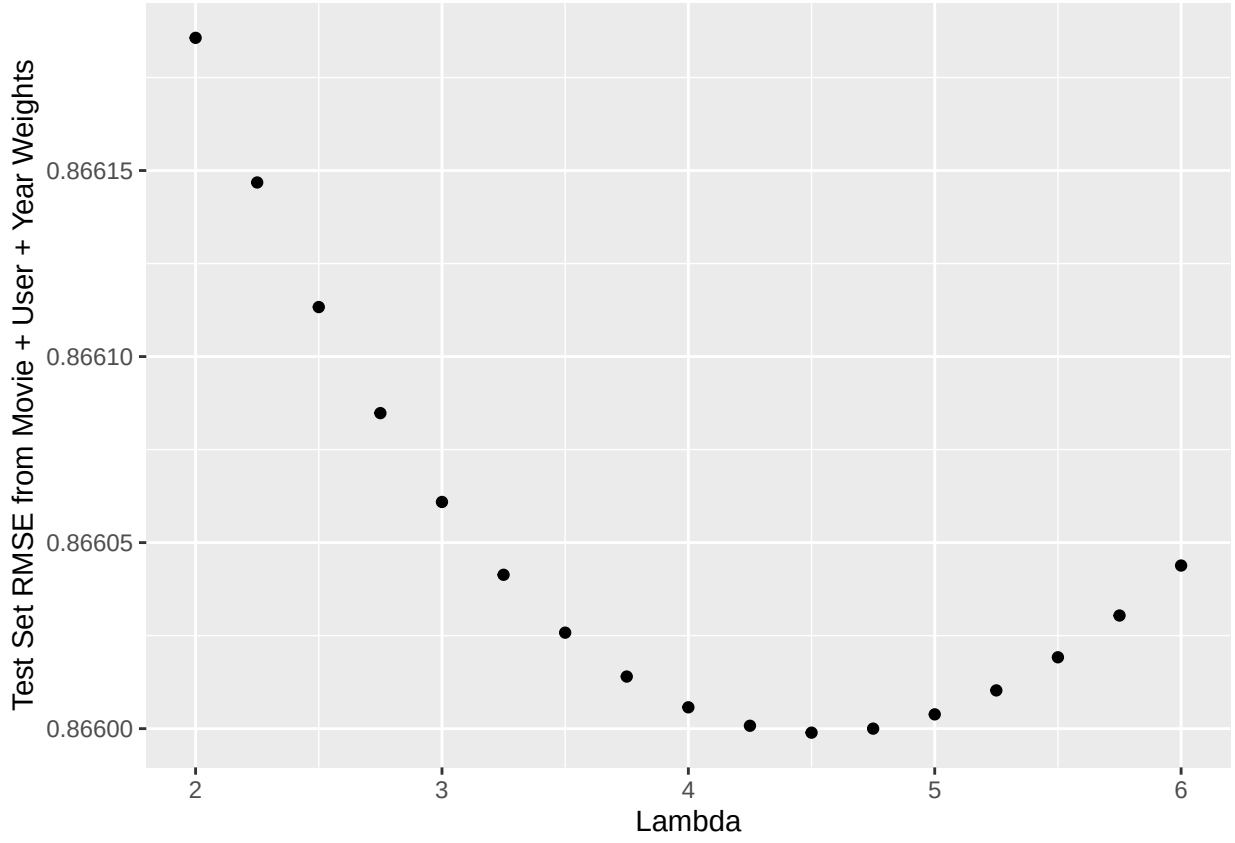
This results in a set of 797 different weights for the original combined genre categories and 18 different weights when genre categories are separated. However, because many movies fit into multiple genres, the 18 genre weights are combined to fit each movie. For example, the movie 'Stargate' fit into the Action, Adventure, and Sci-Fi genres so the three weights from the set of 18 were simply added together. While the plot to visualize the modified/reduced ratings for the 797 combined genres can be difficult to read, the separated data shows genre preferences across the the entire user-base.



The Movie and User weights are then supplemented with two new weights that focus on Movie release year (b_y) and Genres (Combined b_{gc} or Separated b_{gs}). This combination of weights results in the following prediction equation.

$$Prediction = \mu + b_i + b_u + b_y + b_{gc_{or}gs}$$

Regularization for each of the weights involves the parameter λ. This value assists in regularization by reducing the effect of outliers that would otherwise not have a large denominator value; such as a rarely reviewed movie with a 5 rating or the early 1900s when new movies were themselves quite rare. To determine the optimal value of λ, the approach detailed in the course text was extended to encompass the weights for Movies (b_i), Users (b_u), and Release years (b_y). The λ value chosen based on this analysis was $\lambda = 4.75$ despite the lower value for $\lambda = 4.5$ in the following plot. This value oscillated between the two values depending on the randomly sampled test set. Because of the two distinct and separately implemented genre-related weights, calculation of the optimal λ value was carried out without the genre weights.



Results

Following the creation of weights for Movie, User, Year, and Genre data, the weights are used to calculate a series of predictions on the 'validation' data set. Using a series of Left Join commands, the weights calculated for the 'edx' data set are appended to the 'validation' data set. Predictions are made using the aforementioned Prediction equations and the resulting RMSE values are output in the following table:

```
##                                                       RMSE
## Mean Rating                                      1.0612018
## Movie Weights                                    0.9438724
## Movie + User Weights                             0.8648201
## Movie + User + Year Weights                      0.8645223
## Movie + User + Year + Genre Weights (Separated Genres) 0.8644690
## Movie + User + Year + Genre Weights (Combined Genres)  0.8642527
```

While the Movie-based (b_i) and User-based (b_u) weights each provide ~10% progressive improvement in the RMSE values, the other weights provide considerably less benefit. While the Release year (b_y) and Genre (Combined b_{gc} or Separated b_{gs}) weights do provide improved predictions, they each provide $< 0.1\%$ improvement over the previous weight combination.

Conclusion

Creating a movie recommendation system for the MovieLens 10M data set poses an interesting challenge. The development of a weight-based approach similar to the method detailed in the text is still a time-consuming and challenging task given the size of the data set. Despite the initial view that the combined genre groupings was a problem with the dataset, the combined genres provided a superior set of weights for prediction than those calculated for separated genre information.

The final prediction equation which provides the lowest RMSE is:

$$Prediction = \mu + b_i + b_u + b_y + b_{gc}$$

The calculated weights for Movies (b_i), Users (b_u), Release years (b_y), and Combined Genres (Combined b_{gc}) resulted in a final RMSE of:

$$RMSE_{Movie,User,Year,Genre} = 0.8642527$$

The route taken to create this analysis was embarrassingly complex. Initial attempts involved training KNN, Random Forest, Neural Network, and other models based on additional metrics that were formulated. These methods provided ~15% improvement over using a single mean/average value, but modifying and retesting each approach led to extremely long wait times for model training so it was eventually abandoned. A second round of attempts involved skipping over the weight-based approach and leveraging PCA to predict principal components which were then used to train Random Forrest models for prediction of ratings. This approach also provided ~10-15% improvement over using a single mean/average value, but the wait times associated with training and predicting these models also led it being eventually abandoned.

The success of the relatively simple weight-based approach over more complex approaches led to an incredibly useful insight; start simple and build up complexity if needed.