MovieLens Recommendation System

Kaylee Robert Tejeda

9/13/2021

Introduction and Overview

One common application of machine learning is the creation of recommendation systems. These are mathematical models that take as input parameters associated with a particular observation to give as output the value of an other parameter associated with the observation as a prediction. The more accurate this prediction is, the more useful and valuable it is in specific industries. These predictions are used to make recommendations to users based on past behavior and the behavior of similar users under similar circumstances. For example, an online video distribution network might use such a recommendation algorithm to create lists of suggested movies based on what is currently being watched by a particular user.

Goal Summary

Our goal is to create a recommendation algorithm from a standard set of data. Our data set provided by GroupLens is known as the "10M version of the MovieLens dataset", which contains 5 base variables per observation. We will measure the accuracy of our model by comparing the ratings predicted for a subset of our data to the actual ratings for those observations, and calculate the root mean square error (RMSE) for the final set of predictions. An RMSE less than 0.86490 will indicate a successful model for the purposes of this report.

Data

The MovieLens data set contains 10000054 rows, 10677 movies, 797 genres and 69878 users. Each observation in the data set has the following variables associated with it:

- `userId`: Unique user identification number.
- `movieId`: Unique movielens movie identification number.
- `rating`: User-provided ratings on a 5-point scale with half-point increments starting at 0.5
- `timestamp`: Time of user-submitted review in epoch time,
- `title`: Movie titles including year of release as identified in IMDB
- `genres`: A pipe-separated list of film genres

Preparation

The main data set is split into a working set called `edx` and a second set representing 10% of the original set, called `validation`, which will be only used for checking the model once it has been created, and will not be used at all during the model formation process.

The resulting working set `edx` is further split into a training set and a test set which will be used to evaluate various methods. The test set comprises of 20% of the `edx` working set. No attempt is made to ensure that all the users and movies in the test set are also in the training set. This leads to a fortunate insight later, and thus has been included as part of the discovery process.

Analysis

Outline of Steps Taken

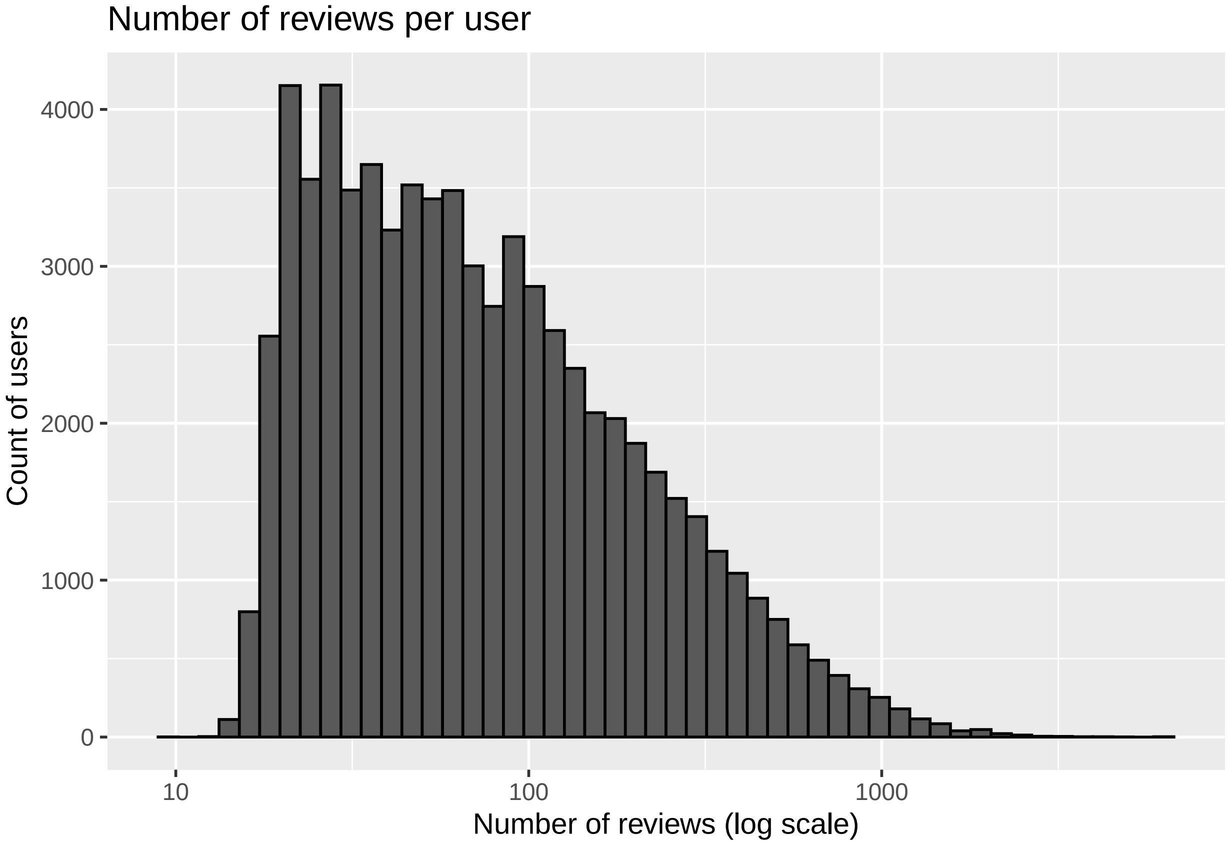
1) Baseline Naive-RMSE
2) Movie Effect Model
3) Notice `NAs` in predictions, need to adjust somehow (leads to an insight that is worth mentioning)
4) Movie and User Effects Model with fixed penalization factor lambda.
5) Penalized User & Movie effect with lambda optimized to the hundredth's place (overkill, but interesting).
6) Simplified Penalized Effect method with integer lambda and manual outlier pruning.
7) Simplified Penalized Effect method with integer lambda and statistical outlier pruning.
8) RMSE now in an acceptable range, test against validation set.

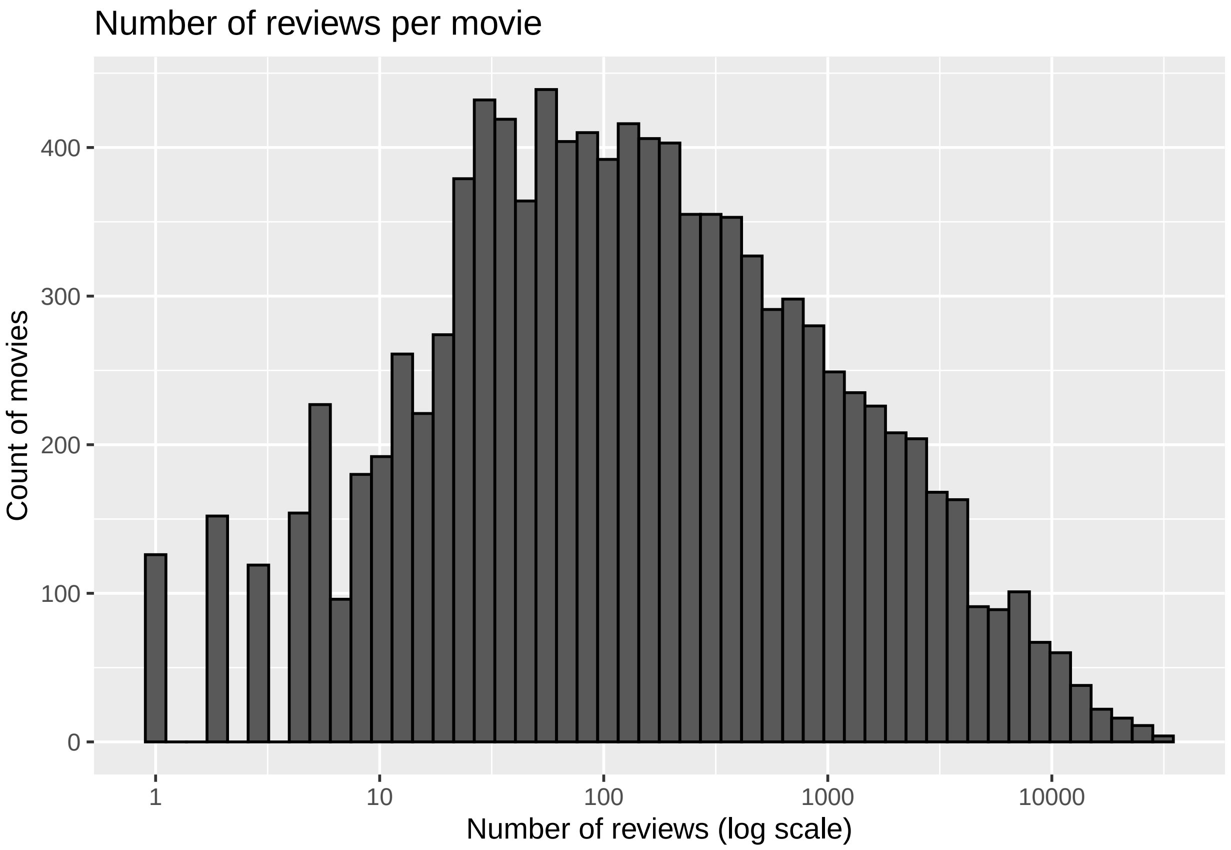
Exploration

Baseline Naive-RMSE Just to get a baseline, a first attempt is made by calculating the arithmetical mean of all movie ratings in the test set, and that is used as a prediction for all movie ratings. This is not a very good approach, but it will give us a worst-case error, of sorts.

Movie Effect Model The next refinement to our approach is to account for a movie bias effect, since some movies naturally get higher ratings than others. This is when `NA` values start showing up in the predicted ratings. To get past this error for now, the option `na.rm=TRUE` is added to the `mean()` part of the loss function. This effectively throws out any observation that does not have a matching user ID or movie title in the training set. This is only a temporary fix and merits a closer look.

```
## [1] "There are 45 NA values in our predictions, which need to be removed."
```





Not very many `NA` values are being generated. This is most likely due to movies and/or users in the training set that are not present in the test set, or vice versa. This was avoided in the original data set by using the `inner_join` function. Instead, we attempt to remove any movies or users with extremely low rating counts. The assumption is that movies with only a few ratings affect the mean and therefore the error without contributing much to the overall effect. The same could be said for users who have only rated a few movies. Removing these low-frequency observations prevents `NA` values in the predicted ratings and allows the RMSEs to be calculated. A more rigorous approach needs to be formulated as this is only a temporary workaround.

Move and User Effect Model Similar to the Movie Effect model, user bias can also be accounted for. The naive average approach is biased for each movie as well as each user, since certain users tend to rate higher or lower than the average user.

Penalized Movie Effect model. Now that we have developed a model that performs better than the Naive RMSE approach that we began with, we can start to regularize the effects by penalizing with a parameter we will call lambda. We start by penalizing both the movie and user effects with the penalization factor lambda arbitrarily set to 3.

Step-Wise Cross-Validation with lambda optimized to two decimal places. Iterated cross-validation can be used to quickly optimize the penalization factor lambda to two decimal places of precision. First, lambda is optimized to the nearest integer, then it is optimized again to the nearest tenth, and then again to the nearest hundredth. This takes 30 tests, instead of the 100 that would be needed to check all values from 1 to 10 at 0.01 increments.

Insights

- `NAs in predicted_ratings`: A failure to use the proper join function inadvertently lead to `NA` values in the predictions, which prevents the proper calculation of an RMSE. Attempts to rectify this leads to an investigation of outliers that helps reduce the overall RMSE.

- `lambda can be refined step-wise`: Lambda can be optimized to two decimal places using 30 tests instead of 100. This ultimately proved to be worthless, but is interesting. The final RMSE does not seem to depend greatly on lambda to a high degree of precision. Optimizing lambda to the integer level seems sufficient and takes 1/3 the computational time.

Final Modeling Approach

Ultimately, what helps more than fine-tuning lambda is to throw away the outliers that are theoretically adding more noise to the mean than they contribute to the predictive power of the algorithm. Hard-coded values are tested, starting by removing any movie with three or fewer ratings submitted, and removing any user that has reviewed nineteen or fewer movies.

```
##                                                         method      RMSE
## 1         Naive method using average rating of 3.51257353601916 1.0607079
## 2                                            Movie Effect Model        NA
## 3                               Movie Effect Model, na.rm=TRUE 0.9437144
## 4                                 Movie and User Effects Model 0.8661625
## 5                         Penalized Movie Effect Model, lambda=3 0.8656010
## 6 Optimized Penalized Movie & User Effect Model, lambda = 4.68 0.8655424
## 7                             OPM&UE Model, integer lambda = 5 0.8655443
## 8                     Hard-Coded Pruned OPM&UE Model, lambda = 5 0.8646404
```

The slight reduction in RMSE is an unexpected surprise. The motivation for pruning out the lowest number of ratings per movie and per user was to eliminate `NA` values from the predictions, since that arose from either movies or users with a low count winding up in the training set but not the test set. It was assumed that if every movie has been rated more than three times, and if every user had rated at least nineteen movies, then the possibility of either not being represented in the test set was much lower. It was realized after the fact that this can be controlled during the partition creation by using `semi_join`, as was done with the original data set. Nonetheless, this attempt at removing the `NA` values results in an unexpected reduction of the RMSE value, and merits further investigation.

Rather than hard-code arbitrary values, a statistical approach based on ignoring the lowest 10th percentile of both movies and users (in terms of numbers of ratings associated with each unique value) can be used.

10% seems like a good number to start with, although this should be refined more in the future, and is itself an arbitrary cutoff point not based on any deeper investigation or theory.

Results

The fine tuning of lambda beyond the integer level does not seem to provide enough reduction in error to justify the increased cost in execution time. Instead, the statistical approach to pruning lower outliers seems to provide the needed optimization. A simple cutoff of the 10th percentile and integer lambda yields the following:

```
## [1] "Movies below the 10th percentile with fewer than 10 ratings will be ignored."
```

```
## [1] "Users below the 10th percentile with fewer than 22 ratings will be ignored."
```

```
## [1] "Penalized User & Movie Effect method using a mean rating of 3.51181194815616 ."
```

```
## [1] "Optimized lambda = 5 gives an RMSE of 0.864238188095698 on the test set."
```

Performance

We can measure the performance of this final algorithm against the `validation` set (for the first time):

```
## [1] "Now testing this optimized lambda against the vaildation set."
## [1] "Optimized lambda = 5 gives an RMSE of 0.863741988998773 on the validation set."
```

Conclusion

Our penalized movie and user effect model gives an error in the desired range when lower outlier pruning is applied at the 10th percentile of ratings for both users and movie titles.

Summary

We began with a linear estimates accounting for both movie and user effect as well as a penalized term for both effects. This got us close to our goal, and further exploration was warranted. The original mistake of not using `semi_join` when generating the training and test sets lead to an investigation of low-value outliers.

Our model is based on the mean rating for each movie across user ID, which is NOT resistant to outliers. Hence, the users and movies with low frequency add noise to the mean without adding much weight to the predictions. an attempt to remove these (to eliminate `NAs` in the final predictions) inadvertently lead to a reduction in RMSE, which is what makes this method powerful.

Limitations

This seems to depend on large data sets. Eliminating below a certain percentile might work against you with a smaller data set. This needs to be explored.

Future Work

Optimizing the percentile removed from each predictor (`movieId` and `userId`) would be interesting, as it is unlikely that the 10th percentile is always the ideal cutoff. It would be nice to keep as much of the original data set as possible while still training a good solid model.

Many predictors were not considered, such as timestamps or genres. No factor importance or influence analysis was done of any kind. These could be included in the model to decrease the overall error further if needed.