Econometrics B: Letterboxd Collaborative Filter Recommendation System

Dan Ehrlich and Johnny Ma

May 2, 2017

1 Introduction

The Netflix Challenge, beginning in 2006 and ending in 2009, was a challenge to reduce the RMSE of Netflix's rating prediction algorithm by at least 10%. The challenge involved over 500,000 users and 12,000 movie ratings, which was entirely made unidentifiable and was about 99% sparse, given that many users rated only a few movies. Many groups of academics, computer scientists, and statisticians created their own algorithms, some of which are backed up by mathematical theory, and used a training data-set to prepare for the test-data set, with each set being a 50% split by user. There were numerous approaches, ranging from naive averaging to ANOVA, neighborhood weighting, Latent Factor Modeling, and even Neural Networks and Boltzmann machines [1]. Though it seems these approaches were successful, a contemporary look at the implementation of these so called "Filtering and Recommendation Systems" in platforms such as Netflix, Spotify, and other Taste based industries leaves a lot to be desired, especially compared to simple word of mouth recommendations. One wonders why these algorithms seem to be bad at returning shows that one would actually want to watch. Problems with the data are obvious: who, when, and why are rating Netflix movies? Can one map song listening or movie watching behavior to actual preference? Economists like to create models where the user has some decision or input, but the Netflix data has no observations that indicate preferences for genres, time periods, actors, directors, etc. beyond watching behavior. In essence, we seek to remedy these problems by implementing the aforementioned Latent Factor Models using a novel data source that can potentially fill these holes. No doubt many further considerations will arise, and potential to answer more economic questions about taste formation or the influence on exogenous shocks (such as Oscar nominations/wins) will be fruitful directions to take. We measure our success in similar fashion; by using a training data set to run our algorithm, then testing with the second half of the data to reduce the RMSE in rating predictions.

2 Data

2.1 Overview

This project was motivated by a desire to improve upon the Recommendation Systems based on less detailed data such as that using in the Netflix Challenge. A few major problems with the Netflix data is that few people actively rate movies (bias downwards for anger), all user and item characteristics are lost, and there is no ability for the user to indicate preference. This blocks off many user and item-based regressions that can be run to improve the system. Therefore, we searched for a more robust and descriptive dataset of films.

We decided to use Letterboxd.com, a small but vibrant film rating site. movie rating system has additional data that can ameliorate these issues. Letterboxd is a small but growing community of film fanatics, where users can rate movies, post reviews, and follow/like other users and their reviews. Since this is mainly a diary-style rating platform for movie enthusiasts, one can expect less sparsity and more truthful review/rating behavior, reducing omission errors. In addition, a user can identify themselves by inputting their four "favorite movies." These favorites give a lot of information about the user supplied by the user himself. Do they love older films, horror films, Japanese animation, etc. In addition, there is a watchlist that many users have that indicates future desire to watch movies. There is also more item (movie) information for each entry, such as genre tags, runtime, year and country of release, a delineated histogram of ratings (from 0.5 to 5 stars, in 0.5 star steps), and all crew and cast names.

In summary, this data source is much more rich in characteristics for both users and movies, with the added bonus of some user input in their "taste profile". A negative involves having to use Data Scrappers to extract this information, as Letterboxd has no workable API. However, all users are in a large directory sorted by activity, and the site is wonderfully built with easy html tags for all possible data of interest. Overall, a great data source for investigations in rating systems. A sample profile is here https://letterboxd.com/JohnnyMa/.

2.2 Scraping Procedure

The letterboxd.com website is well adapted for webscraping. Much of the film and user info stems from the base "letterboxd.com" URL, and are uniquely identified throughout.

The output we want to obtain from scraping is a data frame with users on the rows, films on the columns, and associated ratings for each user and film. We started by generating a list of users from letterboxd's own user list, found https://letterboxd.com/people/popular/here. This is a list sorted by "popularity", a metric that Letterboxd states is dependent on amount of reviews, likes on reviews, and frequency of publication. Since we are taking from the most popular users, we are likely skewing our population towards more the more film-obsessive than your ordinary film Joe. However, this is potentially good for Recommendation systems, as the data will be less sparse, and these high popularity film reviewers can be seen as "taste makers" who spend a lot of time reviewing and thinking about these films.

Once we have a list of users, we can enter their profiles using their user ID and being to scrape user-related information. This includes their average rating, their number of ratings, and up to 4 of their favorite films. The favorite film data is kept for each of these users and

can be used for a clustering exercise in an extension of this model. In addition, we can obtain all the ratings that each user has given off of this user page, where it is stored in many separate pages. Here is an example of such a page https://letterboxd.com/johnnyma/films/page/1/here. We get a movie title and rating, which we can later merge into our final data frame.

Next we turn to the films themselves. We obtain a list of films from another convenient page, https://letterboxd.com/films/here, that lists films by their popularity. Again, the same problems or blessings that the user list has due to it being sorted by popularity. Many larger production and more popular films have many more reviews, while most obscure movies hardly have any, especially for such a small site. Something to note in that this page is populated using a JavaScript AJAX process, something that the scraping method had to deal with in a special and time consuming way. Nevertheless.

Once we have a list of films, we can go into the film data and extract film level characteristics. A sample film page can be seen https://letterboxd.com/film/la-la-land/here. We collected the first two actors cast (usually the leads), the director, genres (up to 5) year of release, number of ratings, and all the rating info from the histogram on the page, which gives us average rating and frequency of each star level.

Thus, we merge our user list and film list together to create our primary data frame we will be working with. Though this data frame does not contain the user and film level characteristics, these are stored in an easily referenced location for future matching and characteristic level regression analysis.

2.3 Summary Statistics

We can think of the data possessing two major attributes: film information and user information. Film information includes number of ratings a movie receives, average rating of a movie, the movie director and actors, the release data, the film genre. Although many of these variables can be scrapped for IMDb, due to time concerns we were unable incorporate many of these elements into our analysis. User information includes the number of ratings made by a user, the average rating of a user, films watched but not rated by a user (the data set denotes this as a 0 rating), a user's favorite films, in addition to others. Finally, we can think about the film-user information: the rating a user u gives movie i, and the time the rating was made. Although the timing information is important to account for and we elaborate later on potential ways to incorporate it into the model, due to the fact that we scrapped our own data, this information is missing from our data set.

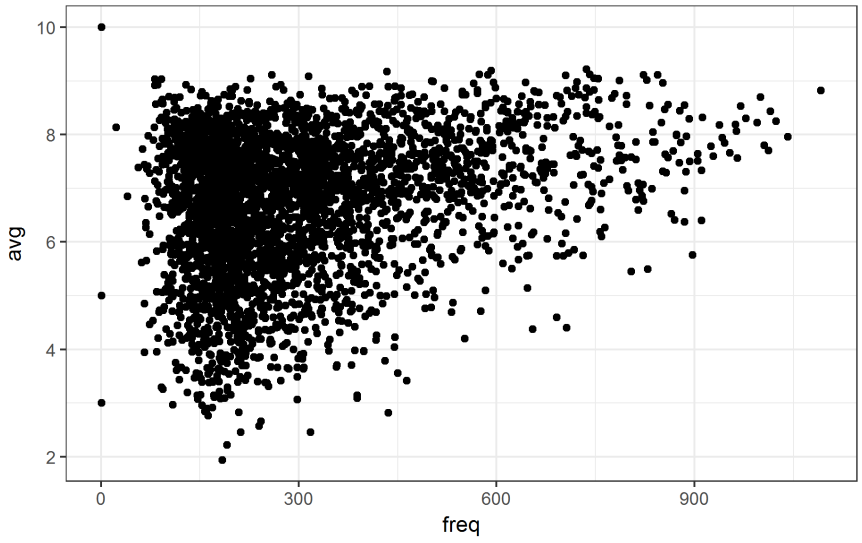

Figure 1: Average Movie Rating vs. Number of Ratings Per Movie

Before we begin to create formal models and predictions of ratings, we examined the data to gain an understanding of some of the characteristics mentioned above. The first graph displays the relationship between the average rating of a movie and the number of times it is rated. One can observe a positive relationship between the two: as the frequency of a movie being rated increases, the average rating for that movie increases as well. Although there are movies that are less frequently ranked but have average ratings just as high as those ranked often, there are also movies that are less frequently ranked and have low averages, which isn't true for the the movies that are ranked often. Furthermore, although it isn't possible to determine a casual relationship between the two, one can think of the relationships as a general indicator of which movies are being watched: users are less (more) likely to watch and rate bad (good) movies. We can test this directly with a kernel regression of the average movie rating on a dummy indicator of whether individual u watched movie i. Similarly, a kernel regression can be run on whether individual u rated movie i. Although the first kernel regression may be prone to error since the information is self-reported, the Letterboxd website consists of users who are more likely to provide consistent and accurate information on their movie watching preferences and behavior. We graph the kernel regression in figure 2.

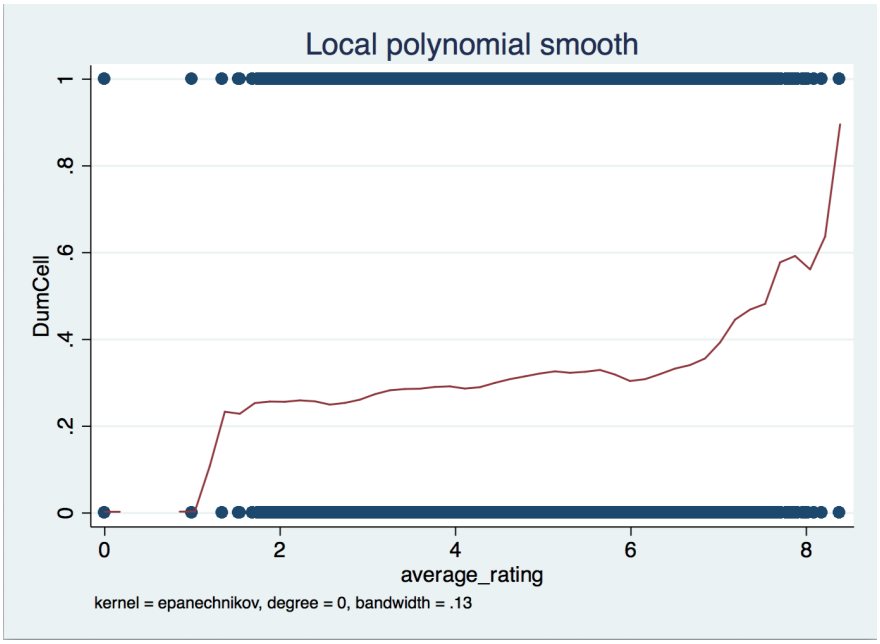

Figure 2: Kernel Regression of Average Movie Rating on Dummy indicating whether individual u watched movie i

Note that movies that are ranked relatively fewer times are still overall ranked often. This is one of the benefits of this data set: there are few movies that are ranked significantly more than others and users rank movies that they like and dislike with more consistency. This is evident form figures 3 and 4, which are the XXX and YYY distributions, respectively.

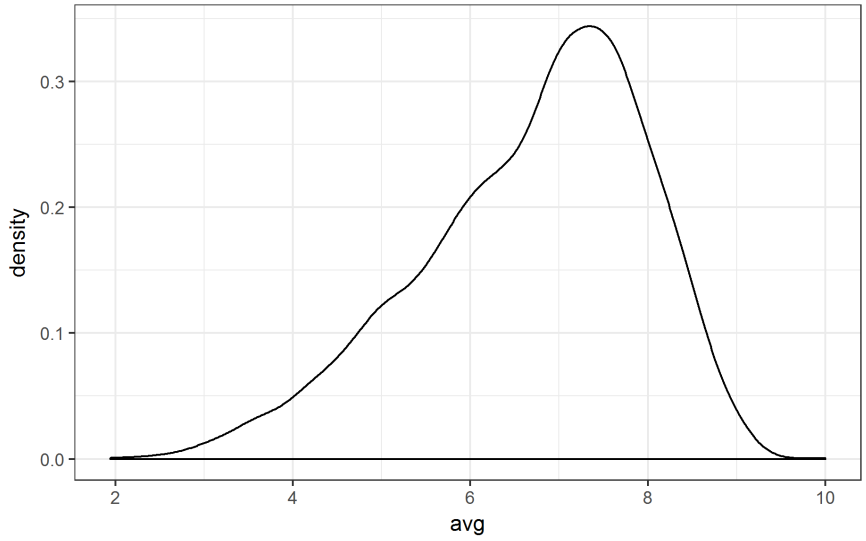

Figure 3: Density of Average Movie Rating by User

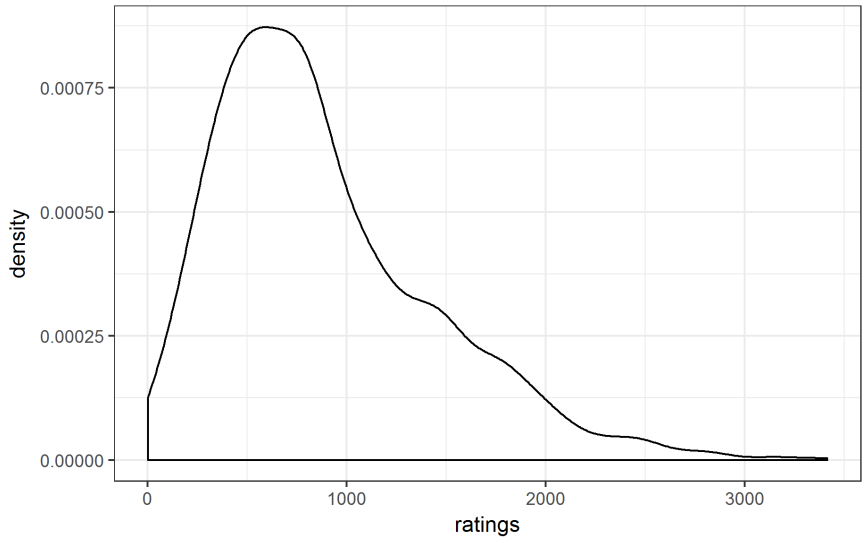

Figure 4: Density of Number of Ratings by User

3 Models

3.1 Naive Models

We start by implementing three naive models to get a better sense of the data. In the first naive model, we set the predicted value of every movie rating to the average value of all the ratings. This gives us a RMSE of 2.0439. The value of the RMSE may vary slightly, since every time we calculate the RMSE we do so on a new training data set randomly generated from all the ratings. The second naive model sets the predicted value of a movie to the average rating of that movie, while the third naive model calculates a similar average but with respect to the individual. The RMSE of the second and third naive model are, respectively, 1.6337 and 1.9356. Note that the RMSE of the second naive model is smalled than that of the third, implying that there is a smaller variance in ratings by movie than ratings by user.

3.2 Baseline Latent Factor Model

We implement a baseline model proposed by the winning team of the Netflix Challenge, BellKor (Koren 2009). Let

$$r_{ui} = \mu + b_i + b_u + \epsilon_{ui}$$

where r_{ui} is the rating of movie i by individual u; μ is the average movie rating for all i and u; b_i is the movie bias; and b_u is the individual bias. We include the movie and individual bias in the model because some movies may be much better or much worse than others, and individuals may be harsher or lighter reviewers than others. To minimize the RMSE, solve

$$\min_{b_i, b_u} \sum_{u,i} (r_{ui} - u - b_i - b_u) + \lambda_1 (\sum_u b_u^2 + \sum_i b_i^2)$$

We estimate b_i and b_u using the following formulas:

$$b_i = \frac{\sum_{u \epsilon R(i)} (r_{ui} - u)}{\lambda_2 + |R(i)|}$$

$$b_u = \frac{\sum_{i \epsilon R(u)} (r_{ui} - u - b_i)}{\lambda_3 + |R(u)|}$$

where $R(u)$ is the set of movies rated by user u; $R(i)$ is the set of users who rated movie i; and λ_2, λ_3 are regularization parameters. Although they are generally determined through validation, we set them equal to the regularization parameters used by the BellKor team: $\lambda_2 = 25, \lambda_3 = 10$. We solve for the predicted values:

$$\hat{r}_{ui} = \mu + b_i + b_u$$

and find the RMSE to be 1.5150. As expected, it is lower than the RMSE of all of the previous naive models. Although the RMSE is relatively high, several factors contribute to its magnitude. First is the size of the data set: our data set is less than 1 percent of that of the Netflix Challenge which results in a less accurate estimate. Most importantly, however, users on Letterboxd are likely to rate their movies regardless of the how much they liked the movie, which results in the ratings having a large variance.

As a check, we graph the residuals of the baseline model and they align with our expectations. The lowest values only have negative residuals, the highest has only positive residuals, and the residuals for each rating have approximately the same range.

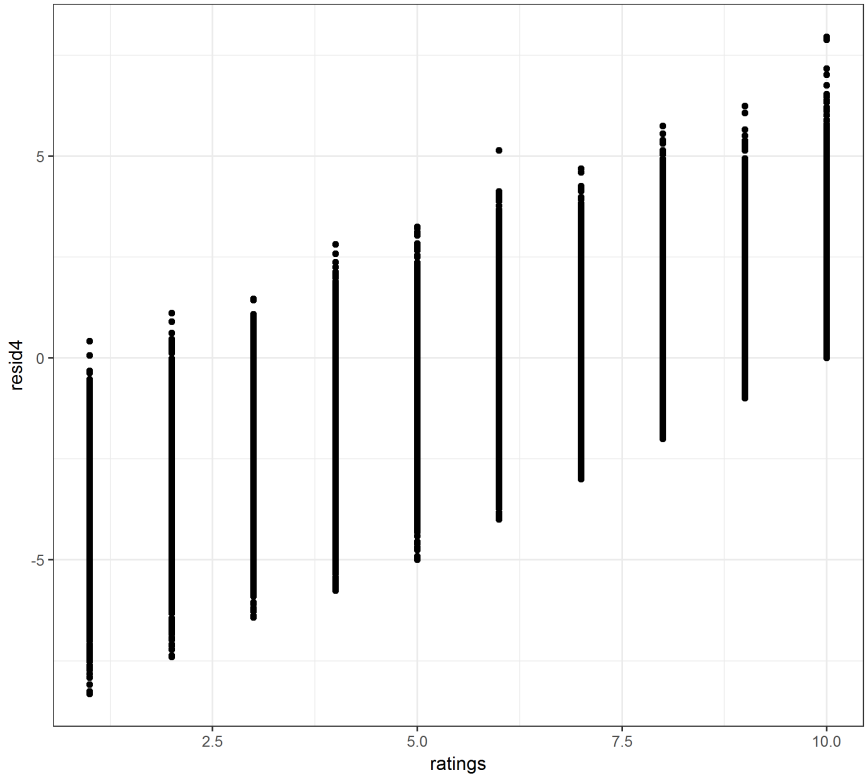

Figure 5: Residuals From Baseline Model

3.3 Neighborhood Model: K-Nearest Neighbor

The neighborhood model is one of the foundational models in Collaborative Filtering. Though it has many problems, such as the first-rater problem and sparsity, it is used in most Recommendation Systems because it is easy to implement, scales well, and costs relatively little, given that all the inputs and outputs are contained within the data frame and more inputs only increase the effectiveness of the model.

The Neighborhood Model involves finding users whose rating score are most correlated with the user of interest. This "correlation" is calculated using a Pearson's R test, and is created for each user compared to each other user. These are used as weights, to weight the ratings of the neighbors in predicting the ratings for the user of interest. This model is mathematically shown as following, taken from [5].

$$P_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \bar{r_a}) * (r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i=1}^{m}(r_{a,i} - \bar{r}_a)^2 * \sum_{i=1}^{m}(r_{u,i} - \bar{r}_u)^2}}$$

where $r_{a,i}$ is the rating given to item i by user a, is the mean rating given by user a and m is the total number of films. From this Pearson Correlation, we have our predicted ratings as:

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n}(r_{u,i} - \bar{r_u}) * P_{a,u}}{\sum_{u=1}^{n} P_{a,u}}$$

where $p_{a,i}$ is the prediction for user a for item i, and $P_{a,u}$ is the Pearson Correlation between users a and u and n is the number of users in the neighborhood. We follow the authors and [6] in choosing a neighborhood size of 30.

We implemented this Collaborative Filtering method on our Letterboxd data. We did not replace NAs with Zeros, as is usual practice. Therefore, some users have no predicted ratings for movies that no one in their neighborhood viewed. However, this problem was minimal as it was largely localized to films the user had also not seen, and the calculation of the RMSE ignores NAs. The RMSE value we obtain from this method is 1.291371, which is the lowest value out of all our models.

4 Conclusion

The baseline latent factor and nearest neighbor models we implement leave much to be desired and require numerous extensions to improve further. One such extension, for example, is the addition of time dependent bias in the latent factor model. User preferences develop over time: one may at first prefer action and adventure movies, but then grow to love drama and comedy. Similarly, movie popularity changes over time: winning the Oscars, for example, may generate more goodwill for a particular movie. In addition to including time dependent variables in the model, we can also directly test for the magnitude of the effect that winning the Oscars has on ratings. Although this information is present in the Netflix Data, since we scrapped our data, the time of each rating was not included in the data set. Furthermore, as the BellKor team notes, there is no perfect model and their best results

(i.e lowest RMSE) came from joining the latent factor and the nearest neighbor models together. Due to time constraints, we are unable to develop either model fully or combine them. Furthermore, our use of the richness of the Letterboxd data set is shallow in terms of developing a Pearson correlation and backing out user preferences. For example, we could use a clustering algorithm to cluster users by their stated favorite movies to group users prior to running our collaborative filter method. In addition, we could use user and item level characteristics and regressions. We leave these as projects for a latter date.

References

[1] Bell, Robert M., and Yehuda Koren. "Lessons from the Netflix prize challenge." Acm Sigkdd Explorations Newsletter 9, no. 2 (2007): 75-79.

[2] Feuerverger, Andrey, Yu He, and Shashi Khatri. "Statistical significance of the Netflix challenge." Statistical Science (2012): 202-231.

[3] Herlocker, Jonathan L., et al. "An algorithmic framework for performing collaborative filtering." Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1999.

[4] Koren, Yehuda. "The BellKor Solution to the Netflix Grand Prize." 2009.

[5] Melville, Prem, Raymond J. Mooney, and Ramadass Nagarajan. "Content-boosted collaborative filtering for improved recommendations." In Aaai/iaai, pp. 187-192. 2002.

[6] Salakhutdinov, Ruslan, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering." In Proceedings of the 24th international conference on Machine learning, pp. 791-798. ACM, 2007.

[7] Zhou, Yunhong, Dennis Wilkinson, Robert Schreiber, and Rong Pan. "Large-scale parallel collaborative filtering for the netflix prize." In International Conference on Algorithmic Applications in Management, pp. 337-348. Springer Berlin Heidelberg, 2008.