Assignment 3: Predicting Netflix Reviews

Caterina Golner
cgloner@princeton.edu

Tom Robbins
thomasrr@princeton.edu

Abstract

The collection of user data in online applications has become so ubiquitous that recommendation systems are now commonplace in areas such as online shopping and social media. Using a dataset of user reviews made available through the Netflix Prize competition, we aim to create a movie recommendation system using a variety of models that employ structural analysis or regression to predict a user's rating for any particular film. Our approaches focused on clustering users based on similarities in their viewing histories and on performing regressions across reviews. When measuring model performance using the R^2 values and MSE scores of the resulting predictions, we conclude that using Random Forests and AdaBoost Regression models yield the best results despite being slower than other methods. Random Forest and AdaBoost Regression achieved average R^2s of .166 and .164 respectively and average MSEs of .982 and .985 respectively.

1 Introduction

In order to generate accurate recommendations for their users, sites such as Netflix analyze user activity to predict what content their users will most enjoy. At such a large scale this task proves to be difficult because while trends may exist across movies (most people would agree that *The Godfather* is a better movie that *The Room*) and across users (some users may always rate movies more harshly on average than others) personal taste means that a high level of natural variability exists in the data and users will often depart from the these general trends. The average science fiction enthusiast, for example, will not want to see the same movies as a period piece fanatic. Moreover, users have rarely seen the exact same movies, and most users don't come even close to having reviewed most of the almost 18,000 movies covered in this dataset. As a result, it can be hard to make accurate predictions of what a user will think of a movie. In this assignment we use the Netflix Prize dataset of user reviews to attempt to predict a user's review for a given film they have already seen. Based on our knowledge of previous recommendation systems and collaborative filtering techniques, we will do this by reducing the dimensionality of the data, performing clustering on the users, and performing regressions in order to model the data and make predictions.

2 Related Work

The Netflix Prize contest in 2006, where our dataset comes from, drew a wide variety of approaches which we observed before employing our own methods of analysis of the data. The eventual winners, team "BellKor's Pragmatic Chaos," used a large ensemble of models to achieve their lowest error. Their efforts proved that no one model is perfect, especially when it comes to data like these [1].

Some of the most important models used in the winning solution were latent-factor models, such as Singular Value Decomposition (SVD), as these can help mitigate the sparseness of the data and prevent over-fitting [1]. Dimensionality reduction through methods like SVD and Principle Component Analysis has also been proven to improve the quality of the Collaborative Filtering systems of sites like eBay and Amazon [6].

Another important factor in a recommendation system is explainability: users, as well as the models that work on them, wish to understand the meaning behind the recommendations –*”You might like Inside Out if you watched Toy Story 3”*– and adds to the intuitive and real-world success of a prediction model. Unfortunately, explainability and accuracy have proven to be difficult to achieve together [3]. Because of this, we will not seek to build as complicated of a model as the winners of the Netflix Prize and will instead explore less sophisticated modeling options in order to better understand the patterns within the data.

3 Methods

3.1 Data Collection and Processing

The Netflix dataset is difficult to work with due to its enormous size in terms of both users and movies. In other applications of machine learning to this Netflix dataset, teams found that investigating *what* users were rating was just important in addition to the ratings themselves [1]. Due this– and to the sparsity of the user review matrix, we were motivated to gather additional information, such as genre and casting information, for each of the films in an attempt to fill out the data. To do this we used the python package `imdbpy` to query the IMDb [database] for these features. This way, we gathered this information for over 90% of the movies in the dataset. We one-hot-encoded these categorical labels and added them to our data. While this added many features, we found that the regressions did not improve when these features were added and selected for, leading us to believe that the underlying structures in the review data are not necessarily drawn on genre lines, and users' preferences of actors may not influence their rating of a film.

3.2 Spotlight Method: Singular Value Decomposition

For large datasets where the number of features is extremely high, reducing the number of dimensions is vital to achieving meaningful results in a reasonable amount of time. SVD can be used to pare down the dataset to the features which together explain the largest proportion of variance in the data in order to allow for faster calculations. In particular, for datasets as sparse as that of the Netflix Challenge, SVD is faster than PCA and can be used for non-square matrices while PCA cannot. [4]

Like PCA, SVD essentially works to determine how to rotate the ”axes” along this high-dimensional space so that they correspond to the directions in which the data contains the most variance, which also serves to uncover latent structures that describe the data generally known as ”components.” Under this system, the component corresponding to the largest singular value is the ”axis” across which the data shows the most variance, and the component with the second-largest singular value corresponds to the direction along which the most data varies in relation to the first axis, and so on. Once the data is represented using this new system, dimension-reduction can be performed by simply choosing the latent components that correspond to the largest singular values. [2]

A real $N \times D$ matrix X, can be decomposed as $U\Sigma V^T$, where U is an orthonormal $N \times N$ matrix, Σ is a $D \times D$ diagonal matrix containing the min(N, D) singular values, and V^T is an orthonormal $D \times D$ matrix. To compute V and Σ we observe the nature of $X^T X$:

$$X^T X = V\Sigma^T U^T U \Sigma V^T = V\Sigma^T \Sigma V^T = VSV^T$$

where S is a diagonal matrix which holds the squared singular values. If we multiply both sides by V we see that the eigenvectors of $X^T X$ must be equal to V and that the eigenvalues of X are equal to S, as

$$(X^T X)V = VS.$$

For the same reason, the eigenvectors of XX^T are equal to U and its eigenvalues are equal to the squared singular values, as

$$(XX^T)U = USV^T VS^T U^T U = U(SS^T)U^T T = US.$$

In the Netflix data, the resulting vectors or ”components” can stand to explain strong latent structures in the data. For example, it could be that movie reviews can be represented as a combination of how much your taste resembles that of a ”Strong Lover of 80s Rom-Coms” and that of a ”Cynical True Crime Thrillers.” If these are the latent components, and X is a $N \times D$ representing N users' reviews for D, then U relates each user to how much they align with each component, V^T relates movies to how much they align with each component, and Σ represents the importance of each component.

For our purposes, we performed SVD in order to be able to repeatedly perform clustering within a reasonable time. We analyzed what number of components optimized model performance and found that while reducing the number of components obviously reduced the proportion of variance that was explained by the combined components, it was only marginal, and the resulting MSE and R^2 values for our clustering-based predictors were not affected whatsoever. As a result, since clustering time is heavily affected by the number of components present, we decided to keep our SVD small and selected for only 5 components.

3.3 Prediction Methods

Using the SciKitLearn Python libraries, we used a variety of different approaches composed of multiple methods to predict a given user's review for a movie they had already seen. All parameters were left as the default unless otherwise stated. [5]

1. Linear Regression (LR)
2. Ridge Regression (RR) using *alpha=0.1*
3. AdaBoost (AB)
4. Random Forest (RF) using *num_estimators=50*
5. SVD using *arpack* + KMeans Clustering using *n_clusters=20* (Clus)
6. SVD + KMeans Clustering + Mean Difference (Clus+)

3.3.1 Regression Models

For each model, we predict a rating for a given movie by regressing over a feature vector of all of a users' other movie reviews. This allowed us to train each model on the the users that reviewed that movie. We chose our models to show both a baseline linear modeling approach and a more sophisticated, but more computationally intensive ensemble modeling approach (AB, RF). We train a model for each movie, so the speed and flexibility of the models is important.

Many movies have fewer than 500 reviews, which makes the regressions run quickly, but over 2,000 movies have over 10,000 reviews, making training a regression on all reviews technically unfeasible. To combat this, we sample 1,000 users to train on for movies with many reviews. This provides a trade-off of computational complexity and accuracy.

3.3.2 Clustering and Mean-Based Model

For our clustering-based approach (Clus), we wish to cluster users based on how similar their movie consumption habits are. Due to the high dimensionality of the data, we first performed SVD to reduce the data to 5 components. We then used KMeans clustering on the transformed data to produce group labels for all users. From then on, for each movie, we selected the users who had seen the movie and divided them into a training and testing set. The users in the testing set had their predicted reviews set to the average review from the users in the training set belonging to their specific cluster.

This simple model performed surprisingly well, but to improve the performance and fit the user specifically, we introduced another model, with a user-mean factor (Clus+). In this model, we also calculate the average review score for each user, and subtract from it the total average review score for all users in their specific cluster. This gives us a user-specific variance from the average of the cluster, allowing us to add it to the predictions generated in the above model to better fit each user.

3.4 Evaluation Methods

To evaluate the effectiveness of our prediction models, we split the users in the dataset who have reviewed a movie into a training and testing set, and evaluating the predictions generated by the model for the testing set. In the Results section, we show metrics of (R^2) and Mean Squared Error (MSE) for each model, per movie.

4 Results

There are many trade-offs inherent in the problem of recommendation. Below is a table showing metrics for the four prediction models we ran on the user review data.

		LR	RR	AB	RF	Clus	Clus+
R2	Min	-11.36	-11.36	-9.0	-5.701	-5.141	-14.25
	Max	0.7786	0.7478	1.0	1.0	0.8460	0.7928
	Median	-0.0655	-0.0663	0.1650	0.1669	.0056	.1673
	Mean	-0.1051	-0.1071	0.1647	0.1664	-0.0508	.0993
MSE	Min	0.0851	0.0851	0.0	0.0	0.0556	0.2038
	Max	14.84	14.84	12.5	10.0	9.53	15.75
	Median	1.222	1.222	0.9314	0.9284	1.175	0.9810
	Mean	1.294	1.296	0.9845	0.9823	1.259	1.066
Time (s)	Min	0.4769	0.4700	0.5073	0.4834	0.0003	0.0009
	Max	81.35	20.55	187.8	83.82	44.31	39.12
	Median	1.180	2.183	18.47	14.99	0.0037	0.0863
	Mean	1.324	2.640	21.36	16.72	0.2441	0.5347

Performance results and timing for prediction models. For each predictor, we include the distribution of R^2 values, MSE scores, and runtimes, in generating predictions across all movies.

Overall, our Random Forest and AdaBoost regressors were the best predictive methods as based on mean MSE scores and R^2 values. Random Forest slightly outperformed AdaBoost based on these metrics. Specifically, Random Forest achieved an mean R^2 value of 0.1664 and a MSE of 0.9823 across the 17,770 movies. While the regressions for some films performed poorly, the middle 50% of the Random Forest's R^2 values lie between 0.06 and 0.28 and the middle 50% of MSE scores lie between 0.77 and 1.14. In terms of R^2 value distributions, AdaBoost and Random Forest had inarguably better results by far than the other methods, as the minimum, median, mean, and maximum R^2 values across movies were all substantially higher with small exceptions. Interestingly, our Modified Clustering model had the best median R^2 value despite having a far lower mean. In the same vein, the MSE scores at each percentile are also far lower for AdaBoost and Random Forest than they are for the other models with the small exception that Clustering has the smallest maximum MSE. Linear Regression and Ridge Regression perform almost identically poorly and fare worse than both clustering methods in terms of R^2 and MSE distributions.

4.0.1 Timing Analysis

While the models that performed the best (Random Forest and AdaBoost) achieved the highest R^2 values, in terms of speed, they were not superior to the other models. The maximum runtime for a movie using AdaBoost is 187.8 seconds, which is 9 times more than that of Ridge Regression and more than double that of Linear Regression. In terms of median runtime, it takes almost 9 times longer than Ridge Regression and 16 times longer than Linear Regression. Compared to the Clustering model, AdaBoost lags heavily and clocks in at almost 5000 times slower. Random Forest has a slightly better median runtime than AdaBoost and a comparable maximum runtime to Linear Regression. Ridge Regression has the best worst-case runtime, and but the clustering models have the best mean and median runtimes by a factor greater than 10, particularly the basic Clustering model. Thereby, while Random Forest and AdaBoost may yield the best predictive results, they also have the slowest runtimes.

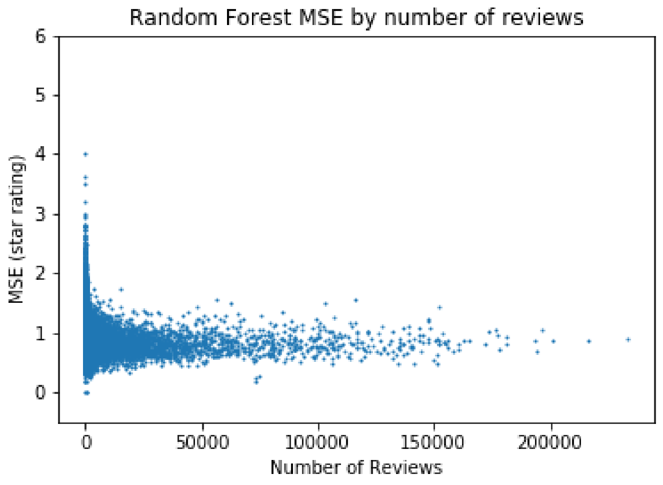
It is worth noting that the Modified Clustering(+) model always has a faster runtime than Random Forest and Ada Boost and its median R^2 value (0.1673) is higher than theirs, even if it lags in other mean metrics. Additionally, the Linear Regression model used nearly half as much memory as the other models,

5 Discussion and Conclusion

While both the movie-based regressions and the user-based mean models performed well, perhaps a combination of both approaches is necessary to achieve more accurate predictions, as research shows.

5.1 Feature Selection: Number of Reviews

We have two methods of sampling the users in order to train the regression models: ”top reviewers,” where we sample the users with the most reviews, as well as a random sampling method. In our findings, the random sampling performed much better, which suggests that the most active Netflix users are not predictive of the ratings of the majority of less-active users. However, an analysis of our models' MSE scores for a movie and the number of ratings for that movie shows that there is a relationship between the two when it comes to MSE variance across movies, especially towards the lower end of review counts. It may be worth considering total number of reviews when selecting a model for predicting reviews for a particular movie and potentially using multiple models that are better-suited for specific review quantity ranges.



5.2 Different Evaluations of the Models

We chose to show the metrics of our models as they performed *per movie*, so that each movie's predictions were weighted equally in our evaluation. However, we could also evaluate them on their performance *per review*, where we weight the models with more reviews (thus views) more by averaging the metrics by the predictions for each user. Evaluating some models this way makes them seem more favorable (such as the Cluster Mean-based model with user-mean factor and Random Forest Regressor). Below are the mean R^2 and MSE metrics for the models when evaluated per review.

	LR	RR	AB	RF	Clus	Clus+
R2	-0.4382	-0.1471	0.1679	0.1707	0.0544	0.1901
MSE	1.436	1.322	0.833	0.831	0.963	0.824

5.3 Future Work

For our clustering methods, we chose to perform SVD to reduce the number of movies considered so that we could perform user clustering. While this seemed most intuitive to us, it is also worth exploring a method which performs SVD to reduce the number of users in order to cluster movies together instead. This would change the nature of the problem from one in which we look at similarity of user types to generate predictions to one in which we observe movie types instead.

References

[1] Robert M. Bell and Yehuda Koren. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, December 2007.

[2] Jeff Ullman Jure Leskovec, Anand Rajaraman. *Mining of Massive Datasets*. Cambridge University Press, 2014.

[3] Yehuda Koren. Tutorial on recent progress in collaborative filtering. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 333–334, New York, NY, USA, 2008. ACM.

[4] Kevin P. Murphy. *Machine Learning : A Probabilistic Perspective*. MIT Press, 2014.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009.