Assignment 3: Netflix Prize

Peter Chen
Princeton University
xi@princeton.edu

Ben Burgess
Princeton University
bburgess@princeton.edu

Abstract

In this study, we analyze Netflix data in which 2,649,430 users have provided ratings on a 1 to 5 scale for 17,770 movies to predict scores for a given movie and user pairing. We implement a variety of different methods, including normalization of global variables, linear regression, and latent class clustering. We observe here that simpler normalization and regression models produce better predictions than the more complex clustering models. Finally, we demonstrate a method to extend the dataset using IMDb ratings and sentiment analysis from the movie's titles to improve the performance of our models.

1 Introduction

Recommendation algorithms provide advertisements and suggestions tailored to specific users interests have drastically improved consumer engagement and have become increasingly feasible provided the large quantities of data collected. Recommendation systems have a wide range of applications. In this case we seek to use patterns in movie viewing data to inform recommendation systems as part of the Netflix Prize Competition. Netflix, a popular streaming platform, collects large amounts of user reviews, rated on a 1 to 5 scale, for their movies. 2,649,430 users with ratings on a 1 to 5 scale are provided for 17,770 movies, with most entries being zeros indicating absence of review (this is provided as as a sparse matrix where the row corresponds to the movie ID and the column corresponds to the user ID). Our goal in this project is to **predict scores for a particular movie for a particular user** given the data we already have.

For our analysis, we implemented progressively more complex models: a simple normalization of global variables, a movie-centric ordinary least squares regression, latent class clustering, and lastly extending our features using various metadata from IMDb and using sentiment analysis. We find that the our initial simple normalization and regression models performed better than the more complex clustering model, but we were able to extend the dataset using IMDb ratings and sentiment analysis from the movie's titles to improve performance.

[6] [5]

2 Related Work

Common approaches used for recommendation systems include normalization of global effects, neighborhood models, matrix factorization, regression, and various ensemble methods. The intuition behind global effects normalization involves taking the mean of all reviews and adding user- and movie-specific deviations to predict for ratings. Neighborhood models involve finding a set of movies similar to movies that a given user also rated and take the user's mean ratings (or find a set of users similar to the user who rated a given movie and taking the movie's mean ratings). Matrix factorization methods, including singular-value decomposition, principle component analysis, and non-negative matrix factorization, decomposes the data set into a set of latent factors. Regression

methods uses all the movies that the user rated (or users that rated the movie) as the dataset to build a linear model for each user (or movie).

[6] [9]

3 Methods

3.1 Global Effects Normalization

In this simple initial model, we calculate the predicted rating as the sum of the overall average, customer offset, and movie offset, defined as follows:

- **Baseline rating**: the mean rating of all movies and users
- **User-specific effect**: the amount by which the mean rating for the given user deviates from the overall mean
- **Movie-specific effect**: the amount by which the mean rating for the given movie deviates from the overall mean

We tested this model by performing the following. We only gather the nonzero values of the matrix. Then for each rating, calculate the baseline rating, user-specific effect, and movie-specific effect not accounting for itself. Then we measure the predicted results against the true values by calculating r^2 as follows:

$$r^2 = 1 - \frac{\text{Model Sum of Squared Error}}{\text{Total Sum of Squared Error}} = 1 - \frac{\sum(y_i - \hat{y_i})}{\sum(y_i - \bar{y_i})}$$

Given the extremely large set of data we're working with, we randomly sampled 6000 movies, and for those with over 5000 user reviews, we randomly sampled 5000 users to speed up calculations.

[6]

3.2 Ordinary Least Squares Regression

We take a movie-centric approach to linear regression. For each movie, we learn a regression model using all the users that rated the movie as the dataset. We begin by removing all users that have never submitted a single review. Then, for each movie i, we consider only users that have submitted a review. The ratings will form the k x 1 vector y_i, where k = number of reviews movie i received. We proceed to regress on a matrix k x m X, containing k vectors for each user that submitted a review for movie i containing all the other m reviews (including 0's but excluding movie i) for the user.

To evaluate our model, we split our dataset into 80% training and 20% testing sets, fitting the model on the training, predicting on the testing, then calculating the prediction r^2. To speed up running time, we randomly sample 1000 training data points for movies that have more than 1000 reviews.

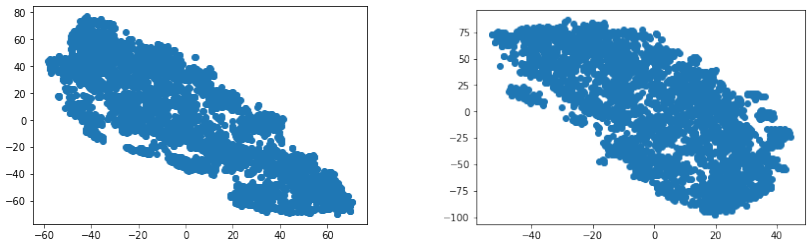
The ordinary least squares regression fits a linear model with coefficients $\hat{\beta}$ to fit a linear relation ship of x_j's on y's in the training data. Here, y would be the target movie rating and x_j's all other movie reviews for a given user. The equation will be in the following form:

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_m * x_m$$

where we calculate the estimate $\hat{\beta}$ by minimize the residual sum of squares between the observed and predicted responses, represented as:

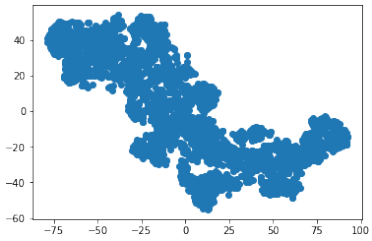
$$min||(\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_m * x_m) - y||^2$$

We can then use the predicted $\hat{\beta}$'s to predict for $\hat{y}$'s using x's from the testing data. We evaluate the model with predicted r-squared r^2, which measures correlation between predicted and true response values with higher values (max of 1 which indicates perfect prediction) indicating better predictions. The equation is as indicated in the previous section. [6] [3]

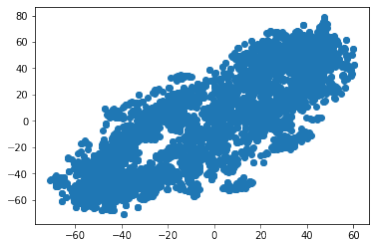


(a) (50—1000—40—10)

(b) (50—1500—40—10)



(c) (50—1500—40—5)



(d) (50—1500—40—50)

Figure 1: Clustering graphs (Perplexity—Learning Rate—Early Exaggeration—SVD Components)

3.3 Latent Class Clustering

We use a hard clustering approach to place each movie in one cluster out of nine. We then run a separate linear regression model on each of these cluster. By performing a separate linear regression on each cluster we may more appropriately evaluate the affect of certain features. For example, a review containing the term ”slow” may be a positive term for a drama while it would be considered a negative term for an action movie.

3.3.1 SVD

TruncatedSVD is used to perform feature selection on the dataset before we perform clustering. We extract 5, 20, and 50 features for the dataset by adjusting the number of components parameter.

3.3.2 t-SNE

t-distributed Stochastic Neighbor Embedding (t-SNE) is used to graph the clusters and select the best parameters for the number of components, perplexity, early exaggeration, and learning rate. We did not see very good feature separation as we varied the parameters as can be seen in Figure 1. We believe that this is partially due to the features in the dataset along with the relatively small subset we are working with. The best parameters we could find can be seen in Figure 1d. Given more time and computational resources, we would like to adjust the size of the subsamples and the amount of feature selection performed to see if we would be able to obtain better clusters.

3.3.3 KMeans

A varant of KMeans, MiniBatchKMeans with the k-means++ initialization method, is used to perform the clustering on the training dataset. We take the output of t-SNE with the optimal parameters and fit KMeans to it. We then estimate the number of clusters visually off of the t-SNE graph and use these for KMeans. We decide to use a value of 9 for the clustering.

The test set is then transformed which results in an array containing each of the cluster tags for every movie in the testing set.

Method	Average R^2	Notes
Normalization	0.109609	Average across random 6000 movies (5000 users max)
Linear Regression	0.089683	Average across all 17770 movies (1000 users max)
Clustering	0.082062	Average across all clusters
Extended Feature Set	0.131268	Average across all clusters

Figure 2: R^2 Results

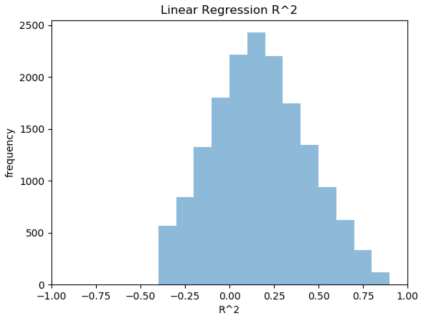


Figure 3: R^2 Distribution

3.3.4 Regression

The test and training set is split into separate data files based on the cluster label. We run our linear regression model on each of these separate clusters to determine an R^2 value for each of them.

[10] [2] [4] [9] [7]

3.4 Extending the Feature Set

To extend our features, we extracted a a number of other features related to the movie that may lead to better predictions. We use the Python package IMDbPY to retrieve data regarding each movie from IMDb's web server. IMDb, or Internet Movie Database, is an online database containing information about movies such as cast, production crew, genre, trivia, as well as reviews/ratings. We choose to use genre and user ratings. We can use the genre metadata to qualify our clustering results automatically generated from k-means. We also use Python package VADER, a rule-based model for general sentiment analysis, to extract the polarity and intensity of the movie titles. We use the calculated compound score which provides the sentiment intensity ranging from -1 to +1 to indicate its negativity/positivity. We would like to extend this in the future to use the movie's description for sentiment analysis since it contains more words. Additionally, we would like to implement stemming and other preprocessing techniques to improve the accuracy.

The sentiment and IMDb ratings were added as features after SVD was performed. We then reran t-TSNE and KMeans to generate new clusters. Our linear regression model was then run on all of these clusters and an R^2 value was calculated.

[1] [8] [11]

4 Results and Discussion

Normalization and Regression As we can see in figure 2, our very simple initial model using global averages performed better than our movie-centric linear regression, which in turn performed better than our attempts at latent class clustering. This is indicative of the recurring trend that simple models that are easier to interpret and implement often perform better than more complex models. Our R^2 across the board appear to follow a normal distribution centered around the mean (here, we show the histogram for R^2 values of our linear regression method in Figure 3).

Cluster 0	Cluster 1	Cluster 2
Nature: Antarctica	Dinosaur Planet	Clifford: Clifford Saves the Day!
Carandiru	The Killing	The Powerpuff Girls Movie
Ken Burns' America: Empire of the Air	Spartan	Lilo and Stitch
Cluster 3	**Cluster 4**	**Cluster 5**
Lady Chatterley	Cold Blooded	Sex and the Beauties
Love Reinvented	Arachnid	6ixtynin9
Fatal Beauty	A Killer Within	Grind
Cluster 6	**Cluster 7**	**Cluster 8**
Silent Service	Justice League	Zatoichi's Conspiracy
Outside the Law	WWE: Armageddon 2003	Rambo: First Blood Part II
Winter Kills	Sam the Iron Bridge	The Last Shot

Figure 4: Clustered Movie Titles

Clustering Figure 3 shows the contents of the KMeans clusters when it was run on a subset of the dataset. The contents of the clusters is somewhat intuitive with movies of similar genre being grouped together. However, when the full cluster is viewed major outliers can be seen. An example of this would be "The Edward R. Murrow Collection" being included in cluster #5 which is primarily more mature films or "Paula Abdul's Get Up & Dance" being included in cluster #9 which is primarily action movies. This most likely why we saw such bad performance when using the clusters with our linear regression model which can be seen in Figure 2. We believe this mediocre performance may be due to using the dataset that was fit by t-SNE for KMeans. t-SNE can generate fake patterns especially when the features are not well defined. Density information is also lost which can cause KMeans to fail even with simple examples.

Extended Feature Set The extended feature set consisting of our sentiment analysis and IMDb ratings combined with a subset of the data produced the best R^2 values. We believe that using IMDb scores provided better predictions as we are cross validating with ratings from a separate platform with more users. Furthermore, we speculate that the more positive or negative the sentiment of the movie title, the more indicative it is of the content of the film so users are more likely to enjoy the selected film as they are more aware of the content. The mean R^2 value can be seen in Figure 2. We believe we could achieve even better results if we ran the sentiment analysis on the IMDb descriptions since this would provide a larger corpus of text to extract the sentiments from. Additionally, adding IMDb reviews to movies that were lacking reviews would significantly help we believe.

5 Conclusion

In this paper, we present methods for prediction a movie's review based only on the contents of user reviews. None of our predictors worked exceptionally well overall. However, we extend the dataset using data from IMDb and a sentiment analysis tool which allows us to achieve better results. We present a detailed methodology for how to perform clustering on the dataset and some of the pitfalls of our method. Some areas we suggest for future work are as follows. Clustering based on the IMDb genre data may produce better results. A drama will most likely have different important features than an action film. Additionally by clustering on the genre data we will generate clearly distinct, disjoint clusters versus the t-SNE and KMeans method which had issues with false clustering.

In conclusion, this paper discusses multiple different methods for predicting movie reviews in the Netflix Prize dataset. We provide a detailed discussion of the tradeoffs and pitfalls for each method. Finally, we demonstrate a way to improve the performance by extended the dataset.

Acknowledgments

Prof. Barbara Engelhardt, Dr. Xiaoyan Li, who provided the lectures for COS 424.

References

[1] imdbpy documentation.

[2] sklearn.decomposition.truncatedsvd scikit-learn 0.19.1 documentation.

[3] sklearn.linear_model scikit-learn 0.19.1 documentation.

[4] sklearn.manifold.tsne scikit-learn 0.19.1 documentation.

[5] James Bennett and Stan Lanning. The netflix prize, 2007.

[6] Edwin Chen. Winning the netflix prize: A summary, 2011.

[7] Luuk Derksen. Visualising high-dimensional datasets using pca and t-sne in python, 2016.

[8] C.J. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. 2014.

[9] Yehuda Koren, Robert Bell, and Volinskym Chris. Matrix factorization techniques for recommender systems, 2009.

[10] Adam Kucharski. The netflix prize: Singular values for $1 million, 2016.

[11] Muthuraj Kumaresan. Sentiment analysis using vader. 2016.