Math 168 Final Group Project: Network Theory Methods for Improving a User Recommendation Algorithm

Daniel Hou[a,1], **Josh Limon**[a,1], **Zhou Yu**[a,1], **and Dawei Zhao**[a,1]

[a]Undergraduate Department of Mathematics, University of California, Los Angeles, CA 90095

This manuscript was compiled on December 27, 2020

In this paper, we see how accurately a variety of similarity metrics can predict what movies a given user would want to watch based on the viewership and ratings of similar users. We also introduce and test the validity of two variations of the Probabilistic Spreading method. The first variation involved introducing negative reviews as a consideration. The second variation incorporates trying two new thresholds for determining what reviews get considered in the calculation.

recommend | similarity | networks | netflix

In 2006, Netflix, Inc. publicly announced a contest with the goal of increasing the accuracy of Netflix's movie rating prediction algorithm that predicted how a given user yet to watch a movie would rate that movie (1). The final grand prize winner was awarded to an algorithm developed by Yehuda Koren, Robert Bell, Chris Volinsky, and some other contributors from AT&T called the "BellKor Pragmatic Chaos" solution which resulted in a 10.06% increase in accuracy increase from the original algorithm at the start of the contest in 2006 (1, 2). The team heuristically employed a blend of 18 individual predictor variables (which account for a variety of variables such as time spent watching movies, cynicism of a user, etc.) trained on data Netflix published publicly (and would try to do so again after 2009 but would retract said data in favor of consumer privacy) to achieve said 10% increase. This seems to imply that employing a variety of prediction techniques is most effective in producing high accuracy predictions (3). This successful methodology motivates us to investigate more methods for producing predictions.

In this paper, our team investigates if there exist techniques in the study of network theory that would help improve user opinion predictive algorithms. The techniques we investigate include adjusted cosine similarity, baseline predictors, probabilistic spreading and a novel variation on probabilistic spreading. While probabilistic spreading usually only counts neighboring node with a binary value of 1 (and non-neighbors as 0), we test if incorporating a negative value for dissimilar users would help create more accurate predictions. We also introduce the novel concept of using a baseline estimator as a more specific mean for various movie-user relations.

Preliminaries and Data

Netflix currently has 195 million paid subscribers for their services. Our study was done based on a dataset taken from 2005 of 2,649,429 users and 17,770 movies. At the time of data collection, Netflix was using a 5-star rating system where a user could review any movie. For our experiments, we randomly selected 10,000 users and 5000 movies to test which recommendation techniques most predicted the ratings users gave to the films they viewed. In the following sections, we will go into greater detail of the specifics of how this selection of users and movies was divided to predict user ratings.

Similarity Metrics and Rating Prediction Methods

A usual concept in the study and analysis of networks is the idea of similarity. This study of similarity between nodes is used to gather information crucial to understanding the overall trends and relationships in a network. Most often, similarity can be used to develop predictor tools of peoples' behaviors in the study of social networks. Our team decided to use similarity in hopes of predicting the Netflix titles that consumers would be more likely to watch and rate highly based on their previous watch history combined with their ratings of these. We tested our data against three methods of measuring similarity. An adjusted cosine similarity, Probabilistic Spreading method, and using baseline predictors as a predictor for ratings were all studied in hopes of finding the most efficient tool in creating an item-based similarity recommendation system. All three methods (and the idea of measuring similarity in general) will be used on a bipartite network between a group of nodes representing Netflix subscribers and a group representing the movies they watch and rate.

Cosine Similarity. There are generally two approaches to measuring similarity in networks, structural and regular equivalence. For our purposes, since we are studying the ratings of the same movies that audience members are watching, we will be using the structural equivalence approach. Cosine similarity takes the most obvious approach to measuring structural

Significance Statement

Rating prediction algorithms are often used to help users sort through a large database of objects, movies, books, and other items. The accuracy with which an algorithm predicts a user's rating of a given object before they rate the object is the usual metric used for comparing rating prediction algorithms. We test the accuracy, using a variety of metrics, of old methods and a new method for sorting movies against a data set of Netflix's user ratings.

Author Contributions: D.Z. was lead programmer and coder, Z.Y. created graphics and led cloud computing, J.L. introduced idea and cowrote paper with D.H.

No competing interests

[1] All authors contributed equally.

[2]To whom correspondence should be addressed. E-mail: jjlemonheads@gmail.com

equivalence, which is to count the number of shared neighbors between two nodes, and takes it one step further by taking into account the varying degrees of nodes in the network (4). This ideally rids the experiment of any misinformation due to the number of common neighbors not being scaled according to the overall size of the network. Although cosine similarity often uses unweighted networks, our study will be using a weighted network to address the different ratings between 1-5 that audience members can give the films they watch. Using the entries of our adjacency matrix, we divide the number of common neighbors by the geometric mean of the degree of nodes, using the following equation:

Baseline Predictors. Baseline Predictors, originally from an earlier iteration of the BellKor solution to the Netflix Grand Prize, introduce a method that incorporates the biases of items and movies on an individual basis. In our paper, we explore and use this fundamental idea of a Baseline Predictor. The Baseline Predictor gives us an expected rating that a given user gives a certain item. The formula for a Baseline Predictor is as follows (2):

$$b_{ui} = \mu + b_u + b_i$$

In this example, μ is the overall average rating a user gives a movie, b_u is the user rating bias and b_i is an item rating bias. For example, maybe Tom is a critical user that prides himself on only the most niche and best movies, so he get a negative b_u score (maybe -0.7). The Godfather is one of the highest rated movies of all time and so it likely gets a positive b_i score (maybe +0.5). The method for computing the b_i is as follows:

$$b_i = \sum_{u \in R(i)} \frac{(r_{ui} - \mu)}{\lambda_1 + |R(i)|}$$

Where $R(i)$ denotes the set of users who rated an item, i, (r_{ui} is the actual rating a user, u gives an item, i, and λ_1 is a regularization parameter and we abitrarily use Koren et. all's value of $\lambda_1 = 25$ (2). Their value was found by shrinking the average towards zero on the probe set (2). The method for computing b_u is similar and follows:

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

In this case, $R(u)$ denotes the set of items rated by a user, u, and λ_2 is again a regularization parameter and we again abitrarily use Koren et. all's value of $\lambda_2 = 10$. Yehuda Koren and the team incorporated more considerations like how many ratings a user gives on a day, how the popularity of a movie changes, etc (2). We opted not to use these more advanced methods in favor of being able to use the baseline estimator as a threshold for determining whether or not a rating gets accounted for, regardless of time.

Probabilistic Spreading (ProbS). The paper, "Network-based recommendation algorithms: A review", by Yu et. all introduced a recommendation method called Probabilistic spreading, or ProbS. The method involves taking the matrix of the users, items, and links (say U, I, and E respectively) in a bipartite network and projecting them onto a matrix of link weights W ($G(U, I, E) \rightarrow G(W)$) (5).

We compute the elements W_{ui} (u corresponds to a user and i corresponds to an item) as follows:

(a) Give each of the items a user is connected to weight $1/k_i$ where k_i is the degree of the item. Give an item weight zero if it is not connected to the user.

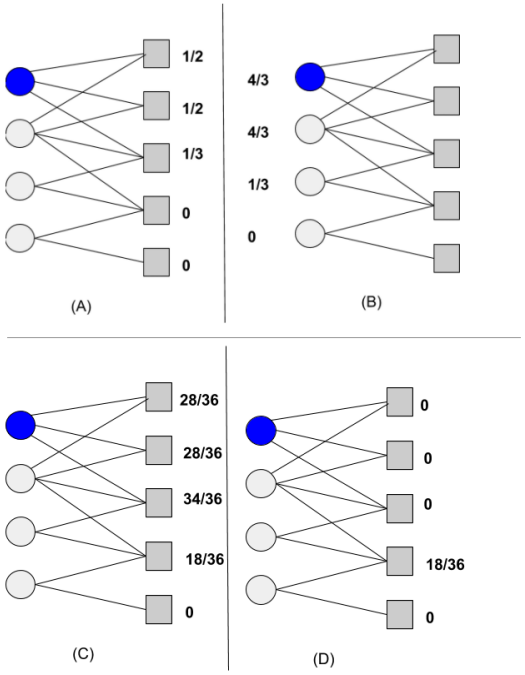


Fig. 1. In this example, circles are users and movies are squares. The highlighted blue circle is the user that we wish to calculate the ProbS value for as outlined by steps a - d. Notice the top circle and the circle beneath it have three movies in common and the only movie not watched by the top circle is recommended somewhat strongly by this algorithm.

(b) Add all the item weights that are connected to a user and assign that summed value to be the weight of the user. Repeat this for all users.

(c) First divide all the user weights after from the last step by the degree of that node. Then, sum all of these corresponding quantities for users connected to a given item. Associate this sum with that item and repeat this step for all items. This is the initial recommendation weight (all the weights should sum to the degree of the original user).

(d) Change the weight of any item the user is originally connected to to zero (we are not interested in predicting if they will purchase, watch or rate a movie again)

The weights throughout can be thought of as probabilities of a random walk starting from the user and going down a given path. In fact, random walks is how this method was originally developed and explained. (5). The usual approach to recommendation algorithms is to drop the ratings in the negative group since we only want to recommend movies the user is likely to enjoy. Our team explores three different thresholds to divide the reviews into positive ones and negative ones:

1. $\gamma = 3$. The control threshold used in the Bellkor solution (2).
2. $\gamma = \mu$ The overall average rating a user gives any movie.
3. $\gamma = b_{ui}$ The Baseline Estimator for that rating.

Probabilistic Spreading (ProbS) with Negative Ratings. To expand on the probabilistic spreading method, we introduce a new method that differentiates and accounts for movies that are rated below a given value. Since we're interested in movies that are likely to be rated highly by a user and would prefer to ignore the movies that are likely to displease a user, our team proposes a system of weighing the ProbS values accordingly while taking this into account. In our proposed method, we take a threshold, for now let's call it γ. Take the matrix of ratings (links) $E = r$ and for any user rating $r_{ui} > \gamma$ above the threshold, we put it into a "positive" group, we'll call a. Similarly, for any user rating $r_{ui} < \gamma$ below the threshold, we put it into a "negative" group, we'll call b.

We are only interested in the accuracy of the top movies in each user's recommendation. Let's call this number L and we'll usually use $L = 20$ For each user, among all top L movies of the user's positive ranking, if any of them happens to be in the top αL ($\alpha > 0$) of the user's negative ranking, we replace it with the highest ranked movie that is not in top L of the positive ranking. The general advantage and motivation for trying such an approach is that For a user u, the higher movie i ranked on u's negativity ranking, the more likely that user u would give this movie a bad review. We should not recommend movie i to user u. In the following section, we look at a variety of metrics used to evaluate this new technique.

Recommendation Evaluation

Our process of evaluating these recommendation methods is based off those frequently used for similar tasks, and most notably those developed in "Evaluating collaborative filtering recommender systems" by Jonathan Herlocker et. al and "Evaluating recommendation systems" by Guy Shani and Asela Gunawardana. The evaluation processes outlined in these studies use a double and triple division evaluation system, where "double" and "triple" refer to the number of subsets data is divided into. In a double division evaluation, the data is divided into a training set and probe set, while triple division evaluation takes into account method parameters that optimize the data to result in bias in its performances when run under tests (5). Our study did not include any variable method parameters (we simply borrow already-optimized values from the papers we learned from, such as the use of 90/10 ratio or the regularization parameters of the Probabilistic Spreading model (2)) so we will be using the double division evaluation system.

The purpose behind creating a training and probe set is to be able to test estimated recommendations against our own data. Training sets, which we will denote as S^T, are used as the input data for our evaluation methods described above, and these predictions are then compared to the probe sets, S^P to test their accuracy. The level of accuracy is quantified in four forms explained below - ranking, precision, differentiability, and novelty.

Ranking. The ranking, or ranking score, of a prediction is the most general way of quantifying the accuracy of an evaluation method. Based on the recommendation methods and algorithms computed on the data, we create a list of recommended films for a user in descending order. We then compare this list with the actual films left in the probe set, also placed in descending order based on their given ratings. Every user and movie pair is evaluated against the training set's prediction, summed together, and then normalized based on the size of the probe set. This gives us the following formula, for our ranking score r (5):

$$r = \frac{1}{|\mathrm{E}^P|} \sum_{(i,\alpha)\in \mathrm{E}^P} r_{i\alpha}$$

Since r can be see as the difference between our computed recommendation lists and legitimate recommendation lists, a small value corresponds to accurate recommendations.

Precision and Differentiability. However, at times this general ranking system raises two issues. In reality, only a limited list of films would need to provided as recommendations, and recommendations only provide useful information if they are unique to a user. To address these issues, we used the measures of precision and differentiability to provide alternative metrics for the accuracy of our evaluations.

For precision, we can simply look at the top L results (our study chose L = 20) of a recommendation list, and see how many entries match the probe set of a given user. Quantitatively, we do this for each user i, and then average over all the nodes in the probe set to generate our entire network's precision (5).

$$P_i(L) = \frac{d_i(L)}{L}$$

Differentiability is the only measure of accuracy that compares the evaluations of two users with each other. For this metric, we look at the number of shared entries in the top L results of the recommendation lists for any two users, convert it into a fraction of L, and then subtract from one. Once we find the average over all pairs of nodes in the network, this resulting differentiability tells us how much recommendation lists differ from one another.

$$D_{ij}(L) = 1 - \frac{C_{ij}(L)}{L}$$

For both precision and differentiability, high values infer a recommendation method that performs well.

Novelty. Finally, novelty looks at the average degrees of all the nodes in the top L results of all the recommendation lists produced, and takes the average of these (5).

$$N(L) = \frac{1}{LU} \sum_{i=1}^{U} \sum_{\alpha\in \mathrm{O}_L^i} k_\alpha$$

In doing so, we are able to see how popular the movies recommended to a user are. Rather than seeing how unique a viewership is compared to other users, we instead see how unique a user's viewership is in the broad scope of films on Netflix. Note that our measure of novelty is not a percentage like our other processes of evaluation, and can have much higher values. Generally, smaller values of novelty correlate to a successful recommendation method.

Running the Trials

Since there are too many users, directly working on the original dataset is beyond the scope of the computing power our team currently have. We randomly picked 5,000 users and 10,000 movies each time (three times in total) to have three bipartite graphs. Within each of these graphs, we randomly picked

90% of the data as training set and the remaining 10% as probe/test set. We used Networkx Python package to store and manipulate the graphs since it is much more efficient than the raw adjacency matrix. On average, in the training set we have 1,846 users and 3,865 movies. All the results represented are the average of all the trials we did on these three graphs.

Results

First of all, we look into the Ranking (r), Precision (P), Hamming Distance (H), and Novelty (N) of the Baseline method. The result is shown in the below table:

Method	r	P	H	N
Baseline	0.2542	0.00015	0.1782	0.2992

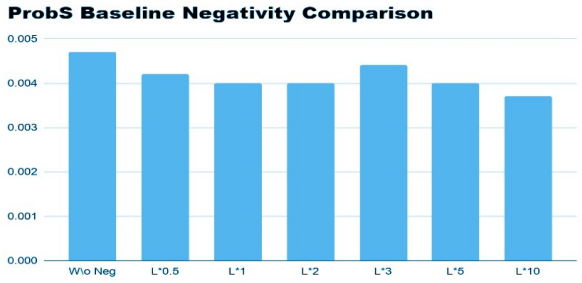
Then, we examine the second method, Probability Spread. Without having negativity, the results with different thresholds are as follow:

Method-Threshold	r	P	H	N
ProbS-3	0.2991	0.0044	0.2289	12.1760
ProbS-Avg	0.3121	0.0046	0.2395	12.8243
ProbS-Baseline	0.3456	0.0047	0.2746	13.3570

where ProbS-3 takes the number 3 as the rating threshold, ProbS-Avg takes the average rating of all movies, and ProbS-Baseline takes the baseline. Clearly, among the three thresholds, the baseline has the best performance. Thus, we select baseline as the threshold and include negativity with different values of L to check if negativity will further improve the result.

L	r	P	H	N
no neg	0.3456	0.0047	0.2746	13.3570
L*0.5	0.3456	0.0042	0.4032	10.7598
L*1	0.3456	0.0040	0.4110	10.2196
L*2	0.3456	0.0040	0.4088	10.2051
L*3	0.3456	0.0044	0.3185	12.1460
L*5	0.3456	0.0040	0.4079	10.1844
L*10	0.3456	0.0037	0.4005	9.8915

The result in terms of precision is also shown in the below bar diagram.



Clearly, for Probability Spread, inclusion of negativity only make prediction worse.

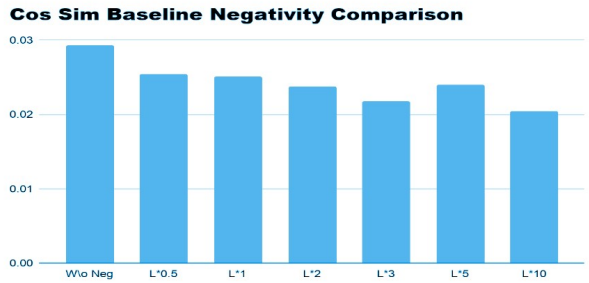
Then, we take the similar analytic approach to our last method, Cosine Similarity. Without negativity, the result is shown as follows:

Method-Threshold	r	P	H	N
CosSim-3	0.1034	0.0268	0.5300	86.9456
CosSim-Avg	0.0897	0.0279	0.5813	85.4216
CosSim-Baseline	0.0583	0.0293	0.5970	82.9105

Clearly, as in the method of Probability Spread, the best threshold is still baseline. Then, we choose baseline as the threshold and take negativity into account. We get

L	r	P	H	N
no neg	0.0583	0.0293	0.5970	82.9105
L*0.5	0.0583	0.0254	0.6333	82.5001
L*1	0.0583	0.0251	0.6555	82.6072
L*2	0.0583	0.0238	0.6594	83.4826
L*3	0.0583	0.0218	0.6750	80.7860
L*5	0.0583	0.0240	0.6350	83.4856
L*10	0.0583	0.0204	0.6833	80.9457

The result in terms of precision is also shown in the below bar diagram.



Surprisingly, for the Cosine Similarity method, inclusion of negativity also worsens the results just as in Probability Spread.

Finally, we compare the best performance of each method and make the following table

Method	r	P	H	N
Baseline	0.2542	0.00015	0.1782	0.2992
ProbS	0.3456	0.0047	0.2746	13.3570
CosSim	0.0583	0.0293	0.5970	82.9105

Conclusion

Based on the results (especially the final table) in the previous section, we conclude:

1. Taking the threshold of baseline, i.e. adjusted average, generally gives the best prediction
2. Including negativity into the prediction model generally worsens the prediction

This experiment has showed us that changing the threshold from the usual standard of $\gamma = 3$ to $\gamma = b_{ui}$ generally results in a higher accuracy for rating predictions using the Probabilistic Spreading model. Some ideas that may warrant research in the future include investigating whether or not using this new threshold with the winning Bellkor Chaos solution would yield more accurate results. Another aspect to investigate is whether or not using time-dependent baseline predictors for different movies would result in a higher accuracy overall.

Code. Can be found at: https://github.com/daweizhao23/math_168_proj

Bibliography

1. I Netflix, Netflix prize (2009).
2. Y Koren, The bellkor solution to the netflix grand prize. (2009).
3. A Töscher, M Jahrer, The bigchaos solution to the netflix grand prize. (2005).
4. ME Newman, Networks. (2018).
5. F Yu, A Zeng, S Gillard, M Medo, Network-based recommendation algorithms: A review. *Phys. A: Stat. Mech. its Appl.* **452**, 192–208 (2016).