CS224W Project Report

Margaret Fu

December 11, 2011

1 Introduction

Recommendation system has become very popular in many aspects in real social networks, such as e-commerce services Amazon.com, movie rating website IMDB, and DVD rental service company Netflix. This project focuses on signed link prediction of recommending movies. Movie rating data has become largely accessible via the Internet. Most rating systems are designed to have 5 score choices from 1 to 5. However, in finding a potential interesting movie for a certain user, there is really not much difference between score 4 and 5. What I care about is wether or not I will like the movie. In fact, this is a classification problem on predicting the category of a movie that has not been watched by the user. So instead of a specific score, we are more concerned with whether the movie is below 3 or above 3, which will be interpreted as "like" and "dislike". A score 3 will be considered undetermined or "indifference".

Another important motivation for this project is to build a social recommendation system. The idea comes from recommendations we get from our friends in everyday life. It is quite common that we tend to value more about reviews given by our friends than those from other strangers. In this project, a `user_graph` is built, based on movie watching history, to help determine whose opinions to consider when making up a new recommendation. But if real social network data of all users are available, we can use that to assign different weights when calculating a score of the recommendation.

2 Prior Work

In [2], Benchetta et al. examined a bipartite graph model. Graph structures are carefully analyzed using different metrics, such as Common Neighbors, Jaccard's coefficient and Preferential Attachment. They also learned the topological structure of the projected graphs. Link prediction in the projected graph is mentioned using statistics of the graph.

In [3], Huang et al. introduced why collaborative filtering algorithm would work in recommendation system. Topological attributes of the bipartite graph are studied by introducing concepts of 4-node clustering coefficients and 6-node clustering coefficients. Results from a basic CF algorithm using statistical correlation are shown for predicting potential interest.

However, no detailed learning algorithms and models are explained in the above two papers. In the rest of this report, a detailed supervised learning algorithms Naive Bayes will be presented. Prediction results are shown at the end. We use the same measurements used in previous papers to evaluate the model.

3 Model

3.1 Graph Structure Analysis

This user-movie network can be modeled as a bipartite graph $G =< U, M, E >$, where U and M are two mutually exclusive sets of users and movies. E is a set of edges of G and also a subset of $U \times M$. Each $e = (u, m) \in E$ indicates an existing rating from a user $u \in U$ to a movie $m \in M$. Also a set of weights W is associated with edges in E such that $+1$ and -1 are used to represent "like" or "dislike". This is the signed weighted bipartite graph we use for the movie recommendation system. Some basic analysis of the structure of this bipartite graph are studied, such as clustering coefficients and degree distribution. Furthermore, we calculate Jaccard's coefficients and cycle clustering coefficients.

1. $$\text{Jaccard's coefficients (of } u_1 \text{ and } u_2 \text{ in } U) = \frac{|\tau(u_1) \cap \tau(u_2)|}{|\tau(u_1) \cup \tau(u_2)|}.$$

   $\tau(u_i)$ denotes for the neighbors of u_i for $i = 1, 2$. Here it refers to the ratio of the number of movies they have both rated over the number of movies that have been rated by either one.

2. In [3], the author introduced 4-node and 6-node bipartite clustering coefficient defined as
   $$C_4 = \frac{4 \times (\text{number of 4-node cycles in the bipartite graph})}{\text{number of 4-node paths}}.$$

   Here for the convenience of computation, we consider a cycle clustering coefficient of two user nodes, which can be viewed as an approximation to the above definition.

   $$C(u_1,\ u_2) = \frac{\#\text{ of actual shared neighbors}}{\#\text{ of possible shared numbers}} = \frac{|\tau(u_1) \cap \tau(u_2)|}{|M|}.$$

   Note that every pair of shared neighbors can create a 4-node cycle in the bipartite graph.

Jaccard's coefficient and cycle clustering coefficient both describe how thoroughly that users are associated with movies. The higher the coefficients, the more movies that have been rated by the same subgroup of users. And thus, when we predict the sign of an non-existing link between a user and a movie, we can extract more useful and correlated information between that user and his neighbors in the `user_graph`.

Moreover, to examine how users are correlated to each other when making recommendations, we project the bipartite graph onto the user nodes U. So the `user_graph` are built over all the users. There is an edge between two users if and only if they share a common neighbor in the bipartite graph, i.e. they have both rated the same movie. In addition, each edge $e = (u_1, u_2)$ is associated with a weight w defined as

$$w(u_1, u_2) = \frac{\text{\# of agreements}}{\text{\# of disagreements}}.$$

In particular, among the subset of movies $\tau(u_1) \cap \tau(u_2)$ that both users have rated before, we count how many times their ratings agree with each other, i.e. both "like" or "dislike" the same movie, and how many times they disagree, i.e. one "like" and one "dislike". The higher the weight, the more common taste that the two users share. So if we want to calculate a recommendation for u_1, we would value more of the opinions from his high weight neighbors and less of the opinions from his low weight neighbors. And of course, weights are mutual to the two end-point users.

3.2 Naive Bayes Learning Algorithm

Naive Bayes algorithm computes the conditional probability

$$p(C_j|F_1, \cdots, F_n) = \frac{p(C_j) \cdot p(F_1, \cdots, F_n|C_j)}{p(F_1, \cdots, F_n)} = \frac{p(C_j) \prod_{i=1}^n p(F_i|C_j)}{p(F_1, \cdots, F_n)},$$

where C_j denotes the categories in the classification problem and $(F_1, \cdots, F_n)$ is the feature vector we define to describe an item. Usually, we want features to be independent with each other so that the likelihood probability $p(F_1, \cdots, F_n|C_j)$ can be calculated by the product of conditional probability of each feature $p(F_i|C_j)$.

In this project, our C_j's are categories of "like" and "dislike" for $j = 1, 2$. Given a certain user u and movie m, we want to predict the sign of the link $e = (u, m)$. In order to do so, we need help from the ratings toward m from other users. We consider neighbors of u in the `user_graph`, i.e. the users $N(u;\ m)$ who have rated common movies with u before and have also rated movie m. Therefore, we define the features F_i of m to be the ratings of $u_i \in N(u;\ m)$. For example,

$$F(m) = [+1,\ +1,\ -1,\ +1,\ -1, \cdots]$$

among neighbors of u who have watched movie m, user 1, 2 and 4 like it, and user 3 and 5 dislike it, and etc.

The prior probability $p(C_j)$ is the fraction of movies that have already been rated by u in each category, i.e.

$$\begin{aligned} p(C_1) &= p(\text{``like''}) = \frac{\text{\# of ``like''s by } u}{\text{\# of rated movies by } u} = \frac{\text{\# of ``like''s by } u}{|\tau(u)|} \\ p(C_2) &= p(\text{``dislike''}) = \frac{\text{\# of ``dislike''s by } u}{\text{\# of rated movies by } u} = \frac{\text{\# of ``dislike''s by } u}{|\tau(u)|}. \end{aligned}$$

The likelihood of each feature $p(F_i|C_j)$ is the fraction, among movies in category C_j, of those having the same rating as m by $u_i \in N(u; m)$, i.e.

$$\begin{aligned} p(F_i = +1 \mid C_1) &= \frac{\text{\# of movies } u \text{ and } u_i \text{ both ``like''}}{\text{\# of ``like''s by } u}, \\ p(F_i = -1 \mid C_1) &= \frac{\text{\# of movies } u \text{ ``like'' but } u_i \text{ ``dislike''}}{\text{\# of ``like''s by } u}, \\ p(F_i = +1 \mid C_2) &= \frac{\text{\# of movies } u \text{ ``dislike'' but } u_i \text{ ``like''}}{\text{\# of ``dislike''s by } u}, \\ p(F_i = -1 \mid C_2) &= \frac{\text{\# of movies } u \text{ and } u_i \text{ both ``dislike''}}{\text{\# of ``dislike''s by } u}. \end{aligned}$$

The evidence $p(F_1, \cdots, F_n)$ is the fraction of movies among all movies that have the same rating as m by these $u_1, u_2, \cdots, u_n$ people.

After defining each of these features and conditional probabilities, Naive Bayes naturally computes the posterior probability of movie m belonging to "like" or "dislike". The category with the higher posterior will be our prediction of the sign of this non-existing link $e = (u, m)$ in the bipartite graph.

3.3 Implementation

The algorithm takes in a pair of specified user u and movie m (usually not neighbors in the bipartite graph) and predicts the signed weight of $e = (u,\ m)$. There are several places that need attention during implementation.

1. Since we compute posteriors for each category to compare which category has the highest probability, we do not normalize all the posteriors by dividing $p(F_1, \cdots, F_n)$. For one thing, given a specified movie m, this value is the same for all posteriors of different categories. No effect would be made on the comparison. For another, this is usually an extremely small number, relative computational error of which can be huge.

2. Now we are only concerned with $p(C) \prod_{i=1}^{n} p(F_i|C)$. Since we are multiplying a series of probabilities, the product gets very small. Actually, at the end of calculation, they are always beyond machine accuracy and are rounded to 0. To reduce computational error, we use log probability instead, i.e.

$$\log p(C_j) + \sum_{i=1}^{n} \log p(F_i|C).$$

3. To incorporate the posterior score with users' weight, we multiply each feature likelihood $p(F_i|C)$ by the weight of (u, u_i) in the `user_graph`, $w(u, u_i)$. Remember that $w(u,\ u_i)$ describes the similarity of two users taste, i.e. the ratio of their agreements over disagreements. And since our features are defined in terms of u_i's, it totally makes sense to assign user weight to the feature likelihood when computing the posterior score.

However, the exact method of adding weights to feature probabilities remains uncertain. We experiment with two ways shown below

$$\log p(C_j) + \sum_{i=1}^{n} [\log w(u, u_i) + \log p(F_i|C)]$$

$$\log p(C_j) + \sum_{i=1}^{n} w(u, u_i) \cdot \log p(F_i|C).$$

There is no conclusive results. Detailed analysis will be shown in Section 4.

4. Sometimes we need to smooth out zeros when counting different situations, by adding 1 to both denominator and numerator.

4 Results

In this project, datasets come from GroupLens Research Data Sets [1]. Three datasets of different sizes are provided, each of which consists of at least 0.1 million ratings from over 1000 users on more than 1700 movies. Within each dataset, there is a partition of 80% for model training and 20% for model testing. The data has been preprocessed such that each user has rated at least 20 movies. Ratings are from 1 to 5. Here is a sample rating formatted as `user id`, `item id`, `rating` and `timestamp`:

1 43 4 878542869,

i.e. user 1 gives movie 43 a rating of 4 out of 5. The time stamps are unix seconds since 1/1/1970 UTC.

Also for each movie, there is a binary vector showing which type the movie belongs to among the total 19 categories such as action, comedy, drama and etc.

4.1 Graph Analysis

The experiments we carried out only use dataset of size 100k, which includes 943 users, 1682 movies and 100,000 ratings (due to computing power limit). We first examine how well are the communications among users, i.e. if they have rated common movies, based on bipartite clustering coefficients, Jaccard's coefficients and cycle clustering coefficients. The histograms are shown below.

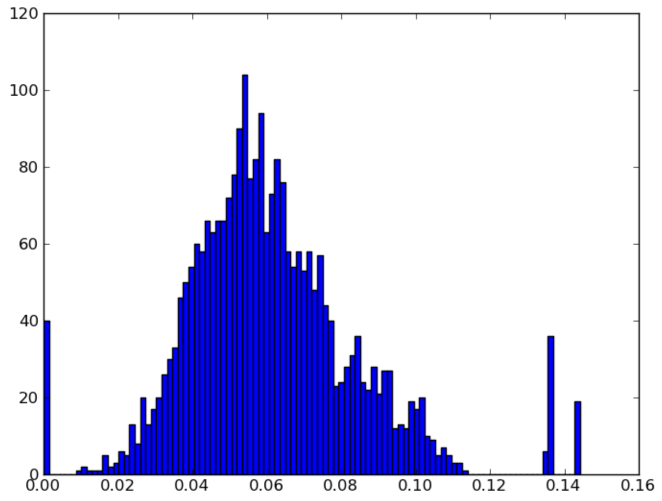

Figure 1: Clustering coefficients

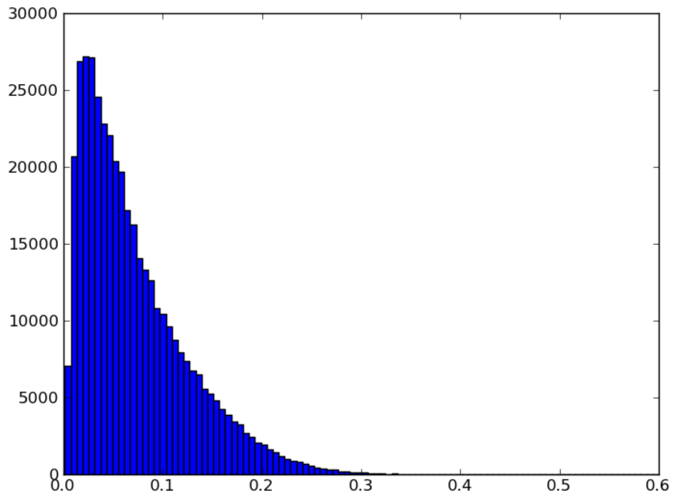

Figure 2: Jaccard's coefficients

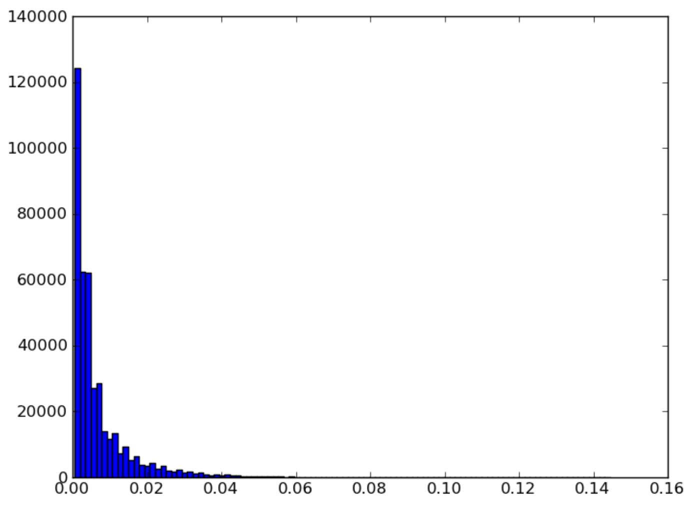

Figure 3: Cycle clustering coefficients

From the statistics above we can see that Jaccard's coefficients and cycle clustering coefficients are mostly small, Jaccard's $\in [0.0013793, 0.6]$ and cycle clustering $\in [0.00059453, 0.14447]$, which may indicate that different users do not share lots of common movies. Thus, it is unclear how well one user's taste or preference can be revealed from his neighbors tastes.

4.2 Learning Results

We use 5 pairs of datasets provided by GroupLens Research [1], each of which is a disjoint partition of 80% for model training and 20% for model testing. After learning and predicting, we count the true positives, real positives, predicted positives and errors from our learning algorithm. Note that "real positive" are the actual positives in the testing set and "true positive" are both actual positives in testing set and are predicted to be positive as well. "predicted positive" is self-explanatory. The statistics of 5 tests are shown below.

	True Positive	Real Positive	Predicted Positive	Error
Test 1 (weighted)	4766	8090	8352	6910
Test 1	3741	8090	6746	7354
Test 2 (weighted)	5619	8089	9425	6276
Test 2	4564	8089	7812	6773
Test 3 (weighted)	5757	8194	9683	6363
Test 3	4684	8194	8028	6854
Test 4 (weighted)	5571	7880	9580	6318
Test 4	4469	7880	7897	6839
Test 5 (weighted)	5376	7857	9451	6556
Test 5	4354	7857	7819	6968

The table above shows the results of both weighted Naive Bayes and unweighted as discussed in Section 3.3. We said there were two ways of adding weights to feature likelihoods. It turns out that adding log-weight to the posterior score makes approximately no difference than unweighted algorithm. So results of log-weight algorithm are omitted.

At first glance, it seems that weighted algorithm has a significantly higher "True Positive", which is preferable. However, if we examine carefully, we notice that it also has a higher "Predicted Positive", which indicates that weighted algorithm tend to recommend more movies than unweighted algorithm, regardless of correctness. To evaluate which of the two models is better, we introduce the classic set of metrics commonly used [2].

$$\begin{aligned} Precison &= \frac{\text{True positive examples}}{\text{Predicted positive examples}} \\ Recall &= \frac{\text{True positive examples}}{\text{Real positive examples}} \\ F\text{-}measure &= 2 \times \frac{Precision \times Recall}{Precision + Recall}. \end{aligned}$$

The results of the two models are re-calculated as followed. Note that total number of ratings in the testing set is 20,000. We also compute the error rate as Errors over 20,000.

	Precision	Recall	F-measure	Error Rate
Test 1 (weighted)	0.57	0.59	0.58	0.34
Test 1	0.55	0.46	0.50	0.37
Test 2 (weighted)	0.60	0.69	0.64	0.31
Test 2	0.58	0.56	0.57	0.34
Test 3 (weighted)	0.59	0.70	0.64	0.32
Test 3	0.58	0.57	0.57	0.34
Test 4 (weighted)	0.58	0.71	0.64	0.31
Test 4	0.56	0.57	0.56	0.34
Test 5 (weighted)	0.57	0.68	0.62	0.32
Test 5	0.55	0.55	0.55	0.35

From this table, we can see that even though the differences are not much, weighted algorithm always has a better precision, recall and F-measure and a smaller error rate than the plain unweighted algorithm. This indicates that neighbors with common tastes do have a positive effect on predicting recommendations. Moreover, the error rates of all tests lie between .30 to .35 which means up to 70% of all our predictions of both "like" and "dislike" are correct. This seems to be a satisfying result.

5 Further Discussion

In this section, we discuss a few thoughts about this project. The original motivation of this project was to predict “likes” or “dislikes” based on users social network. Friends’ ratings will be assigned higher weights than strangers’. However, most of the movie rating data available do not involve users personal social network. Thus, in order to assign weights appropriately to different user opinions, we build a weighted `user_graph` based on users’ movie watching history and common ratings. And hopefully, our neighbors on the `user_graph` could provide enough information on our own tastes or preference.

1. In Section 4.1, we show the histograms of Jaccard’s coefficients and cycle clustering coefficients. These statistics are defined to reveal how much overlap of movies that different users have both rated. The more overlap in the data, the more we think that users have communicated via common movies they watch and thus, the more desirable to our model idea. However, histograms show that these coefficients are quite low. Thus, it is possible that the dataset we use is not sufficient enough for predicting one user’s taste from other users.

2. Since we aim to find out potential interesting movies based on other people’s opinion, we care less about the attributes of the movie itself, such as type, cast or production. It is quite possible that we share difference opinion with our friends towards a certain movie. In this case, if a movie is liked by me but hated by several friends, this model would tend to give a “dislike” to it. This would also cause errors in Section 4.2, because the model does not take into account how I actually think about the movie. Therefore, this model is meant for a movie that I have not seen before. Thus, the predictability is limited in that we design our feature vectors only based on other people’s opinions. To fix this, adding more features about the movie’s intrinsic attributes may provide a more precise prediction.

3. Adding weight to Naive Bayes helps the correctness of prediction. One possible improvement is to find another way to add those weights so that they can have more influence on the scores.

4. When preprocessing the dataset, we simply cut all ratings by the threshold 3, higher to be “like” and lower to be “dislike”. Ratings of 3 are discarded. One situation happens is that some users show more mercy and biased opinions over all movies, i.e. they tend to give very few “dislike”s (rating of 1 or 2) and a lot more “like”s (rating of 4 or 5). For these users, threshold 3 is probably not their “indifferent” level of attitude. To make it more sophisticated, regarding each user, we compute his average rating and set his average to be his threshold. Thus, each user has his own threshold for “like” and “dislike”. This should also help the precision of prediction.

References

[1] http://www.grouplens.org/taxonomy/term/14

[2] N. Benchetta, R. Kanawati, and C. Rouveirol, Supervised machine learning applied to link prediction in bipartite social network. Social Network Analysis and Mining. International Conference on Advances in, 2010.

[3] Z. Huang and D. Zeng, Why does collaborative filtering work? - Recommendation model validation and selection by analyzing bipartite random graphs. Workshop of Information Technologies and Systems, Las Vegas, NV, 2005

[4] J. Leskovec, D. Huttenlocher, J. Kleinberg. Signed networks in social media. In Proc. CHI, 2010.

[5] J. Leskovec, D. HuttenLocher, J. Kleinberg. Predicting positive and negative links in online social networks. In Proc. WWW, 2010.