Inferring Social Networks Based on Movie Rating Data

Chaofei Fan
Department of Computer Science
Stanford University, CA 94305
stfan@stanoford.edu

Le Yu
Department of Computer Science
Stanford University, CA 94305
billyue@stanford.edu

***Abstract*—Social network analysis has been a hot research area recently thanks to the availability of many large dataset. One of the problems in social network analysis is to recover the underlying network structure given information about how nodes in the network interact with each other in the past. In the context of social movie website (e.g. Flixster[1]), if we know when users rate movies, can we find an optimal network of users that best explain the propagation of information which influences their movie watching behavior? Unveiling such a network would be useful to identify how product information is spread among users. Moreover, it is useful for recommendation of new products to users since recent studies have pointed out that using social network information could potentially improve the accuracy of recommendation. But many movie website such as Netflix[2] do not have built-in social network to take this advantage. Our contributions are three-folds: (1) We analyze the temporal movie rating data from Flixster; (2) We propose a social network inference algorithm MOVINF, which is more accurate and efficient; (3) We evaluate the inferred network using recommendation ('Netflix Challenge'), which outperforms baseline by 1.41% and is almost as good as using original social network.**

I. Introduction

The diffusion of information is one of the fundamental processes taking place in networks [1]. As one piece of information propagates over the network, it forms an information diffusion cascade: a set of nodes "infected" with that piece of information and their "infected" time. But usually we only observe the cascades without knowing the underlying network over which information propagates. The problem is thus to uncover the real social network given the observed cascades.

[1]http://www.flixster.com
[2]http://www.netflix.com

In online social networks, connected users influence each other. For example, in social movie website such as Flixster, user can see what movies their friends have watched recently and decide to watch the same movie. As in Figure 1, user's movie watching behaviour influences their friends, which forms a special kind of information diffusion cascade: *recommendation cascade*. Given these recommendation cascades, we would like to use them to uncover the underlying social network over which they propagate. On one hand, it is interesting to solve this problem because we can know from the result that how actively users tend to recommend movies to their friends. On the other hand, it is also useful in terms of recommendation as previous works have pointed out that using social network can potentially improve the accuracy and personalization of recommender system [2], [6], but many social movie website such as Netflix do not have built-in social network to take this advantage. This paper is structured as follows:

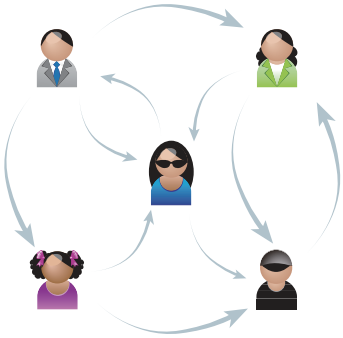
1) we formulate our problem as a network inference problem.
2) we introduce the original NETINF algorithm and explain why it does not fit in recommendation cascades.
3) we analyze features of the dataset and present the MOVINF algorithm.
4) we evaluate MOVINF and show that it can be 26x faster than NETINF and the social network it inferred is as good as the original social network in terms of recommendation.

II. Related work

Our problem is a special case of the network inference problem. Gomez-Rodriguez et al. have developed an effective algorithm for the general network inference problem [1]. They formulate a generative proba-

	item					
user	1			3		
		3			2	
		4		4		
	4					1
		5			5	
			3			5

(a) Ratings in Recommender Systems



(b) Social Network

Figure 1. Recommender Systems and Social Network.

bilistic model of how information is spread through blog and media sphere. However, in the context of online movie rating, the rating cascade does not capture as much underlying network structure as in blog and media sphere. We discuss the difference between information propagation on social movie website and blog and media sphere in section IV.

We can also formulate our problem as a link prediction and solve it using supervise learning (e.g. [9], [8]). But this does not solve the problem when a social movie website does not have a built-in social network.

Other works focus on analyzing the recommendation cascade patterns in online social network. Leskovec et al. analyze the influence patterns in recommender system [3]. Their work focuses on given a real social network, how does information propagate through network. In this paper, we do the opposite, that is, given the information propagation, how can we infer the underlying social network.

After predicting the social network, we also evaluate the network with social recommendation[5],[6]. These papers incorporate the social network into recommendation and reached a better performance on predicting ratings. We use the social recommendation framework to evaluate how well our predicted network is.

III. Problem Formulation

We now formally describe the problem where movie rating cascades propagate over an unknown user network. We observe when a user rates a movie but we do not know who or what influences this user to rate the movie. The problem is then to infer the unknown user network over which cascades propagate. We treat all the ratings for one movie as a big cascade.

A cascade c contains all the ratings for a particular movie m_c. Each rating r_i for movie m_c can be represented in the form of $(u_i, t_i)_c$ which means that user u_i rates movie m_c at time t_i. We want to infer a social network between users $\hat{G}$ that is as close as possible to the real network G^* on which information propagates. Note that G^* is not necessarily the original social network since information could propagate through channels (e.g. shared interests) other than friend links.

We define the transmission probability of cascade c as $P_c(u, v)$, which represents the probability that node (node means user) u influences node v in the cascade c. We use $P(c|T)$ as the probability that the cascade c propagates in a tree T. The tree pattern T describes which node might propagate to another node. Finally, we refer to $P(c|G)$ as the probability that network G contains the cascade c.

IV. Movie Network Inferrence Algorithm

In this section, we will first introduce the NETINF[1]. To improve the time complexity and precision, we proposed a MOVINF algorithm based on NETINF to predict the social network using recommendation cascades.

A. NETINF Model

Gomez-Rodriguez et al. develop an algorithm NETINF to infer influence network based on a generative probabilistic model [1].

The transmission probability $P_c(u, v)$ represents that node u influences node v. Therefore, given the tree structure T, the probability of cascade c can be calcu-

lated by bayesian rules, we have

$$P(c|T) = \prod_{(i,j)\in T} P_c(i,j), \tag{1}$$

where $P(c|T)$ is the probabilities of each edges in tree T, which shows how the cascade propagates. We can compute the probability $P(c|G)$ of a cascade propagates in graph G, based on $P(c|T)$. We can search all possible trees structure T in the tree set $T(G)$, where $T(G)$ is the undirected spanning trees on graph G. Here we follow the assumption in [1] that $P(T|G)$ follows the uniform distribution

$$\begin{aligned} P(c|G) &= \sum_{T\in T(G)} P(c|T)P(T|G) \\ &\propto \sum_{T\in T(G)} \prod_{(i,j)\in T} P_c(i,j) \end{aligned} \tag{2}$$

The probability of a cascades C in graph G can be presented as

$$P(C|G) = \prod_{c\in C} P(c|G) \tag{3}$$

With the diffusion network C, we would try to find a graph G with at most e edges that can maximize the probability of the social network:

$$\hat{G} = \arg\max_{|G|\leq e} P(C|G) \tag{4}$$

NETINF is an approximation greedy algorithm that is guaranteed to achieve the precision of at least 63% of the optimal solution. It also uses lazy evaluation and local update to achieve two orders of magnitude of speedup without any loss in solution quality. NETINF can achieve very good results if the underlying model is information cascade. This model assumes that each cascade has a single start node (i.e. the origin of the information), and with probability $p(\Delta t)$, it will transmit the information to its followers. The transmission probability is a function of time Δt, i.e., the longer the time difference between two nodes displaying the same piece of information, the less likely they are connected.

However, NETINF is not suitable for movie cascade analysis. In our case of movie rating cascades, we treat all ratings of a movie as a single big cascade. This big cascade is possibly made of many small cascades since in [3] Leskovec et al. find that recommendation cascades in e-commerce website tend to be shallow, i.e., one-to-one recommendations consist more than

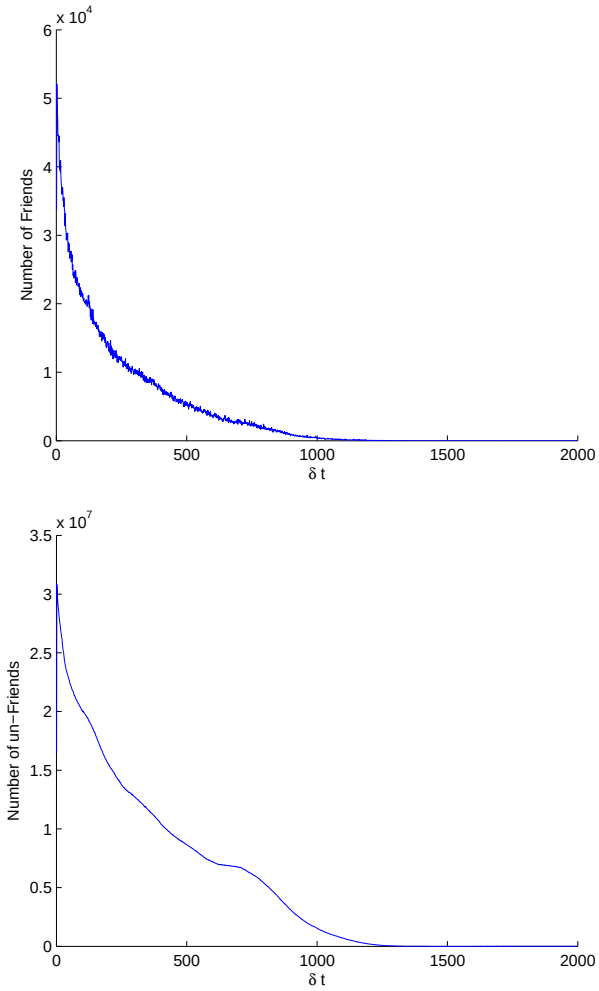


Figure 2. Friend/un-Friend distribution

70% of total recommendations. Here one-to-one recommendation means cascade of two nodes. Thus it is very unlikely that the first user who watch a movie would influence all the subsequent users who watch the same movie. On the contrary, it is more likely that the majority of users watch the movie because they see the advertisement of the movie or they like that movie genre.

- The first assumption in NETINF that a cascade has a single start point does not satisfy in our case. Moreover, we find out that in our dataset, 34% friends have rated at least one movie together. Even if NETINF performs very well, we could only expect to recover 34% of the original friend relations. Also, Flixster offers weak social feature. It is not very easy for users to see what their

friends have watched. As a result, if we treat all ratings of a movie a single cascade, we can only expect a very small fraction of that cascade are actually made through friend recommendations.

- The second assumption of transmission probability holds in our case of movie rating cascades. We calculate $\Delta t = t_j - t_i$ for all ratings pairs r_i and r_j for the same movie (Figure 2). We find the distribution of follows the power-law distribution, i.e., the shorter the Δt between two ratings, the more likely the two users of the ratings are friends. We also notice that the number of pairs of users who are not friends but rate a movie with time difference Δt is significantly higher than the number of pairs who are friends. This conforms with the hypothesis that very small fraction of a movie rating cascade are actually made through friend recommendations.

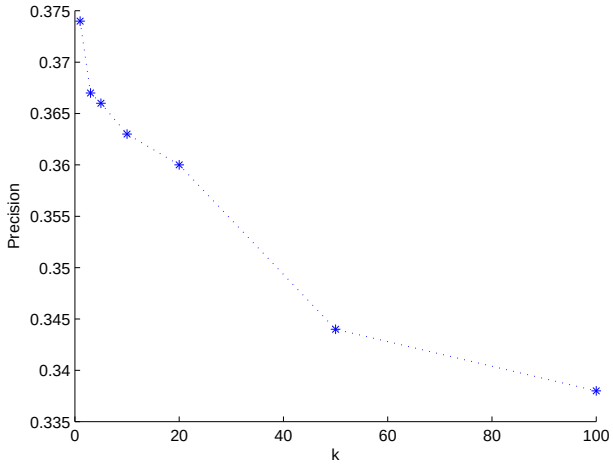
B. MOVINF model

Given the analysis above, it seems unlikely that the original NETINF could infer the original network in our case. What it infers is *general friendship* between users. For example, if two users tend to rate movies together within a short time period quite often, NETINF will link them together even if they are not friends in the real social network. But such links are useful because they indicate that a shared interest between these two users. We call such link as *general friendship*.

Definition 1 (general friendship)**.** *General friendship does not need to be real friendship. It refers to two users who have similar interests or hobbies.*

Identifying general friendship is useful in terms of recommendation because almost all recommender system aims to find users of similar interests as their first step. Thus it is interesting to try the original NETINF on movie rating cascades and evaluate it in terms of recommendation accuracy.

An interesting observation about recommendation gives us possibility to optimize the original NETINF. Recommendation is a special form of information cascade. Leskovec et al. find that shallow recommendation cascades in e-commerce website consist more than 70% of total recommendations [3]. Since the real cascade information is not available, we cannot do the same cascade pattern recognizing as in [3]. But we can



f

Figure 3. Precision theory on k Nearest Neighbor

estimate the average number of friends user u have in k users $u_1, ..., u_k$ who rate the same movie earlier than u. If $k = 1$, it means that we assume all recommendation cascades are one-to-one. These k users are called k nearest neighbor of user u.

Definition 2 (k Nearest Neighbor)**.** *In a certain cascade c, if user i and user j are both in the cascade, and there are less then k users between their timestamp in the cascades, then user i and user j are k Nearest Neighbor.*

Based on the conclusion in [3], we can modify NETINF to only consider nearest ratings in terms of time.

In Figure 3, we see that as k increases, the average number of friends decreases. This indicates that considering larger cascades would not give us advantage of finding more friends. Thus we limit the possible edges NETINF to the k nearest edges. This modification to the original NETINF reduces the number of possible edges in a cascade from $O(n_c^2)$ to $O(kn_c)$, where n_c is the number of nodes in cascade c.

Based on the analysis above, our model cares about the cascade propagates between k nearest neighbor. Therefore, we ignore the influence on user u if the influencing source user j is not u's k nearest neighbor. Here we remove those edges in the cascade c.

Therefore, the probability that the new cascade c propagates in a tree pattern T in Equation 1 can be

presented as

$$P(c|T) = \prod_{(i,j)\in T} P_c(i,j), \tag{5}$$

where j is k nearest neighbor to user i in cascade c.

Algorithm 1 MOVINF Algorithm

Require: C, e
$G \leftarrow \bar{E}$;
for all $c \in C$ **do**
  $T_c \leftarrow dagtree(c)$;
end for
while $|G| < e$ **do**
  for all $(j,i) \in C \backslash G$, where j is the k nearest neighbor of user i in cascade c **do**
    $\delta_{j,i} = 0, M_{j,i} \leftarrow \emptyset$;
    for all $c : (j,i) \in c$, where j is the k nearest neighbor of user i in cascade c **do**
      $define\ w_c(m,n)\ as\ weight\ of\ (m,n)\ in\ G \cup \{(j,i)\}$;
      if $w_c(j,i) \geq w_c(Par_{T_c}(i), i)$ **then**
        $\delta_{j,i} = \delta_{j,i} + w_c(j,i) - w_c(Par_{T_c}(i), i)$;
        $M_{j,i} = M_{j,i} \cup \{c\}$;
      end if
      $(j^*, i^*) \leftarrow argmax_{(j,i)\in C\backslash G} \delta_{j,i}$;
      $G \leftarrow G \cup \{j^*, i^*\}$;
    end for
  end for
  for all $c \in M_{j^*,i^*}$ **do**
    $Par_{T_c(i^*)} \leftarrow j^*$;
  end for
end while
Return G

V. Experiment

In this section, we analyze the inferred social network using MOVINF described in Section IV. We then apply the predicted network to improve recommendation.

A. Datasets

We test the proposed methods on the public-domain recommendation datasets with timestamp. Flixster [4] is a social networking service in which users can rate movies, and connect with other users. It consists of 1,049,511 users who have rated a total of 66,726 different items. On average, each users watched 19.5 movies, and have 8.9 friends.

Here each movie is referred to as each cascade. We sort each cascade with respect to the timestamp indicating when users watched the movies. First we select 1000 densely connected and active users from millions of users, meaning these users have the largest number of ratings as well as friends. The motivation is that we would like to ignore the sparsity problem of socal network and focus on evaluating the effectiveness of MOVINF. We also remove movies rated by less than users since they are not cascades, resulting in 16,333 movies(cascades) left. We use those cascades for social network prediction.

B. Evaluation Metrics: Precision and Recall

We run MOVINF on the selected users and their ratings. There are 1000 users and 21740 edges between them. On average, each user has 21 friends. These 1000 users have rated 16333 movies. We run both NETINF and MOVINF to infer the social network of users G^* with 21740 edges, and we set the parameter k for k nearest neighbour in MOVINF to 1. We measure the precision (i.e. the percentage of predicted edges in G^* that are also in the original social network G) and recall (the percentage of edges in G that are also in G^*). A higher precision and a higher recall represent a better performance.

Table I
Precision and recall of MOVINF, NETINF, and random guess

Method	Precision	Recall	Time
MOVINF	4.38%	4.38%	53s
NETINF	2.85%	2.85%	22m55s
Random	2.17%	2.17%	N/A

In Table I, NETINF is 31.3% better than random guess while MOVINF achieves 101% better performance. Even though MOVINF has relatively better performance compared with NETINF and random guess, it can only predict 4.3% of the original network, which is far less satisfactory. But note that as we mention in Section IV, due to the large number of irrelevant ratings in a big movie cascade, MOVINF tend to infer general friendships. If MOVINF links two users together, they could either be friends or they just have similar movie taste. As a result, it is not surprising that MOVINF gets low precision and recall.

Meanwhile, if we compare the inferred network with a subset of the original network, which consists of users who have rated at least 400 movies together, we find that the recall is 40%. This means that MOVINF can be accurate if two users who are friends rate movies together very often. In the next part, we show that the general friendship inferred by MOVINF is useful. Note that since MOVINF reduces the number of possible edges in cascade c from $O(n_c^2)$ to $O(n_c)$, MOVINF is $26X$ faster than NETINF in this setting (Table I).

Now we turn to the effect of parameter k in MOVINF. As identified earlier, we expect that as k increases, the precision and recall decrease. In this experiment, we fix the number of edge in inferred network to be 21740, and vary k from 1 to 20. The results are in Figure 4. We see that both precision and recall decrease monotonically as k increases. This conforms with our theoretical analysis in Figure 3. By reducing the possible edges in a cascade, we not only increase precision and recall but also reduce the execution time.

C. Evaluation Metrics: Social Recommendation

Recommendation is a heated topic in data mining, and one of the most competition is Netflix Challenge. The challenge is to build a system to predict user ratings for films based on previous ratings. Recently, social recommendation[5] utilizes the social network to improve the recommendation. The social recommendation is made, not only based on the previous rating history of a user, but also based on his/her friend. In this section, we will describe social evaluation methods.

1) Matrix Factorization[5]: One of the effective ways for recommendation is to factorize the user-item rating matrix, and make prediction on user-feature matrix and item-feature matrix. Considering a rating matrix matrix $R = (R_{ij})_{M\times N}$, where there are M users and N items. R_{ij} is the ratings given by user u_i on item v_j. Matrix factorization method can be approximate the rating matrix R by

$$R \approx U^T V, \tag{6}$$

where $U = (U_{id})_{M\times D}$, $V = (V_{jd})_{N\times D}$, $D < min(M, N)$. Matrix U and V can be regarded as the user latent matrix and item latent matrix. We define $I = (I_{ij})_{M\times N}$, if $R_{ij} \neq 0$, $I_{ij} = 1$; else $I_{ij} = 0$. Then

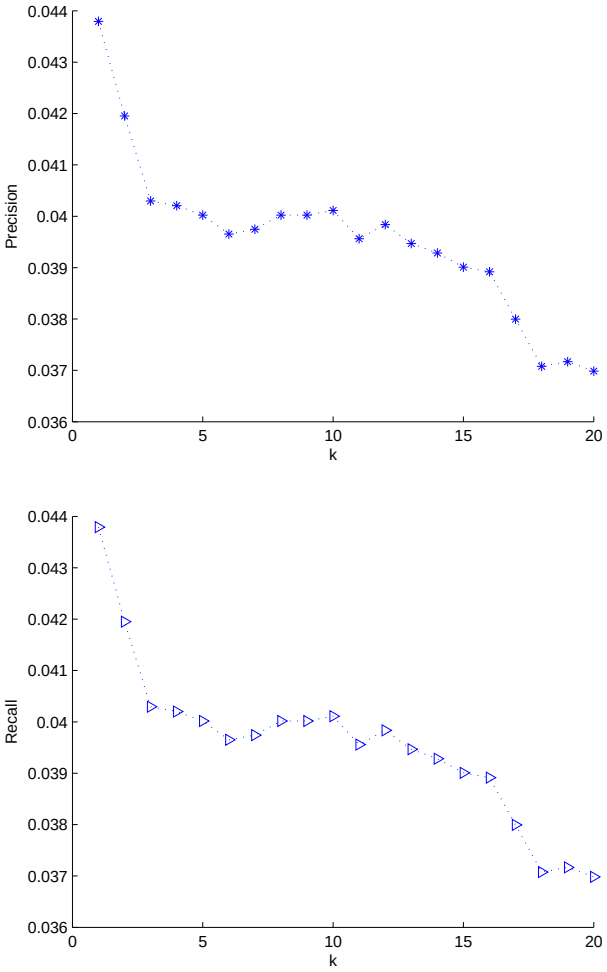


Figure 4. Precision and Recall on k Nearest Neighbor

we can rewrite the objective Loss funcion as

$$L(U, V) = \sum_{i,j} I_{ij}(R_{ij} - U_{i.}V_{j.}^T)^2. \tag{7}$$

To prevent overfitting problems, a regularization term is appendded on the loss function,

$$\begin{aligned} L(U, V) &= \sum_{i,j} I_{ij}(R_{ij} - U_{i.}V_{j.}^T)^2 + \\ &\quad \lambda_1||U||_F^2 + \lambda_2||V||_F^2, \end{aligned} \tag{8}$$

where $||.||_F$ denotes the Frobenius norm.

2) Social Matrix Factorization[5][6]: In the social age, people will ask friends for recommendation. Uses' taste might be close to their friends' tastes. Therefore,

we add a social regulizer to the loss function:

$$\begin{aligned} L(U,V) &= \sum_{i,j} I_{ij}(R_{ij} - U_{i.}V_{j.}^T)^2 + \\ &\quad \lambda_1||U||_F^2 + \lambda_2||V||_F^2 + \\ &\quad \frac{\beta}{2}\sum_{f\in F(i)} Sim(i,f)||U_i - U_f||_F^2. \end{aligned} \tag{9}$$

Here we impose a social regulizer term to Eq. (8) to constrain user's interest. β is the influencing impact of social network. $F(i)$ is the friends of user u_i. Taste difference between two friends can be described as $Sim(i,f)||U_i - U_f||_F^2$: U_i means the feature of user u_i. If user $u_i's$ interest does not like $u_i's$ friend u_f, and their characters might be more different and $||U_i - U_f||_F^2$ will be larger. The friends' overall opinions are to combine all the friends' interest. However, user u_i might not treat every friend equally, user u_i might trust u_f^1 more than u_f^2. u_i will much more follow the taste of u_f^1. So $Sim(i,f)$ depicts the similarity between u_i and u_j.

Our model is to minimize the loss function. By performing gradient descent on U_i and V_j for user i and item j, we obtain

$$\begin{aligned} \frac{\partial^2 L}{\partial U_i} &= \sum_{j=1}^{N} I_{ij}(R_{ij} - U_{i.}V_{j.}^T) + \lambda_1 U_i + \\ &\quad \beta \sum_{f\in F(i)} Sim(i,f)(U_i - U_f), \end{aligned} \tag{10}$$

$$\frac{\partial^2 L}{\partial V_j} = \sum_{i=1}^{M} I_{ij}(R_{ij} - U_{i.}V_{j.}^T) + \lambda_2 V_j. \tag{11}$$

3) Evaluation Measurement: We use 5-fold cross validation to estimate the performance of different algorithms. In each fold, the validation datasets are divided into train sets and test sets randomly. The training set contains 80% examples and the other 20% elements of the matrix are treated as unknown.

The evaluation metric we use in the experiment is the Root Mean Square Error (RMSE). The metrics RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i,j}(R_{ij} - \hat{R}_{ij})^2}{T}}, \tag{12}$$

where $\hat{R_{ij}}$ and T means predicted score for user u_i on item v_j and pairs number of (i,j) in the test set. Notice smaller RMSE value means a better performance.

D. Comparisons

In this section, we compare the recommendation results of the following algorithms:

- PMF: Probabilistic Matrix Factorization. This method is proposed by Salakhutdinov and Minh in [7]. It only uses rating matrix to recommend. It represents the collaborative filtering algorithms.
- SR: Social Recommendation, which is proposed in [5]. It models users' rating based on social relationship. We use it as the social baseline. The social network is provided in the Flixster datasets.
- $NETR$: NETINF Recommendation. We utilize the NETINF in [1] to infer the social network, then we apply the inferred social network for recommender system.
- $MOVR$: MOVINF Recommendation. We use the MOVINF method to infer the social network, which is proposed in Section IV. and use the inferred social network for recommender system.

There are three main parameters in the experiments. λ_1 and λ_2 are the regularization parameters which are determined with cross-validation. In all of the algorithms, we set $\lambda_1 = 0.1$ and $\lambda_2 = 0.1$. Another parameter β controls how much influence should social networking impose on the social network. The $\beta = 0.1$ in the latter two algorithms. The result of feature dimension $D = 5$, $D = 10$ are shown in Figure 5.

As we can see from Table II, when $D = 5$, $MOVR$ improves RMSE of by 1.41% compared with PMF, while $MOVR$ improves RMSE of by 0.52% compared with $NETR$. The result of MOVR is almost as good as SR, which uses the real social network. This means that MOVINF can effectively identify users sharing the same interests. Although the social network inferred by MOVINF differs a lot from the original social network, they are equivalently effective in terms of recommendation.

Table II
RMSE ON DIFFERENT DATA SET

Dimension	PMF	SR	$NETR$	$MOVR$
D = 5	0.9868	0.9728	0.9780	0.9729
D = 10	0.9908	0.9762	0.9801	0.9770

VI. CONCLUSION

In this paper, we try to solve the problem of inferring a social network given movie rating data. We

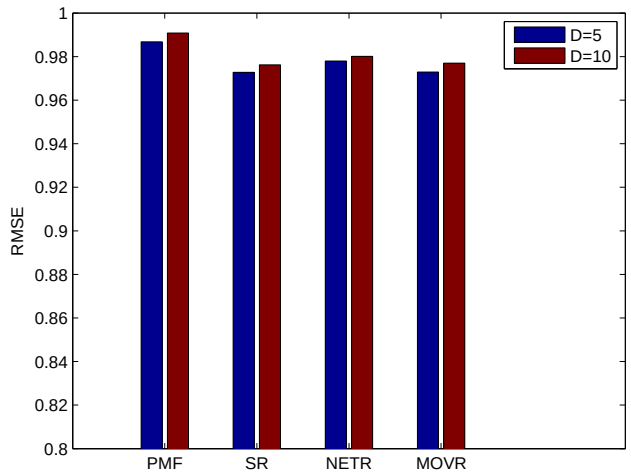


Figure 5. Comparison on RMSE

analyze our dataset and propose an optimized network inference algorithm MOVINF based on NETINF [1], which can be 26x faster and 50% more accurate than the original NETINF. Moreover, since we identify that movie rating does not satisfy the model of information cascade, we instead turn to evaluate the inference algorithm based on recommendation. Our results show that the inferred network can be as good as the original social network in terms of recommendation. This result is very interesting because other websites that do not have built-in social network can use MOVINF to infer a shared interest network of users and take advantage of those social recommendation algorithms.

As for future work, we would like to evaluate our algorithm on dataset without social network (e.g. MovieLens, Netflix) and compare the recommendation accuracy with PMF. Also, there a lot of opportunities to refine the model of MOVINF. For example, because we know that most recommendations are one-to-one, we can partition a big cascades into many small ones.

REFERENCES

[1] Gomez-Rodriguez, M., Leskovec, J., & Krause, A., *Inferring Networks of Diffusion and Influence*, arXiv.org, 2010.

[2] Jamali, M., & Ester, M., *Trustwalker: a random walk model for combining trust-based and item-based recommendation* Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 397-406, 2009.

[3] J. Leskovec, A. Singh, J. Kleinberg, *Patterns of Influence in a Recommendation Network*, Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2006.

[4] *Flixster Dataset*, http://www.cs.sfu.ca/~sja25/personal/datasets/.

[5] H. Ma, D. Zhou, C. Liu, M. R. Lyu, I. King, *Recommender Systems with Social Regularization*, In Proceedings of ACM WSDM 2011.

[6] L. Yu, R. Pan, Z. Li, *Adaptive Social Similarities for Recommender Systems*, In Proc. 5th ACM Conference on Recommender Systems, 2011.

[7] R. Salakhutdinov and A. Mnih, *Probabilistic matrix factorization*, In NIPS 2008, volume 20.

[8] J. Leskovec, D. Huttenlocher, J. Kleinberg. *Predicting Positive and Negative Links in Online Social Networks*. In Proc. 19th WWW, 2010.

[9] Backstrom, L., Leskovec, J. (2010, November 17). Supervised Random Walks: Predicting and Recommending Links in Social Networks. arXiv.org.