A General Framework for Implicit and Explicit Social Recommendation

Chin-Chi Hsu, Mi-Yen Yeh, Shou-De Lin

Abstract—Research of social recommendation aims at exploiting social information to improve the quality of a recommender system. It can be further divided into two classes. Explicit social network recommendation assumes the existence of not only the users' ratings on items, but also the explicit social connections between users. Implicit social network recommendation assumes the availability of only the ratings but not the social connections between users, and attempts to infer implicit social connections between users with the goal to boost recommendation accuracy. This paper proposes a unified framework that is applicable to both explicit and implicit social network recommendation. We propose an optimization framework to learn the degree of social correlation and rating prediction jointly, so these two tasks can mutually boost the performance of each other. Furthermore, a well-known challenge for implicit social network recommendation is that it takes quadratic time to learn the strength of pairwise connections. This paper further proposes several practical tricks to reduce the complexity of our model to be linear to the observed ratings. The experiments show that the proposed model, with only two parameters, can significantly outperform the state-of-the-art solutions for both explicit and implicit social recommender systems.

Index Terms—Recommender Systems, Social Networks

1 Introduction

Social recommendation, a study aiming at incorporating social information of users into a recommender system, has attracted decent attention in recent years. It can further be divided into two tracks: *explicit social recommendation* and *implicit social recommendation*. In explicit social recommendation, a variety of models have been developed to exploit the existing social network information to enhance the performance of a recommender system [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. A common and arguably most successful strategy is to integrate the social information, such as *trust* or *friendship*, into a collaborative filtering model in a certain way. Figure 1 describes an explicit social recommendation system given edge strength information is available; while Figure 2 shows another kind of explicit social recommender system where only binary relationship information (e.g., whether two people are friends) is available. Suppose there is a rating dataset including some ratings of four users $\{U_1, U_2, U_3, U_4\}$ to four items $\{V_1, V_2, V_3, V_4\}$. Such data can be denoted by a matrix where the "?" entries represent unknown ratings. A social-based recommender system reads the matrix together with a given or inferred user social network as the training examples, and then predicts the unknown ratings.

The information of the strength of social relationship can be very useful to a recommender system, as it is reasonable to assume people trust the ratings from their closer friends comparing to those from their acquaintances. Given the rating data along with a binary social network, several works

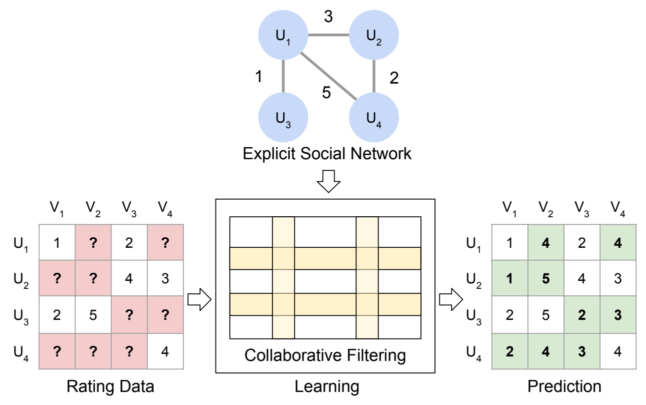


Fig. 1. Explicit social recommendation based on collaborative filtering techniques, assuming the social strength information(i.e. edge weight, scale of trust or friendship) is available.

[2], [9], [13] of explicit social recommendation have proposed methods to determine the social connection strength between users to enhance the quality of recommendation (see Figure 2).

Unfortunately, such trust or friendship data may not necessarily be available for every recommendation scenario due to the budget or privacy concerns. To address such concern, there emerges another research direction named *implicit social recommendation*, which aims at mining implicit user social relationship from historical rating data for better recommendation. Without any explicit social data, certain methods [14], [15], [16], [17] have been proposed to generate an implicit social network from given ratings (see Figure 3). The pseudo links and/or their strengths can then play as a surrogate of the explicit social network to be incorporated

- *C.-C. Hsu and M.-Y. Yeh are with the Institute of Information Science, Academia Sinica, Taipei, Taiwan*
E-mail: chinchi, miyen@iis.sinica.edu.tw
- *S.-D. Lin are with the Department of Computer Science and Information Engineering National Taiwan University Taipei, Taiwan*
E-mail: sdlin@csie.ntu.edu.tw

Manuscript received April 19, 2005; revised August 26, 2015.

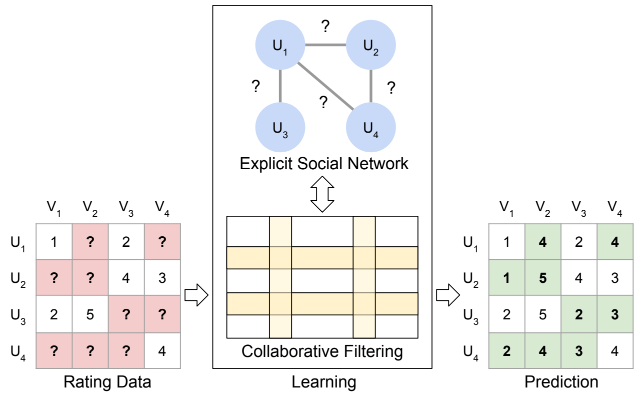


Fig. 2. Explicit social recommendation without social strengths between two connected users. Several existing solutions attempt to guess the social strengths using historical ratings to improve the overall performance.

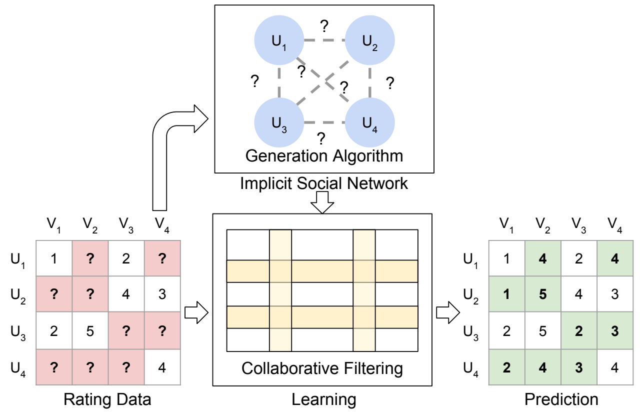


Fig. 3. Implicit social recommendation. A pseudo user social network is generated according to the rating data. It plays as the surrogate of the true social network for existing explicit social recommendation.

into any explicit social recommendation model.

The goal of this paper is to propose a unified framework to accommodate both scenarios described above. Furthermore, it aims to address the following concerns in the existing social recommender systems.

- Concerns for explicit social recommender systems: the quality of the given social information is sometimes questionable. Since most of the social data are collected from the web or social network services, inevitably they contain noises. For example, past empirical studies [4], [18] have shown that the auxiliary of *friendship* links is less useful than *trust* links in boosting the recommendation performance. Furthermore, although it is generally believed trust or friendship are positively correlated with the level of common-taste of people, this study [19] has shown that two users may not have similar rating tastes even they strongly trust each other. Thus, directly utilizing any given social connection may harm the recommendation performance.
- Concern for implicit social recommender systems: most related works attempt to empirically *define* ad-hoc metrics to generate an implicit social network, and then feed it into an explicit social recommendation model. Such strategy requires careful tuning of thresholds and parameters in the metrics for different domains and different rating scales, thus can hardly be generalized. As will be shown in our experiments later on, efforts spent to determine a common parameter that can be effective in inferring implicit social networks across different datasets are usually futile, thus hinders the effectiveness of such models.

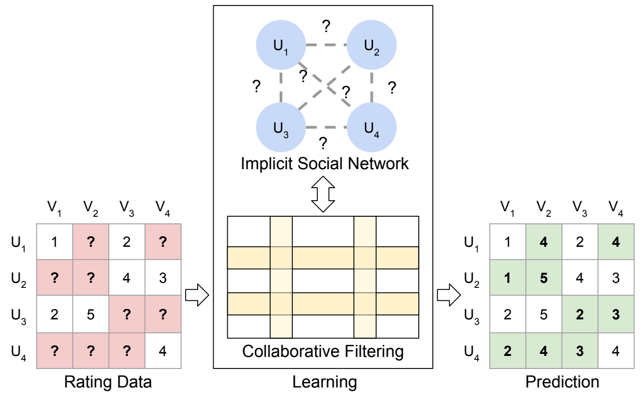


Fig. 4. The goal of our work in this paper. If there exists a binary social network from external source, then our proposed model works as Figure 2 illustrates. However, even without any given information of social structure, we would like to design a collaborative filtering model to automatically learn the underlying pairwise social connections from existing rating data. We hope that such social learning could boost the prediction accuracy of the recommender system.

To address the above concerns, we propose a general social recommender model applicable to the scenarios with or without an explicit social network, as shown in Figure 4. Given an explicit binary social network, our model learns the strength of links from rating data to boost the quality of rating prediction. When the explicit social network is missing, our model learns jointly the existence and strength of social relationships from ratings. Different from most of the previous solutions for implicit social network that treats the learning of the social network and recommendation as two sequential but independent tasks, we propose a Variational Expectation Maximization (VEM) based solution that conducts the learning of social structure and rating prediction together. By jointly learning the social relationship and ratings from data, our data driven solution can not only absorb the potential damage brought up by noisy explicit social network, but also alleviates the concern of certain ad-hoc, less-general rules for deriving implicit social networks. To improve the usability and scalability, we further describe some implementation tricks to avoid quadratic time and space complexity. Experiments show that the proposed solution outperforms the state-of-the-art models in both explicit and implicit scenarios.

Another major strength of our model is that it learns the importance of each latent feature to determine the similarity between users. Conventional social recommender systems tend to regularize the latent factors of users so that the users with social connections tend to have similar latent factors.

However, in the traditional setup, each latent dimension is assumed to contribute equally to the determination of similarity. We argue that this assumption is problematic since the "area of similarity' between every pairs of friends can be very different. For instance, A and B may become friends because they both like watching romance movies (assuming romance as one latent dimension), but it does not mean that A and B shall like action movies equally (assuming action represents another latent dimension). Similarly, friends A and C may have similar taste for action movies, but not necessary for other types of movies. The example shows that assuming two friends share similar latent vector (i.e. possess similar taste on all the latent dimensions) is an overly-strong assumption. In our model, such assumption is relaxed. Instead we let the model learn which latent dimension(s) best capture the similarity between two friends. In the previous example, our model can learn from data that A and B become friends because they have similar taste on romance movies, thus will only synchronize their preference on this dimension. We believe this is another major reason that our model can significantly outperform the competitors.

Another interesting finding we would like to share is that our solution can be applied to learn "implicit item relationship" as well. That is, we treat items as humans and learn their relationships for recommendation. Technically, our model learns which latent dimension(s) between two items should be aligned, and experiments show that considering such implicit item relationship in our model can further boost the recommendation quality significantly.

Finally, we share the omitted mathematical derivation as well as source code for reproducing the results in [20].

2 Related Work

As summarized in Section 1, prior works consist of either explicit social recommendation (using information from accessible external social networks) or implicit social recommendation (network topology and edge strengths inferred from the rating data). We review these two classes of works in this section.

Explicit social recommendation usually incorporates accessible social relations among users. In real-world applications, the relation of two users is usually defined by *trust* or *friendship*. Roughly speaking, trust indicates an one-way relation created by a user toward another. User A trusting user B does not necessarily imply B also trusting A. Conversely, friendship represents a mutual relationship between two users. User A making friends with user B implies A is also a friend of B. We refer readers to researches [18] and [4] that conduct experiments to compare the properties of these two types of relationships. Though some previous works rely on using the trust networks while others focus on friendship networks, we find that most of the proposed models are applicable for either case, regardless of whether directed trust networks or undirected friendship networks are exploited. Early solutions of explicit social recommendations [3], [5], [10] determine the recommended items by exploring the trust networks. Later on, researchers start to bring social information into the matrix factorization models with various assumptions social recommendation integration. SoRec [7] uses the shared user latent factors for both the rating matrix and the social matrix factorization. RSTE [8] predicts a user's ratings by the linear combination of the user and his or her trustees' latent factor vectors. SocialMF [6] defines that a user's latent factors should be close to the linear combination of his or her trusted friends' latent factors. Social Regularization [9] considers a pairwise assumption that two users trusting each other should have similar latent factors, and thus appends a regularization term to the classical matrix factorization model. There exist works modeling social influence based on collaborative filtering, such as conditional random field [11] and probabilistic Poisson factorization [1]. TrustSVD [4] incorporates the trust networks with SVD++, a variant of matrix factorization modeling the implicit influence from user latent factors through observed ratings. However, often an explicit social network is available but the edge strength information is missing or less believable due to privacy or the budget constraints. Some solutions are proposed to extract helpful edge strength features from the existing given ratings. For example, combining with SocialMF, Fazeli et al. [13] survey and compare the performance of different trust strength metrics on an explicit social network. Fang et al. [2] fuse support vector regression with matrix factorization to learn both ratings and strengths from an explicit trust network. Social Regularization [9] considers cosine similarity or Pearson correlation coefficient between two users as candidates of social strength definition. Compared with the above approaches, the main contribution of our model is that it obtains the strength of each individual social connection through *learning the importance of each latent dimension*. Therefore, our model is flexible enough to determine the different types of social relationships between users, and thus learns the importance of each individual connection.

Implicit social recommendation attempts to extract latent social relations between two users from the given rating behaviors. The generated information serves as the surrogate of the explicit social networks in the explicit social recommender systems. Despite the quadratic time complexity of evaluating pairwise user or item social relations, Guo et al. [14] study user-based collaborative filtering that recommends items using a trust network generated from the predefined trust metrics. With the existence of extra features, Lin et al. [15] (rating time) and Guo et al. [21] (text review) present methods to obtain implicit social networks. There are some works that apply matrix factorization techniques on implicit social networks. RSTE [17]/Social Regularization [16] read implicit social networks generated by the evaluation of cosine similarity/Pearson correlation together with predefined thresholds to determine the social connections. Comparing to the above models, the main strength of our approach is that *it models the rating prediction and the social strength learning as a joint optimization task.* These two tasks can mutually reinforce the quality of each other to achieve better results. Our approach is data-driven and thus does not need ad-hoc metrics or handcrafted thresholds to determine the existence/strength of social relationship.

3 Model Design

In this section, we would like to present our unified social recommendation model, named *Social Covariance Prior*

(SCP).

We first state the formal problem definition in Section 3.1, followed by the review of PMF in Section 3.2. Section 3.3 presents the overview of SCP. Since Section 3 and 4 introduce numerous notations for presentation, we list all these notations and their physical meanings again in Section 3.4.

3.1 Problem Definition

We are given *rating data* denoted by matrix $R \in \mathbb{R}^{N \times M}$, where N is the number of users and M is the number of items. An observed or non-missing entry R_{ij} records a numerical rating score that user i, $1 \leq i \leq N$, gives to item j, $1 \leq j \leq M$, as a training instance. Some datasets also provide an *explicit social network* to represent the social relations among users. An adjacent matrix $S^{(E)} \in \mathbb{R}^{N \times N}$ denotes the social network, where entry $S^{(E)}_{if}$ is binary (whether user i trusts or makes friends with user f). The goal of explicit social recommendation is to predict the unobserved values in R by incorporating both observed values in R and social information from $S^{(E)}$. For implicit social network recommendation, the goal is, even without $S^{(E)}$, to mine an *implicit social network* $S^{(I)}$ from R to enhance the prediction accuracy.

3.2 Probabilistic Matrix Factorization (PMF)

Since the proposed SCP model is an extension of probabilistic matrix factorization, we first briefly introduce PMF. Mnih et al. [22] proposed a probabilistic view of matrix factorization. Matrix factorization assumes a low-rank $R \approx U^T V$. That is, R can be approximated by the multiplication of user latent matrix $U \in \mathbb{R}^{K \times N}$ and item latent matrix $V \in \mathbb{R}^{K \times M}$ ($K \ll N, K \ll M$), where column vectors U_i, V_j reflect latent rating characteristics of user i and item j, respectively. Note that throughout this paper, we use K to denote the number of latent factors. The corresponding objective function is shown as follows:

$$\arg\min_{U,V} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} \delta_{ij} \left(R_{ij} - U_i^T V_j \right)^2 + \frac{\alpha_U}{2} \sum_{i=1}^{N} \|U_i\|_2^2 + \frac{\alpha_V}{2} \sum_{j=1}^{M} \|V_j\|_2^2 \quad (1)$$

$\delta_{ij} \in \{0, 1\}$ indicates whether R_{ij} is observed in the training data. α_U, α_V are regularization parameters. Under probabilistic view, an observed rating R_{ij} is generated by a normal distribution of mean $U_i^T V_j$ as the likelihood function. Corresponding to $l2$-norm regularization, zero-mean spherical Gaussian priors are used to generate U_i and V_j. Hence, the overall posterior to be optimized is

$$\arg\max_{U,V} p(U, V|R, \theta) = \arg\max_{U,V} \prod_{i=1}^{N} \prod_{j=1}^{M} \mathcal{N}(R_{ij}|U_i^T V_j, \sigma_R^2)^{\delta_{ij}} \prod_{i=1}^{N} \mathcal{N}(U_i|0, \sigma_U^2 I) \prod_{j=1}^{M} \mathcal{N}(V_j|0, \sigma_V^2 I). \quad (2)$$

$\mathcal{N}$ denotes a normal distribution, where the mean is the inner product $U_i^T V_j$ and the variance is σ_R^2. σ_U^2, σ_V^2 are shared variances among users and items, and I is the identity matrix. $\theta = \{\sigma_R^2, \sigma_U^2, \sigma_V^2\}$ denotes all the hyperparameters. Note that $\mathcal{N}$ for vectors U_i and V_j becomes multivariate normal distribution. PMF itself can be illustrated in Figure 5. A common solution to (2) is to take $(-\log)$ (then we obtain (1)) and perform optimization using stochastic gradient descent (SGD). The probabilistic view enables us to combine matrix factorization with other probabilistic models like Latent Dirichlet Allocation (LDA) [23] more naturally. Moreover, the Bayesian treatment such as Markov Chain Monte Carlo (MCMC) [24] and variational inference [25], [26] can then be applied.

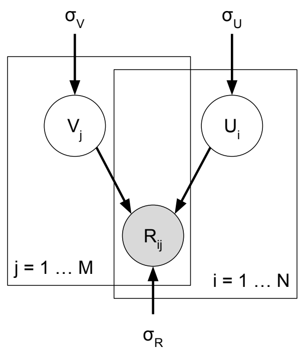


Fig. 5. Graphical model for Probabilistic Matrix Factorization (PMF).

3.3 Social Covariance Prior (SCP)

3.3.1 SCP for Implicit User Social Recommendation

We first describe a general version of our model that assumes the non-existence of the social connections, where the implicit relationship has to be inferred using the ratings.

The fundamental idea is that we propose a series of social-based priors to regularize the underlying pairwise distances among user latent factors:

$$\prod_{(i,f) \in E_U} \left[\mathcal{N}(U_i|U_f, S_{Uif}^{-1}) \mathcal{W}(S_{Uif}|K, \Lambda_U) \right]^{\frac{1}{T_{Ui}}}, \quad (3)$$

where E_U represents the set of directed edges in the user social network. Since an explicit social network is not available, here we assume E_U is a *fully connected graph*. T_{Ui} is the number of user i's neighbors (i.e., trustees or friends). Such prior will be incorporated into the original PMF equation to describe the user social network structure to be learned. Similar to previous works such as [9], SCP assumes that the social influence is implicitly transferred through the factorized matrix U. We model a pairwise social relation using a K-dimensional multivariate normal distribution $\mathcal{N}$ with the mean as neighbor f's latent factor vector, which regularizes the latent factors of social neighbors to be similar "to some extend" (i.e. rather than forcing the whole latent vector to be similar). The major difference from the previous works is that we introduce the personalized inverse covariance matrix S_{Uif} to fine-tune the similarity metrics between users. For each potential connection between persons i and f, we want to learn S_{Uif} to capture their similarity, or social edge strength (i.e. scale of trust or friendship), on the latent vectors U_i and U_f. The S_{Uif} is designed as a diagonal matrix to be learned, where the values of each diagonal element represent the variance of each dimension in the latent factor U_i. That says, S_{Uif} captures the strength of each latent dimension toward defining the edge strength. With such prior, we are now capable of modeling different "types" of social relationships, such as "A and B are friends because they are similar in latent dimension D1 and D2; while A and C are friends since they are similar in dimension D3, D4, and D5". Besides, to facilitate efficient learning, we choose Wishart distribution $\mathcal{W}$, the conjugate prior of precision matrix, with a fixed degree of freedom K and a scale matrix Λ_U shared by all S_{Uif} as regularization for S. As will be described later on, Λ_U will also be learned from data. Note that the zero-mean Gaussian spherical prior can be replaced with more general Gaussian priors such as those used in BPMF [24] or PPMF [25]. Nevertheless, here we apply the simple prior as the probabilistic view of $l2$-norm for fair comparison with previous social recommendation models widely using $l2$-norm regularization.

One major issue to address for implicit social recommender systems lies in the concern of efficiency. Because we need to learn S_{Uif} for a fully connected graph, it normally takes $O(N^2)$ time and space to evaluate all pairwise strengths between nodes. In Section 4.5.1, we will discuss how to address this issue.

3.3.2 SCP for Explicit User Social Recommendation

Explicit social recommendation can simply be treated as a special case in our model: when we are allowed to access explicit binary social networks together with rating data, we can simply define the edge sets E_U as the explicit social network. If the input network is undirected, each edge (i, f) is modelled as two directed edges $(i, f), (f, i) \in E_U$. Thus, instead of going through all pairs of nodes, if edge $(i, f) \notin E_U$ for users i, f, then we do not attempt to learn the similarity S_{Uif} and assume U_i is independent of U_f. Note that SCP does not utilize explicit edge strength even if it is provided. First, to our knowledge, there are very few publicly available explicit social network with edge strength information. Second, as mentioned previously, noisy edge strength could degrade the prediction performance, so we would rather learn it from data. Figure 6 marks the difference between incorporating an explicit social network and learning an implicit social network.

3.3.3 SCP for Item Social Networks

Since our model mines the implicit relationship between users, we can as well apply the same concept to mine the implicit relationship between items. Thus, the item social prior that is symmetric to (3) can be incorporated into our optimization procedure:

$$\prod_{(j,g)\in E_V} \left[\mathcal{N}(U_j|U_g, S_{Vjg}^{-1})\mathcal{W}(S_{Vjg}|K, \Lambda_V)\right]^{\frac{1}{T_{Vj}}}. \tag{4}$$

Similar to the user social prior, the item social prior captures the similarity between two items by aligning their latent factors. Furthermore, every two items can be "similar" in different aspects captured by latent dimensions. For instance, movie A and B are similar because there is overlapping between the main actors. On the other hand, movie C and D are similar because they are talking about a similar topic.

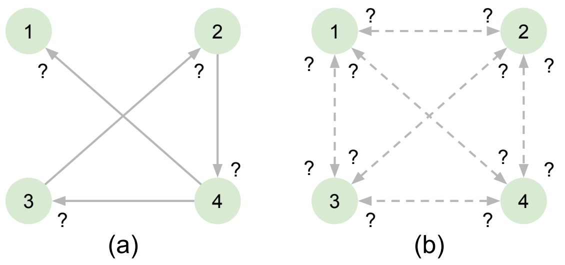


Fig. 6. Two scenarios of SCP learning a user social network. (a) If an explicit binary social network is accessible, then SCP learns the similarity (i.e. social strength) of each edge by rating data, regardless of external social strengths. (b) Without explicit social networks, SCP assumes that any pair of users have potential social relationships, and then learns every pairwise similarity. Note that one edge contains two directions to be learned different similarity values

3.3.4 Social Covariance Prior Overview

Integrating the components from previous sections, we now propose the general SCP model as follows:

$$\begin{aligned}\arg\max_{U,V,S_U,S_V} &\prod_{i=1}^{N}\prod_{j=1}^{M}\mathcal{N}(R_{ij}|U_i^T V_j, \sigma_R^2)^{\delta_{ij}} \\ &\prod_{i=1}^{N}\mathcal{N}(U_i|0, \sigma_U^2 I)^{1-b_U}\prod_{j=1}^{M}\mathcal{N}(V_j|0, \sigma_V^2 I)^{1-b_V} \\ &\prod_{(i,f)\in E_U}\left[\mathcal{N}(U_i|U_f, S_{Uif}^{-1})\mathcal{W}(S_{Uif}|K, \Lambda_U)\right]^{b_U\frac{1}{T_{Ui}}} \\ &\prod_{(j,g)\in E_V}\left[\mathcal{N}(V_j|V_g, S_{Vjg}^{-1})\mathcal{W}(S_{Vjg}|K, \Lambda_V)\right]^{b_V\frac{1}{T_{Vj}}},\end{aligned} \tag{5}$$

where $b_U, b_V \in [0, 1]$ are balance parameters. They control the regularization ratio between the social prior and the zero-mean Gaussian spherical prior. Later on in our experiment, we will show that setting $b_U = b_V = 1$ yields better results, meaning that the original Gaussian spherical prior can be ignored. Figure 7 displays the graphical structure of SCP.

3.4 Notations

In Section 3 and 4, we exploit numerous notations to denote variables of different usage. To raise the readability of our presentation, here we clarify and classify the notations.

- $N \in \mathbb{N}$: number of users
- $M \in \mathbb{N}$: number of items
- $K \in \mathbb{N}$: number of latent factors
- $R_{ij} \in \mathbb{R}$: rating of user i to item j
- $\delta_{ij} \in \{0, 1\}$: whether user i rates item j in the training rating data
- (i, f): directed edge from user i to f
- (j, g): directed edge from item j to g
- E_U: set of edges in the user social network

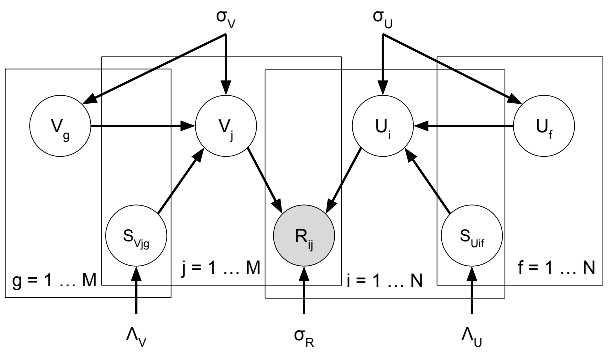


Fig. 7. Graphical model for Social Covariance Prior (SCP). In Section 4, we learn SCP using Variational Expectation Maximization (VEM) where E-step learns variational parameters θ' of random variables $\{U_i, V_j, S_{Ui}, S_{Vj}\}$ and M-step learns model parameters $\theta = \{\Lambda_U, \Lambda_V, \sigma_U^2, \sigma_V^2, \sigma_R^2\}$.

- E_V: set of edges in the item social network
- $T_{Ui} \in \mathbb{N} \cup \{0\}$: number of neighbors (either trustees or friends) of user i
- $T_{Vj} \in \mathbb{N} \cup \{0\}$: number of neighbors (either trustees or friends) of item j
- Hidden variables Z
  - $U_i \in \mathbb{R}^K$: Latent factor vector of user i
  - $V_j \in \mathbb{R}^K$: Latent factor vector of item j
  - $S_{Uif} \in (\mathbb{R}^+)^{K\times K}$: Similarity or social edge strength matrix from user i to f
  - $S_{Vjg} \in (\mathbb{R}^+)^{K\times K}$: Similarity or social edge strength matrix from item j to g
- Variational parameters θ'
  - $\lambda_{Ui} \in \mathbb{R}^K$: mean vector of latent factors of user i
  - $\lambda_{Vj} \in \mathbb{R}^K$: mean vector of latent factors of item j
  - $\gamma_{Ui} \in (\mathbb{R}^+)^{K\times K}$: covariance matrix of latent factors of user i
  - $\gamma_{Vj} \in (\mathbb{R}^+)^{K\times K}$: covariance matrix of latent factors of item j
  - $\Lambda_{Uif} \in (\mathbb{R}^+)^{K\times K}$: scale matrix of similarity or social edge strength from user i to f
  - $\Lambda_{Vjg} \in (\mathbb{R}^+)^{K\times K}$: scale matrix of similarity or social edge strength from item j to g
- Model parameters θ
  - $\sigma_R^2 \in \mathbb{R}^+$: rating variance
  - $\sigma_U^2 \in \mathbb{R}^+$: latent factor variance shared by all users
  - $\sigma_V^2 \in \mathbb{R}^+$: latent factor variance shared by all items
  - $\Lambda_U \in (\mathbb{R}^+)^{K\times K}$: similarity scale matrix shared by all edges in the user social network
  - $\Lambda_V \in (\mathbb{R}^+)^{K\times K}$: similarity scale matrix shared by all edges in the item social network
- $b_U \in [0,1]$: user balance parameter
- $b_V \in [0,1]$: user balance parameter

4 Model Learning

In this section, we will discuss how the variables in our model can be derived through the *mean-field Variational Expectation Maximization (VEM)* [27] method, one of the common Bayesian learning algorithms. The learning process and the prediction are described in Section 4.1 to 4.4. Section 4.5 describes several implementation tricks to improve the time and space efficiency. Finally we analyze the complexity of both time and space in Section 4.6. Section 4.7 reports the pseudo code of the proposed model, which states our implementation of all the mathematical techniques introduced in this section.

4.1 Overview

According to Bayesian treatment, we need to maximize the likelihood of of the observed ratings, given the parameters $\theta = \{\Lambda_U, \Lambda_V, \sigma_R^2, \sigma_U^2, \sigma_V^2\}$ and averaging over all possible values of the hidden variables $Z = \{U, V, S_U, S_V\}$:

$$p(R|\theta) = \int_Z p(R, Z|\theta) dZ,$$

where $p(R, Z|\theta)$ is represented in (5). Unfortunately, the integration incurs the intractability of the optimization. Instead, we can apply a tractable auxiliary probability $q(Z|\theta')$ *to maximize the lower bound of the log likelihood*. The lower bound can be derived from Jensen's inequality as follows:

$$\begin{aligned}\log p(R|\theta) &= \log \int_Z q(Z|\theta') \frac{p(R,Z|\theta)}{q(Z|\theta')} dZ \qquad (6)\\ &= \log \mathbb{E}_{q(Z|\theta')}\left[\frac{p(R,Z|\theta)}{q(Z|\theta')}\right]\\ &\geq \mathbb{E}_{q(Z|\theta')}\left[\log \frac{p(R,Z|\theta)}{q(Z|\theta')}\right]\\ &= \mathbb{E}_{q(Z|\theta')}\left[\log \frac{p(Z|R,\theta)p(R|\theta)}{q(Z|\theta')}\right]\\ &= \mathbb{E}_{q(Z|\theta')}\Big[\log p(R|\theta)\Big] - \mathbb{E}_{q(Z|\theta')}\left[\log \frac{q(Z|\theta')}{p(Z|R,\theta)}\right]\\ &= \log p(R|\theta) - \mathbb{KL}\Big[q(Z|\theta') \| p(Z|R,\theta)\Big].\end{aligned}$$

$\mathbb{E}_{q(Z|\theta')}(X)$ is the expected value of X over probability distribution $q(Z|\theta')$, and $\mathbb{KL}(p\|q)$ is the non-negative Kullback-Leibler divergence of two distributions p and q. To tighten the lower bound, we should have the divergence be 0, which implies $q(Z|\theta') = p(Z|R,\theta)$. However, the intractability of $p(Z|R,\theta)$ demands a tractable replacement. An approach to variational tractable $q(Z|\theta')$ is assuming all the involved hidden variables being independent of each other:

$$\begin{aligned}q(Z|\theta') = &\prod_{i=1}^{N} \mathcal{N}(U_i|\lambda_{Ui}, \gamma_{Ui}) \prod_{j=1}^{M} \mathcal{N}(V_j|\lambda_{Vj}, \gamma_{Vj})\\ &\prod_{(i,f)\in E_U} \mathcal{W}(S_{Uif}|K+1, \Lambda_{Uif})^{\frac{b_U}{T_{Ui}}}\\ &\prod_{(j,g)\in E_V} \mathcal{W}(S_{Vjg}|K+1, \Lambda_{Vjg})^{\frac{b_V}{T_{Vj}}},\end{aligned}$$

where $\theta' = \{\lambda_{Ui}, \lambda_{Vj}, \gamma_{Ui}, \gamma_{Vj}, \Lambda_{Uif}, \Lambda_{Vjg}\}$. One advantage is that there is no shared parameter among the hidden variables in $q(Z|\theta')$. The independence assumption is called *mean-field approximation*.

Now we have two sets of parameters: θ from the model distribution, and θ' from the variational distribution. VEM learns patterns by optimizing both sets of parameters that determine the presence of all probability distributions. Like classical expectation maximization (EM) algorithm, we iteratively execute variational *E-step* and *M-step*, updating θ' and θ respectively until a local optimum or predefined maximum number of iterations is reached.

Due to page limits and the symmetric forms of probability distributions of users and items, the following mathematical details focus on the derivation of user parameters. Readers can take a similar derivation of item parameters. Last but not least, we leave additional mathematical derivations in [20].

4.2 Variational E-step

In variational E-step, we have to optimize variational parameters θ' as fixing model parameters θ. Mean-field approach leads to a general principle [28]. It incurs convenience to derive θ' in $q(Z|\theta')$. For instance, for some user i, we have his or her variational distribution $q(U_i)$ as follows:

$$\log q(U_i|\theta') = \mathbb{E}_{-q(U_i|\theta')}\Big[\log p(R, Z|\theta)\Big] + C_0,$$

where $-q(U_i|\theta')$ represents the joint distribution of all the random variables Z except U_i, and C_0 absorbs the terms not relevant to U_i. We extend the expected value over independent variational random variables based on the mean-field assumption.

$$\begin{aligned}
&\log q(U_i|\theta') \\
=&\frac{-1}{2\sigma_R^2}\sum_{j=1}^{M}\delta_{ij}\Big[-2R_{ij}U_i^T\mathbb{E}(V_j) + U_i^T\mathbb{E}(V_jV_j^T)U_i\Big] \\
&-\frac{1-b_U}{2}U_i^T(\sigma_U^2)^{-1}IU_i \\
&-\sum_{(i,f)\in E_U}\frac{b_U}{2T_{Ui}}\Big[\Big(U_i - \mathbb{E}(U_f)\Big)^T\mathbb{E}(S_{Uif})\Big(U_i - \mathbb{E}(U_f)\Big)\Big] \\
&-\sum_{(f,i)\in E_U}\frac{b_U}{2T_{Uf}}\Big[\Big(\mathbb{E}(U_f) - U_i\Big)^T\mathbb{E}(S_{Ufi})\Big(\mathbb{E}(U_f) - U_i\Big)\Big] \\
&+C_1, \qquad (7)
\end{aligned}$$

where C_1 absorbs the terms not relevant to U_i. The extension derivation refers to [29]. All the expected values as well as covariance matrices can be written as parameters in θ:

$$\begin{aligned}
\mathbb{E}(V_j) &= \lambda_{Vj} \\
\mathbb{E}(V_jV_j^T) &= \lambda_{Vj}\lambda_{Vj}^T + \gamma_{Vj} \\
\mathbb{E}(U_f) &= \lambda_{Uf} \\
\mathbb{E}(S_{Uif}) &= (K+1)\Lambda_{U_if}.
\end{aligned}$$

In the end, we complete the square to the right-hand side of (7) in order to write down a logarithmic form of multivariate normal distribution with mean λ_{Ui} and covariance matrix γ_{Ui}. They are what we need to update at E-Step:

$$\begin{aligned}
\gamma_{Ui} = &\Big[\frac{1}{\sigma_R^2}\sum_{j=1}^{M}\delta_{ij}\Big(\lambda_{Vj}\lambda_{Vj}^T + \gamma_{Vj}\Big) + (1-b_U)(\sigma_U^2)^{-1}I \\
&+ b_U\sum_{(i,f)\in E_U}\frac{1}{T_{Ui}}(K+1)\Lambda_{Uif} \\
&+ b_U\sum_{(f,i)\in E_U}\frac{1}{T_{Uf}}(K+1)\Lambda_{Ufi}\Big]^{-1}, \qquad (8)
\end{aligned}$$

$$\begin{aligned}
\lambda_{Ui} = \gamma_{Ui}&\Big[\frac{1}{\sigma_R^2}\sum_{j=1}^{M}\delta_{ij}R_{ij}\lambda_{Vj} \\
&+ b_U\sum_{(i,f)\in E_U}\frac{1}{T_{Ui}}(K+1)\Lambda_{Uif}\lambda_{Uf} \\
&+ b_U\sum_{(f,i)\in E_U}\frac{1}{T_{Uf}}(K+1)\Lambda_{Ufi}\lambda_{Uf}\Big]. \qquad (9)
\end{aligned}$$

We repeat similar derivations for item random variables, and thus obtain the update rules:

$$\begin{aligned}
\gamma_{Vj} = &\Big[\frac{1}{\sigma_R^2}\sum_{i=1}^{N}\delta_{ij}\Big(\lambda_{Ui}\lambda_{Ui}^T + \gamma_{Ui}\Big) + (1-b_V)(\sigma_V^2)^{-1}I \\
&+ b_V\sum_{(j,g)\in E_V}\frac{1}{T_{Vj}}(K+1)\Lambda_{Vjg} \\
&+ b_V\sum_{(g,j)\in E_V}\frac{1}{T_{Vg}}(K+1)\Lambda_{Vgj}\Big]^{-1}, \qquad (10)
\end{aligned}$$

$$\begin{aligned}
\lambda_{Vj} = \gamma_{Vj}&\Big[\frac{1}{\sigma_R^2}\sum_{i=1}^{N}\delta_{ij}R_{ij}\lambda_{Ui} \\
&+ b_V\sum_{(j,g)\in E_V}\frac{1}{T_{Vj}}(K+1)\Lambda_{Vjg}\lambda_{Vg} \\
&+ b_V\sum_{(g,j)\in E_V}\frac{1}{T_{Vg}}(K+1)\Lambda_{Vgj}\lambda_{Vg}\Big]. \qquad (11)
\end{aligned}$$

We treat the similarity variables S_U the same ways. Here we write down the final equation:

$$\Lambda_{Uif} = \Big[(\lambda_{Ui} - \lambda_{Uf})(\lambda_{Ui} - \lambda_{Uf})^T + \gamma_{Ui} + \gamma_{Uf} + \Lambda_U^{-1}\Big]^{-1}, \qquad (12)$$

$$\Lambda_{Vjg} = \Big[(\lambda_{Vj} - \lambda_{Vg})(\lambda_{Vj} - \lambda_{Vg})^T + \gamma_{Vj} + \gamma_{Vg} + \Lambda_V^{-1}\Big]^{-1}. \qquad (13)$$

4.3 Variational M-step

This step takes care of the optimization of model parameters θ with fixed variational parameters θ'. We can perform the task by taking the derivative of the lower bound in (6), and find the closed-form solution of each parameter in θ.

We use σ_U^2 as example. The lower bound in (6) is viewed as function of σ_U^2:

$$\begin{aligned} L(\sigma_U^2) &= \mathbb{E}_{q(Z|\theta')}\left[\log \frac{p(R,Z|\theta)}{q(Z|\theta')}\right] \\ &= -\frac{1}{2}\sum_{i=1}^{N}\left[\lambda_{Ui}^T(\sigma_U^2)^{-1}I\lambda_{Ui} + \text{tr}\Big((\sigma_U^2)^{-1}I\gamma_{Ui}\Big)\right. \\ &\quad \left. + \log|\sigma_U^2 I|\right] + C_2. \end{aligned}$$

The term C_2 is not relevant to σ_U^2. Let $\frac{\partial L}{\partial \sigma_U^2} = 0$ and then obtain the closed-form solution:

$$\sigma_U^2 = \frac{1}{KN}\sum_{i=1}^{N}\Big(\|\lambda_{Ui}\|_2^2 + \text{tr}(\gamma_{Ui})\Big). \tag{14}$$

Again, we apply similar derivation to the corresponding item variable:

$$\sigma_V^2 = \frac{1}{KM}\sum_{j=1}^{M}\Big(\|\lambda_{Vj}\|_2^2 + \text{tr}(\gamma_{Vj})\Big). \tag{15}$$

At M-step, we update σ_U^2 and σ_V^2 using the above assignment. Following similar derivations, we have the update rules of other model parameters:

$$\Lambda_U = \frac{1}{KN}\sum_{(i,f)\in E_U}\frac{1}{T_{Ui}}(K+1)\Lambda_{Uif}, \tag{16}$$

$$\Lambda_V = \frac{1}{KM}\sum_{(j,g)\in E_V}\frac{1}{T_{Vj}}(K+1)\Lambda_{Vjg} \tag{17}$$

and

$$\begin{aligned} \sigma_R^2 = \frac{1}{|R|}\sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{ij}\Big[R_{ij}^2 - 2R_{ij}\lambda_{Ui}^T\lambda_{Vj} + (\lambda_{Ui}^T\lambda_{Vj})^2 \\ + \text{tr}(\gamma_{Ui}\gamma_{Vj}) + \lambda_{Ui}^T\gamma_{Vj}\lambda_{Ui} + \lambda_{Vj}^T\gamma_{Ui}\lambda_{Vj}\Big], \end{aligned} \tag{18}$$

where $|R| = \sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{ij}$ is the number of ratings in the training data.

4.4 Prediction

After the convergence of the VEM, we are able to predict an unobserved rating R_{ij} using the learned parameters. Without knowing the true rating likelihood, we apply the same strategy as M-step: finding R_{ij} to maximize the lower bound $L(R_{ij})$ of likelihood. This strategy can be done by letting $\frac{\partial L}{\partial R_{ij}} = 0$. The closed-form solution is used to predict unobserved ratings:

$$\hat{R}_{ij} = \lambda_{Ui}^T\lambda_{Vj}. \tag{19}$$

4.5 Improving Efficiency

In this section we mention some practical tricks to improve the efficiency of SCP.

4.5.1 Time: Randomized Implicit Social Network

Generating implicit social connections is computationally expensive since we need to evaluate all possible pairwise connections. Previous works thus define heuristics to eliminate edges that are less likely to reflect strong social relations. For example, Ma et al. [16] leave merely those edges connecting two users who rate at least 10 identical items. However, such manually-crafted heuristics might not be suitable for every scenario, and it still costs $O(N^2)$ complexity to run the algorithm to determine which edge to be removed.

To address the efficiency issue, our idea is to do *random sampling* from a user's neighbors for each VEM iteration, as illustrated in Figure 8. To explain the trick, let us have a deeper analysis on the updating rules of our VEM model. In fact, there are three updating rules that require the enumeration of all possible pairs E_U. That is, a set of updated similarity scale matrices Λ_{Uif} for all neighbors f of user i is computed mainly for the updates of variational mean vector $\lambda_{Ui}, \lambda_{Vj}$ (shown in (9) (11)), variational covariance matrix γ_{Ui}, γ_{Vj} (shown in (8) (10)), and model scale matrix Λ_U, Λ_V (shown in (16) (17)). Here we focus on the pairwise update terms in (8),

$$\begin{aligned} b_U \sum_{(i,f)\in E_U}\frac{1}{T_{Ui}}(K+1)\Lambda_{Uif} + \sum_{(f,i)\in E_U}\frac{1}{T_{Uf}}(K+1)\Lambda_{Ufi} \\ = \frac{b_U}{N-1}\sum_{f=1,f\neq i}^{N}(K+1)\Big(\Lambda_{Uif} + \Lambda_{Ufi}\Big). \end{aligned} \tag{20}$$

The right-hand side of (20) reveals the mean of similarities between user i and all other users. Note that it is the mean value that we need. Thus, the law of large number tells us that for each VEM iteration, as long as we can sample sufficient amount of users $F \subseteq \{f : 1 \le f \le N, f \neq i\}$ uniformly at random, then the mean of similarities of the sampled neighbors will approach the original mean as follows:

$$\begin{aligned} &\frac{b_U}{N-1}\sum_{f=1,f\neq i}^{N}(K+1)\Big(\Lambda_{Uif} + \Lambda_{Ufi}\Big) \\ \approx &\frac{b_U}{|F|}\sum_{f\in F}(K+1)\Big(\Lambda_{Uif} + \Lambda_{Ufi}\Big). \end{aligned} \tag{21}$$

The same idea is applicable for the terms in (9) (10) (11) (16) (17) as well since they all involve the estimation of mean values. In summary, at E-step, we only need to sample a constant number of neighbors for each user, which allows us to avoid the curse of evaluating a fully-connected graph.

4.5.2 Space: Saving Matrix Parameter

SCP is composed of three kinds of $K \times K$ matrix parameters: variational covariance matrices $\gamma_{Ui}, \gamma_{Vj}\forall i,j$, variational scale matrices $\Lambda_{Uif}, \Lambda_{Vjg}\forall i,j,f,g$, and model scale matrices Λ_U, Λ_V. There are overall NK^2 values in $\gamma_{Ui}\forall i$ and $|E_U|K^2$ values in $\Lambda_{Uif}\forall i,f$ to be trained, which incurs certain burden on the optimization process. Furthermore, overfitting is also a concern since we have to train a number of parameters. A simple trick we have applied here is to make all the matrix parameters *diagonal*. In other words, all

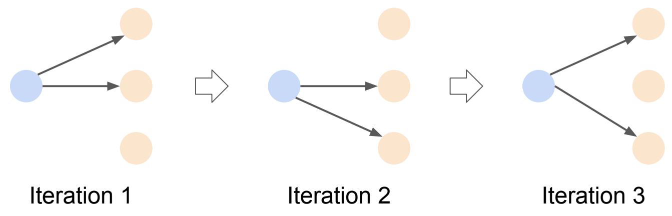


Fig. 8. Interpretation of randomized implicit social network. For each iteration (running E-step and M-step once), every user or item samples a small number of friends uniformly at random.

the off-diagonal elements are fixed to zero. Some previous works [25], [30] have also utilized such trick. When performing VEM, we need not to update off-diagonal elements at all. There are more details of this trick in Section 4.5.3.

In Section 4.5.1 we have described a trick to reduce the time complexity to linear using sampling techniques. It still requires $|E_U|K$ non-zero elements to store each of the variational scale matrices Λ_{Uif} and Λ_{Vjg}, and up to $O(|F|NK)$ storage for implicit social networks. Inspired by [26], we observe that for our task there is no need to store those variational scale matrices throughout the process. First, those variables are used only in the training stage. They play no role during the prediction stage (see Section 4.4). Second, in the VEM training stage, we observe that an updated Λ_{Uif} is always used in summation in (8), (9) and (16) later on. Hence, after Λ_{Uif} updates its value, we can use it in (8), (9), and (16) for the summation term right away, and then its space can be released since it is no longer needed. In the end, $O(|F|NK)$, the most dominant factor to space complexity, can be eliminated. In terms of space, our model becomes more scalable for big datasets after applying the tricks mentioned. We refer reader to Algorithm 3 to understand the implementation of this trick.

4.5.3 Diagonal Matrix Parameter

In Section 4.5.2, we propose one of the implementation tricks to save the space of matrix parameters: *making them diagonal.* That is, off-diagonal elements in these matrix parameters are always fixed 0 such that we do not have to compute or save them. For clear presentation of VEM derivations, we do not introduce the diagonal forms in Section 4.2 and 4.3. As the extension of Section 4.5.2, here we would like to explicitly write down the diagonal expressions of the parameters. In other words, *all the $K \times K$ matrix parameters $\{\gamma_{Ui}, \gamma_{Vj}, \Lambda_{Uif}, \Lambda_{Vjg}, \Lambda_U, \Lambda_V\}$ in the section change to K-dimensional vector parameters* $\{\vec{\gamma}_{Ui}, \vec{\gamma}_{Vj}, \vec{\Lambda}_{Uif}, \vec{\Lambda}_{Vjg}, \vec{\Lambda}_U, \vec{\Lambda}_V\}$. Elements in such vector parameters correspond to non-zero diagonal elements in original matrix parameters.

At first, we introduce the diagonal function diag$(\cdot)$. The input of the function can be either a column vector $x \in \mathbb{R}^K$ or a square matrix $X \in \mathbb{R}^{K\times K}$. The definition is shown below:

$$\operatorname{diag}(x) = \begin{bmatrix} x_1 & & & \\ & x_2 & & \\ & & \ddots & \\ & & & x_K \end{bmatrix}$$

$$\operatorname{diag}(X) = \left[x_{11}x_{22}\dots x_{KK}\right]^T.$$

For page limits, we discuss only user parameters in the following paragraphs. In variational E-Step, if we require γ_{Ui} to be diagonal, then (8) becomes:

$$\begin{aligned} \hat{\gamma}_{Ui} &= \text{Right-hand side of (8)} \\ \vec{\gamma}_{Ui} &= \operatorname{diag}(\hat{\gamma}_{Ui}). \end{aligned} \tag{22}$$

Note that $\hat{\gamma}_{Ui} \in \mathbb{R}^{K\times K}$ must be utilized in the update of (9). We cannot replace $\hat{\gamma}_{Ui}$ with diag$(\vec{\gamma}_{Ui})$ in (9); otherwise, the learning of λ_{Ui} would fail in practice.

Next, we derive the diagonal form of matrix Λ_{Uif}:

$$\begin{aligned} \vec{\Lambda}_{Uif} &= \left[\operatorname{diag}\Big((\lambda_{Ui}-\lambda_{Uf})(\lambda_{Ui}-\lambda_{Uf})^T + \gamma_{Ui} + \gamma_{Uf} + \Lambda_U^{-1}\Big)\right]^{-1} \\ &= \operatorname{inv}\Big((\lambda_{Ui}-\lambda_{Uf})\circ(\lambda_{Ui}-\lambda_{Uf}) + \vec{\gamma}_{Ui} + \vec{\gamma}_{Uf} + \vec{\Lambda}_U^{-1}\Big), \end{aligned} \tag{23}$$

where operator $\circ$ represents entrywise product and function inv$(\cdot)$ implies entrywise reciprocal. Another trick here is to perform diag$(\cdot)$ before inverse operation. The inverse of a diagonal matrix is to simply calculate the reciprocal of each diagonal element, which takes $O(K)$ time. Hence, the diagonal form can also prevent us from $O(K^3)$ general inverse calculation.

At last, Λ_U has its diagonal form as follows:

$$\begin{aligned} \vec{\Lambda}_U &= \operatorname{diag}\Bigg(\frac{1}{KN}\sum_{(i,f)\in E_U}\frac{1}{T_{Ui}}(K+1)\Lambda_{Uif}\Bigg) \\ &= \frac{1}{KN}\sum_{(i,f)\in E_U}\frac{1}{T_{Ui}}(K+1)\vec{\Lambda}_{Uif}. \end{aligned} \tag{24}$$

Therefore, when building SCP, as long as we respectively replace (8) (12) (16) with (22) (23) (24), then we are enabled to keep only the space of diagonal elements.

Due to the symmetric form, we list the diagonal matrix parameters for items without discussions:

$$\vec{\gamma}_{Vj} = \operatorname{diag}(\hat{\gamma}_{Vj}), \tag{25}$$

$$\vec{\Lambda}_{Vjg} = \operatorname{inv}\Big((\lambda_{Vj}-\lambda_{Vg})\circ(\lambda_{Vj}-\lambda_{Vg}) + \vec{\gamma}_{Vj} + \vec{\gamma}_{Vg} + \vec{\Lambda}_V^{-1}\Big), \tag{26}$$

$$\vec{\Lambda}_V = \frac{1}{KM}\sum_{(j,g)\in E_V}\frac{1}{T_{Vj}}(K+1)\vec{\Lambda}_{Vjg}. \tag{27}$$

4.6 Complexity Analysis

Section 4.5.2 has concluded that keeping only diagonal matrix parameters takes only $O(NK+MK)$ in terms of space. Other parameters do not demand space larger than such. Consequently, the overall space complexity is constrained to $O(|R|+|E|+NK+MK)$, where $|R|$ is the number of ratings in the training data, $|E|$ is the number of edges if an explicit social networks is provided, or zero for the case of implicit recommendation.

As for running time, since the number of iterations needed for convergence varies for different datasets, we analyze the time complexity of one VEM iteration. For variational variable sets $\lambda_U, \lambda_V, \gamma_U$ and γ_V, the update rules (8) and (9) take total time $O(|R|K^2 + NK^3 + MK^3)$ including $O(K^3)$ time to generate the inverse matrix. In practice, K is usually small as the size of the latent factor (e.g., 10 in our experiments). With the trick mentioned in Section 4.5.2, the similarity-related variables Λ_U, Λ_V can be computed in $O(|E|K)$ time for explicit social networks. However for implicit social networks, the complexity becomes $O(|F|NK + |F|MK)$ where $|F|$ is the fixed number of sampled users or items, which we have discussed in Section 4.5.1. To update σ_R^2, it takes time $O(|R|K)$. Note that in practice $|F|$ and K are far smaller than $|R|, |E|, N$ and M. Therefore, our model is scalable since the time complexity is linear to $|R|, |E|, N$ or M.

4.7 Pseudo Code of Social Covariance Prior

The learning algorithm of our SCP is composed of Algorithm 1, 2 and 3.

Algorithm 1 Function Learning

```
Input: Rating data $R$, explicit user social network $S^{(E)}$ if provided, convergence threshold $\epsilon$.
Output: Mean of user latent matrix $\lambda_U$, mean of item latent matrix $\lambda_V$.
1: Select partial rating data $R'$ from $R$ as validation set
2: Initialize parameters
3:                  ▷ $\lambda_U, \lambda_V$ must be initialized as random values
4: $e \leftarrow$ RMSE($R'$)
5: repeat
6:     MStep($R, S^{(E)}$)
7:     EStep($R$)
8:     $e' \leftarrow e$
9:     $e \leftarrow$ RMSE($R'$)
10: until $e' - e < \epsilon$
```

Algorithm 2 Function EStep

```
Input: Rating data $R$.
Output: Parameters $\{\gamma_{Ui}, \lambda_{Ui}\}$ for each user $i$, $\{\gamma_{Vj}, \lambda_{Vj}\}$ for each item $j$ and summation terms $\{\psi_{Ui}, \psi_{Vj}, \omega_{Ui}, \omega_{Vj}\}$ from MStep.
1: for each user $i$ do
2:     Update $\gamma_{Ui}$ by Equation (8) (or (22)) and $\psi_{Ui}$
3:     Update $\lambda_{Ui}$ by Equation (9), $\psi_{Ui}$ and $\omega_{Ui}$
4: end for
5: for each item $j$ do
6:     Update $\gamma_{Vj}$ by Equation (10) (or (25)) and $\psi_{Vj}$
7:     Update $\lambda_{Vj}$ by Equation (11), $\psi_{Vj}$ and $\omega_{Vj}$
8: end for
```

Algorithm 3 Function MStep

```
Input: Rating data $R$, explicit user social network $S^{(E)}$ if provided.
Output: Parameters $\{\Lambda_U, \Lambda_V, \sigma_R^2, \sigma_U^2, \sigma_V^2\}$ and summation terms $\{\psi_{Ui}, \psi_{Vj}, \omega_{Ui}, \omega_{Vj}\}$ in Equation (8) (9) (10) (11) (16) (17) as stated in Section 4.5.2.    ▷ $\Lambda_{Uif}, \Lambda_{Vjg}$ should be updated in E-step, but we put them here to handle all the social-network-related parameters inside a function
1: Update $\sigma_R^2$ by Equation (18)
2: Update $\sigma_U^2$ by Equation (14)
3: Update $\sigma_V^2$ by Equation (15)
4: $\psi_{Ui} \leftarrow \omega_{Ui} \leftarrow 0$ for each user $i$
5: $\psi_{Vj} \leftarrow \omega_{Vj} \leftarrow 0$ for each item $j$
6: $\phi_U \leftarrow \phi_V \leftarrow 0$
7: for each user $i$ do
8:     if $S^{(E)}$ provided then
9:         $F \leftarrow$ neighbors of user $i$ in $S^{(E)}$
10:    else
11:        $F \leftarrow$ uniformly sampled neighbors from other $(N-1)$ users as described in Section 4.5.1
12:    end if
13:    for each neighbor $f \in F$ do
14:        Obtain $\Lambda_{Uif}$ by Equation (12) (or (23))
15:        $\psi_{Ui} \leftarrow \phi_{Ui} + \frac{1}{T_{Ui}}(K+1)\Lambda_{Uif}$
16:        $\psi_{Uf} \leftarrow \phi_{Uf} + \frac{1}{T_{Uf}}(K+1)\Lambda_{Uif}$ if $T_{Uf} > 0$
17:        $\omega_{Ui} \leftarrow \omega_{Ui} + \frac{1}{T_{Ui}}(K+1)\Lambda_{Uif}\lambda_{Uf}$
18:        $\omega_{Uf} \leftarrow \omega_{Uf} + \frac{1}{T_{Uf}}(K+1)\Lambda_{Uif}\lambda_{Ui}$ if $T_{Uf} > 0$
19:        $\phi_U \leftarrow \phi_U + \frac{1}{T_{Ui}}(K+1)\Lambda_{Uif}$
20:                                  ▷ Release the space of $\Lambda_{Uif}$ here
21:    end for
22: end for
23: Update $\Lambda_U$ by Equation (16) (or (24)) and $\phi_U$
24: for each item $j$ do
25:    $G \leftarrow$ uniformly sampled neighbors from other $(M-1)$ items as described in Section 4.5.1
26:    for each neighbor $g \in G$ do
27:        Obtain $\Lambda_{Vjg}$ by Equation (13) (or (26))
28:        $\psi_{Vj} \leftarrow \phi_{Vj} + \frac{1}{T_{Vj}}(K+1)\Lambda_{Vjg}$
29:        $\psi_{Vg} \leftarrow \phi_{Vg} + \frac{1}{T_{Vg}}(K+1)\Lambda_{Vjg}$ if $T_{Vg} > 0$
30:        $\omega_{Vj} \leftarrow \omega_{Vj} + \frac{1}{T_{Vj}}(K+1)\Lambda_{Vjg}\lambda_{Vg}$
31:        $\omega_{Vg} \leftarrow \omega_{Vg} + \frac{1}{T_{Vg}}(K+1)\Lambda_{Vjg}\lambda_{Vj}$ if $T_{Vg} > 0$
32:        $\phi_V \leftarrow \phi_V + \frac{1}{T_{Vj}}(K+1)\Lambda_{Vjg}$
33:                                  ▷ Release the space of $\Lambda_{Vjg}$ here
34:    end for
35: end for
36: Update $\Lambda_V$ by Equation (17) (or (27)) and $\phi_V$
```

5 EXPERIMENTS

5.1 Experiment Settings

5.1.1 Datasets

To evaluate the proposed approach, we conduct experiments using four datasets (Epinion [1], Ciao [2], Flixster [3] and

1. www.public.asu.edu/˜jtang20/datasetcode/truststudy.htm
2. www.public.asu.edu/˜jtang20/datasetcode/truststudy.htm
3. www.cs.ubc.ca/˜jamalim/datasets/

TABLE 1
Statistics of datasets in our experiments; "N/A" means no explicit user social network.

Name	Number of ratings	Number of users	Number of items	Number of edges	Social Relation	Reference
Epinion	912441	22164	296277	354897	Trust	[31]
Ciao	282650	7375	105114	57544	Trust	[31]
Flixster	8196077	147612	48794	2538746	Friendship	[32]
FilmTrust	35494	1508	2071	1632	Trust	[33]
MovieLens 1M	1000209	6040	3706	N/A	N/A	[34]

FilmTrust [4]) containing explicit user binary social networks and one dataset (MovieLens 1M [5]) as the representative of a large rating dataset without explicit social network. For each dataset, we discard the repeated ratings such that if a user rated an item more than once, then we keep only the latest rating. The statistics of the datasets are shown in Table 1. For each rating in each dataset, we subtract it by a constant mean of all ratings in that dataset.

5.1.2 Competitors

Next, we introduce the following baseline and the state-of-the-art social recommendation models for comparison.

- *Probabilistic Matrix Factorization (PMF)* [22]. As have described in Section 3.2, PMF is a popular model-based collaborative filtering approach without using social relationship. It serves as the baseline in our experiments.
- *Explicit Social Recommendation Models*. We implemented several well-known explicit social recommender systems. The corresponding objective function for each is shown below:
  - *Social Regularization (SocReg)* [9]

$$\arg\min_{U,V} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{ij}\left(R_{ij}-U_i^T V_j\right)^2 + \frac{\alpha_U}{2}\|U\|_2^2 + \frac{\alpha_V}{2}\|V\|_2^2 + \frac{\beta}{2}\sum_{(i,f)\in E_U}\omega_{if}\|U_i-U_f\|_2^2, \quad (28)$$

    where ω_{if} denotes the edge strength between user i and f.
  - *SocialMF* [6]

$$\arg\min_{U,V} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{ij}\left(R_{ij}-U_i^T V_j\right)^2 + \frac{\alpha_U}{2}\|U\|_2^2 + \frac{\alpha_V}{2}\|V\|_2^2 + \frac{\beta}{2}\sum_{i=1}^{N}\|U_i-\sum_{f\in F_i}\omega_{if}U_f\|_2^2, \quad (29)$$

    where F_i denotes the set of neighbors of user i.
  - *TrustSVD* [4]

$$\arg\min_{U,V,Y,W,b} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{ij}\left(R_{ij}-\hat{R}_{ij}(U,V,Y,W,b)\right)^2 + \frac{\beta}{2}\sum_{(i,f)\in E_U}(\omega_{if}-\hat{\omega}_{if}(U,W))^2 + \frac{\alpha}{2}C_0(U,V,Y,W,b) + \frac{\beta}{2}C_1(U), \quad (30)$$

    where Y represents latent factors of items that a user rates in the training data, W denotes latent factors of users that a user trusts, b is the bias of users and items. Functions $\hat{R}, \hat{\omega}$ are designed to predict unobserved entries in R, ω respectively, and functions C_0, C_1 contain regularization terms.

  Both SocReg and SocialMF are well-known explicit social recommendation works, while TrustSVD is the state-of-the-art solution in this domain which outperforms most of the existing algorithms.
- *Implicit social recommendation*. For each of the five datasets, we follow the proposal in [16] to generate implicit user social networks. That is, the existence of an edge is confirmed if two users commonly rate at least 10 items and the Pearson correlation coefficient between their ratings is larger than 0.5. Then the generated networks are fed into the above models (i.e. SocReg, SocialMF and TrustSVD) as the input social network to generate the recommendation results. Note that although [16] proposes the concept of dissimilar edges, it is not applied in our experiments since it is specifically designed for SocReg. Recall that SCP learns an implicit user social network optimization itself, therefore it does not use the generated implicit social network.

All the competitor approaches are trained using stochastic gradient descent (SGD), as mentioned in the corresponding paper. The learning rate is dynamically adapted by ADADELTA [35] where the insensitive parameters is set to $\rho = 0.95, \epsilon = 10^{-6}$, according to the original paper. We tune α, β by searching values in $\{10^p | p \in \mathbb{Z}, -4 \le p \le 1\}$, and let $\alpha_U = \alpha_V$. We always fix the number of latent factors $K = 10$. Finally, the number of sampled neighbors $|F| = 100$ for SCP.

5.1.3 Evaluation

We report model performance using a common evaluation metric: *Root Mean Squared Error (RMSE)*. It is widely used

4. www.librec.net/datasets.html
5. grouplens.org/datasets/movielens/

TABLE 2
RMSE and running time (seconds) of PMF using SGD and VEM methods on Epinion (EP), Ciao (CI), Flixster (FL), FilmTrust (FT) and MovieLens 1M (ML); (*) denotes the RMSE being significantly lower than the other under 95% confidence level.

		EP	CI	FL	FT	ML
SGD	RMSE	1.1506	1.0740	0.8744	0.8187	0.8797
	Time	11.53	3.05	102.32	0.34	14.75
VEM	RMSE	1.1078*	1.0610*	0.8474*	0.8063	0.8582*
	Time	12.67	1.18	184.87	0.43	7.80

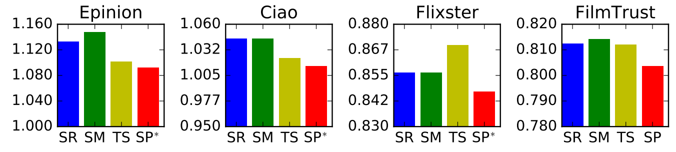


Fig. 9. RMSE of recommendation models using explicit user social networks. Labels are SocReg (SR), SocialMF (SM), TrustSVD (TS) and SCP (SP); (*) denotes the best model significantly better than the second best one under 95% confidence level.

for evaluating rating prediction in a recommender system.

$$\text{RMSE} = \sqrt{\frac{1}{|R|}\sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{ij}(\hat{R}_{ij} - R_{ij})^2}$$

where R is the set of test rating records as a sparse matrix, $\delta_{ij} \in \{0, 1\}$ represents whether the labelled rating R_{ij} exists in the test data, and $\hat{R}_{ij}$ is the predicted rating from a recommendation model. Smaller RMSE implies better prediction performance.

We use 5-fold cross validation, and report average results of the 5 folds. Moreover, we choose 10% of the training data as the validation data for each of the model. After each epoch of the VEM or SGD optimization, we evaluate RMSE on the validation data. If we observe RMSE decreasing by less than 10^{-4}, the model stops training as convergence is considered to be reached.

5.2 Experiment Results

Here we would like to evaluate five hypotheses (H1 to H5) to justify the effectiveness of our model. Furthermore, we conduct sensitivity analysis of parameters b_U, b_V in SCP to provide some suggestions for parameter selection.

5.2.1 Performance Comparisons

H1: VEM-based learning model produces competitive results in learning PMF comparing to SGD-based learning methods. One key idea of our model is to resort to VEM instead of SGD in optimization, since VEM allows us to embed a complex social prior and learn both latent user/item factors as well as implicit social co-variance together. Thus we think it is meaningful to compare whether VEM learns as good and fast as SGD in a traditional PMF setting to confirm VEM being a competitive alternative. Table 2 lists RMSE on PMF using both techniques. The results show that the VEM-based learning strategy is at least as good as SGD in terms of accuracy and efficiency, which strengthen our motivation of replacing SGD with VEM.

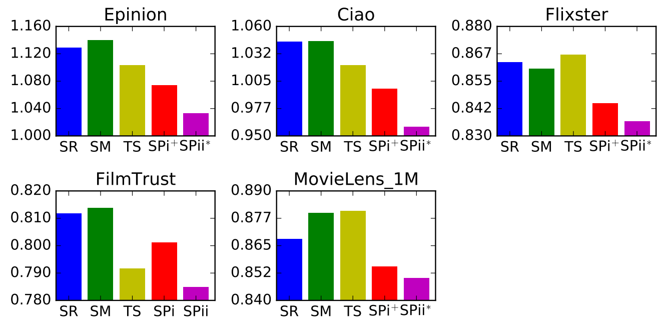


Fig. 10. RMSE of recommendation models under implicit user social networks. Labels are SocReg (SR), SocialMF (SM), TrustSVD (TS), SCP given implicit network (SPi) and SCP using both implicit user and item social networks (SPii); (*) denotes the best RMSE significantly better than the second best one under 95% confidence level; ($^+$) denotes the second best RMSE significantly better than the third best one under 95% confidence level.

H2: Given an explicit social network, our model can outperform existing explicit social recommender model. Since we claim our model as a general solution for both explicit and implicit social recommendations, we would like to first evaluate its effectiveness on explicit social recommendation. Figure 9 demonstrates the performance differences of various social recommender systems. For three of the four datasets with explicit user social networks, SCP reaches the lowest RMSE and is significantly better than the second best model. As for FilmTrust, SCP still keeps competitive with all the other models. The results confirm that SCP can be regarded as an effective solution to the classical explicit social recommendation problem.

H3: Without a given social network, our model can still outperform the other implicit social recommender solutions. Now we assume that the explicit user social networks are not available for training a model. Here we can evaluate the models with one additional dataset, the MovieLens dataset, since no explicit social network is needed in training. For competitor models, we generate implicit user social networks as presented in 5.1.2. Note that SCP does not need the generated network since the approach assumes a fully-connected implicit social network and learns the edge strength automatically. Figure 10 shows the performance in terms of RMSE. The results show that for 4 out of the 5 datasets, our model outperforms the competitors significantly. The only exception is the FilmTrust dataset where there is no statistically significant difference between the top 2 models TrustSVD and SCP. Also it is interesting to learn that using implicit networks, each of the two superior models (i.e. SCP and TrustSVD) outperforms its explicit-network counterpart, while on the contrary SocReg and SocialMF does a better job with explicit networks. It reflects the finding from previous studies that the social ties of users may not be fully transferable to the similarity of tastes on items. Our model shows that directly learning the implicit connection among people yields a more effective recommender system.

H4: Learning implicit item networks can further boost our model's performance. In Section 3.3.3, we propose extended SCP to consider implicit item social networks. Here

TABLE 3
Mean average precision of each user's top-k similar implicit neighbors of the Pearson correlation coefficient-defined network (PCC) and the SCP-learned network (SCP) on datasets FilmTrust (FT) and Ciao (CI).

	k	10	20	50	100	200
FT	PCC	1.200%	1.313%	1.393%	1.403%	1.403%
	SCP	**1.381%**	**1.604%**	**1.960%**	**2.146%**	**2.275%**
CI	PCC	**0.216%**	0.218%	0.221%	0.223%	0.223%
	SCP	0.189%	**0.230%**	**0.296%**	**0.373%**	**0.473%**

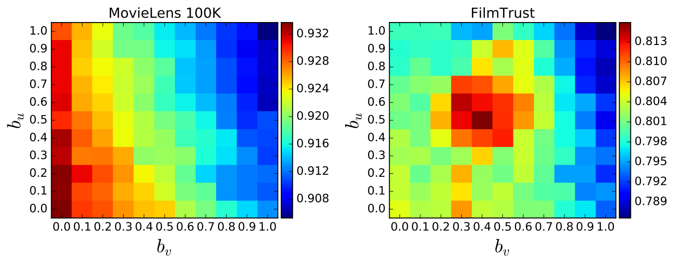


Fig. 11. RMSE in terms of different parameter combinations for datasets MovieLens 100K and FilmTrust.

we want to verify whether learning both the user and the item networks together can indeed boost the performance. The results are presented also in Figure 10 (SPii). We have observed a significant jump of performance after adding the item network. It does reveal an interesting finding that modeling implicit connection among items is indeed a very promising direction.

H5: SCP can discover hidden user social connection better than the human-defined implicit user social network. Although learning hidden social connections between people is not the goal of this paper, it is not hard to realize that using Equation 3 we can generate the probability that the latent factors of two users are similar as a heuristic to infer social connections. Here we want to verify whether the discovered network matches the true network better than the one identified by competitors. To verify our hypothesis, we utilize Mean Average Precision (MAP) to compare two uncovered implicit user social networks: one comes from SCP learning and another comes from the competitor's solution using Pearson correlation coefficient as described in Section 5.1.2. We report the comparison using two smallest datasets, FilmTrust and Ciao, with explicit user social networks as ground truth. Table 3 lists the MAP comparisons of k most similar neighbors of each user in both user implicit social networks. Results reveal that in almost all cases, our model outperforms the competitor in revealing the connections between users.

5.2.2 Parameter Sensitivity

As have been claimed in Section 3 to 4, our model automatically learns almost all the hidden variables from data. Nevertheless, to balance the contribution between Gaussian spherical prior and social prior, there remains two parameters b_U, b_V that still need to be determined through cross validation. Here we show how these two parameters influence the overall performance to suggest reasonable assignments for them.

From 0 to 1, we experiment different combinations of b_U, b_V settings, and then draw the RMSE distributions in Figure 11. Due to the page limit, we present the distributions for only implicit social recommendation with the incorporation of user and item social networks on MovieLens 100K, sharing the same reference as MovieLens 1M, and FilmTrust. The resulting distribution shown in the left figure (i.e., MovieLens 100K) is different from that in the right (i.e., FilmTrust), implying distinct spectrum of social influences between the two datasets. For example, for MovieLens 100K, the smooth distribution simply indicates that the overall performance does improve with more social influence factors considered, where the worse performance lies in the plane where only smaller social influence exists. For FilmTrust, it seems that the worst performance lies in the middle where these two factors are close to 0.5. We also discover that in general setting b_U close to 1 yields good results. In fact $b_U = b_V = 1$ generates the best results for all the datasets we have tested. Such finding supports the idea of eliminating Gaussian spherical priors because of the strong regularization capability of fully-connected implicit social networks. Without Gaussian priors, our model gains more freedom to train background social relations given rating information. That is, if the users do not have time to tune the parameters using validation dataset, we suggest the above values as default.

6 Conclusion

Probabilistic matrix factorization has been a successful machine learning model toward recommender systems. Based on the social correlation intuition, we propose a new approach to incorporate social network information into probabilistic matrix factorization. We list our contributions again:

- In terms of effectiveness, we successfully build a joint model simultaneously to learn factorized matrices and social network structures. Experiments support that out new approach outperforms previous works that either focus on explicit social recommendation or implement implicit social recommendation in two separate stages.
- Distinct from learning a shared social strength, our work allows learning an individual social strength for each latent factor. We believe that the multi-dimensional social strength learning can benefit the overall recommendation quality.
- In terms of efficiency, to address the scalability problem resulting from fully connected implicit social networks, we propose several practical tricks during the learning process so the complexity can be reduced from quadratic to linear.

Future works consist of two parts. First, we would like to investigate how the content information can be incorporated into the existing model for a hybrid recommender system. Second, we will also consider bringing into our model the temporal information from both social and ratings sides to further boost the performance.

REFERENCES

[1] A. J. Chaney, D. M. Blei, and T. Eliassi-Rad, "A probabilistic model for using social networks in personalized item recommendation," in *Proc. of ACM RecSys*, 2015, pp. 43–50.

[2] H. Fang, Y. Bao, and J. Zhang, "Leveraging decomposed trust in probabilistic matrix factorization for effective recommendation," 2014.

[3] J. A. Golbeck, "Computing and applying trust in web-based social networks," Ph.D. dissertation, 2005.

[4] G. Guo, J. Zhang, and N. Yorke-Smith, "Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings," 2015.

[5] M. Jamali and M. Ester, "Trustwalker: A random walk model for combining trust-based and item-based recommendation," in *Proc. of ACM SIGKDD*, 2009, pp. 397–406.

[6] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proc. of ACM RecSys*, 2010, pp. 135–142.

[7] H. Ma, H. Yang, M. R. Lyu, and I. King, "Sorec: Social recommendation using probabilistic matrix factorization," in *Proc. ACM CIKM*, 2008, pp. 931–940.

[8] H. Ma, I. King, and M. R. Lyu, "Learning to recommend with social trust ensemble," in *Proc. ACM SIGIR*, 2009, pp. 203–210.

[9] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proc. ACM WSDM*, 2011, pp. 287–296.

[10] P. Massa and P. Avesani, "Trust-aware recommender systems," in *Proc. of ACM RecSys*, 2007, pp. 17–24.

[11] X. Xin, I. King, H. Deng, and M. R. Lyu, "A social recommendation framework based on multi-scale continuous conditional random fields," in *Proc, ACM CIKM*, 2009, pp. 1247–1256.

[12] B. Yang, Y. Lei, D. Liu, and J. Liu, "Social collaborative filtering by trust," in *Proc. IJCAI*, 2013, pp. 2747–2753.

[13] S. Fazeli, B. Loni, A. Bellogin, H. Drachsler, and P. Sloep, "Implicit vs. explicit trust in social matrix factorization," in *Proc. ACM RecSys*, 2014, pp. 317–320.

[14] G. Guo, J. Zhang, D. Thalmann, A. Basu, and N. Yorke-Smith, "From ratings to trust: An empirical study of implicit trust in recommender systems," in *Proc. ACM SAC*, 2014, pp. 248–253.

[15] C. Lin, R. Xie, X. Guan, L. Li, and T. Li, "Personalized news recommendation via implicit social experts," *Information Sciences*, vol. 254, pp. 1 – 18, 2014.

[16] H. Ma, "An experimental study on implicit social recommendation," in *Proc. of ACM SIGIR*, 2013, pp. 73–82.

[17] H. Ma, I. King, and M. R. Lyu, "Learning to recommend with explicit and implicit social relations," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 29:1–29:19, May 2011.

[18] H. Ma, "On measuring social friend interest similarities in recommender systems," in *Proc. of ACM SIGIR*, 2014, pp. 465–474.

[19] C.-m. Au Yeung and T. Iwata, "Strength of social influence in trust networks in product review sites," in *Proc. of ACM WSDM*, 2011, pp. 495–504.

[20] "Supplemental material website," https://www.dropbox.com/sh/wj736mqbutwxdy4/AACDm4Fnu-ZAVK8SwiJmEPgIa/.

[21] G. Guo, J. Zhang, D. Thalmann, and N. Yorke-Smith, "Etaf: An extended trust antecedents framework for trust prediction," in *Proc. of IEEE/ACM ASONAM*, 2014, pp. 540–547.

[22] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. of NIPS*, 2008, pp. 1257–1264.

[23] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[24] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proc. of ICML*, 2008, pp. 880–887.

[25] H. Shan and A. Banerjee, "Generalized probabilistic matrix factorizations for collaborative filtering," in *Proc. of IEEE ICDM*, 2010, pp. 1025–1030.

[26] Y. J. Lim and Y. W. Teh, "Variational bayesian approach to movie rating prediction," in *Proc. of KDD Cup and Workshop*, 2007.

[27] J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith, M. W. (eds, M. J. Beal, and Z. Ghahramani, "The variational bayesian em algorithm for incomplete data: with application to scoring graphical model structures," 2003.

[28] E. P. Xing, M. I. Jordan, and S. Russell, "A generalized mean field algorithm for variational inference in exponential families," in *Proc. of UAI*, 2003, pp. 583–591.

[29] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," nov 2012.

[30] H. Jing, A.-C. Liang, S.-D. Lin, and Y. Tsao, "A transfer probabilistic collective factorization model to handle sparse data in collaborative filtering," in *Proc. of IEEE ICDM*, 2014, pp. 250–259.

[31] J. Tang, H. Gao, H. Liu, and A. D. Sarma, "eTrust: Understanding trust evolution in an online world," in *Proc. of ACM SIGKDD*, 2012.

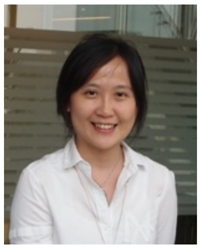
[32] M. Jamali, "Flixster dataset," http://www.cs.ubc.ca/~jamalim/datasets/.

[33] G. Guo, J. Zhang, and N. Yorke-Smith, "A novel bayesian similarity measure for recommender systems," in *Proc. of IJCAI*, 2013, pp. 2619–2625.

[34] GroupLens, "Movielens datasets," http://grouplens.org/datasets/movielens/.

[35] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *ArXiv e-prints*, Dec. 2012.

Chin-Chi Hsu received the master degree from the Department of Computer Science and Information Engineering, National Taiwan University in 2014. He is a research assistant in the Institute of Information Science, Academia Sinica, Taiwan. His research domains include machine learning, data mining and social network analysis. In particular, he researched time series mining, social recommendation and node importance ranking in networks.



Mi-Yen Yeh is currently Associate Research Fellow of Institute of Information Science (and Research Center for IT Innovation under joint appointment) at Academia Sinica, Taiwan. She received her B.S. and Ph.D. degrees in Electrical Engineering from National Taiwan University, Taiwan, in 2002 and 2009, respectively. Her main research area is on data mining and databases, with a specific focus on mining ordered data, social network analysis, and data management on non-volatile memory. She received the best paper award (in system software and security) of the 28th annual ACM Symposium on Applied Computing (SAC 2013), Distinguished Postdoctoral Fellowship in Academia Sinica, and Research Exploration Award in Pan Wen Yuan Foundation.

Shou-De Lin is currently a full professor in the CSIE department of National Taiwan University. He holds a BS in EE department from National Taiwan University, an MS-EE from the University of Michigan, and an MS in Computational Linguistics and PhD in Computer Science both from the University of Southern California. He leads the Machine Discovery and Social Network Mining Lab in NTU. Before joining NTU. Prof. Lin's research includes the areas of machine learning and data mining, social network analysis, and natural language processing. His international recognition includes the best paper award in IEEE Web Intelligent conference 2003, Google Research Award in 2007, Microsoft research award in 2008, merit paper award in TAAI 2010, best paper award in ASONAM 2011, US Aerospace AFOSR/AOARD research award winner for 5 years. He is the all-time winners in ACM KDD Cup, leading or co-leading the NTU team to win 5 championships. He also leads an NTU research team to win WSDM Cup 2016. He has served as the senior PC for SIGKDD and area chair for ACL. He is currently the associate editor for International Journal on Social Network Mining, Journal of Information Science and Engineering, and International Journal of Computational Linguistics and Chinese Language Processing.