Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Enhancing recommender systems by incorporating social information*

Li-wei HUANG†[1,2], Gui-sheng CHEN[2], Yu-chao LIU[2], De-yi LI†[2]

([1]*Institute of Command Information System, PLA University of Science and Technology, Nanjing 210007, China*)

([2]*Institute of Electronic System Engineering, Beijing 100039, China*)

†E-mail: huangliwei.1985@gmail.com; lidy@cae.cn

Received Mar. 15, 2013; Revision accepted June 26, 2013; Crosschecked Aug. 7, 2013

Abstract: Although recommendation techniques have achieved distinct developments over the decades, the data sparseness problem of the involved user-item matrix still seriously influences the recommendation quality. Most of the existing techniques for recommender systems cannot easily deal with users who have very few ratings. How to combine the increasing amount of different types of social information such as user generated content and social relationships to enhance the prediction precision of the recommender systems remains a huge challenge. In this paper, based on a factor graph model, we formalize the problem in a semi-supervised probabilistic model, which can incorporate different user information, user relationships, and user-item ratings for learning to predict the unknown ratings. We evaluate the method in two different genres of datasets, Douban and Last.fm. Experiments indicate that our method outperforms several state-of-the-art recommendation algorithms. Furthermore, a distributed learning algorithm is developed to scale up the approach to real large datasets.

Key words: Recommender system, Social information, Factor graph model
doi:10.1631/jzus.CIIP1303 **Document code:** A **CLC number:** TP301.6

1 Introduction

In the big data age, there has been an exponential growth of information generated on the World Wide Web. Recommender systems, which aim to alleviate information overload for social media users by presenting the most attractive and relevant content, have become more and more important. Although extensive research for recommender systems has been conducted by a broad range of communities including social and computer scientists, physicists, and interdisciplinary researchers, most of these techniques suffer from several inherent weaknesses. One well-known challenge is the data sparseness problem. As reported in Sarwar *et al*. (2001), the density of the available ratings in commercial recommender systems is often less than 1%. But now, the growing and readily available social information such as user generated content and various relationships between users, is raising new opportunities to mine users' interests accurately and mitigate the data sparseness problem considerably.

In the real world, users have their own preferences, which are determined by factors such as experience, background, knowledge level, and beliefs. In the meantime, users often ask their friends or trusted people for recommendations. It was found that given a choice between recommendations from friends and recommender systems, in terms of quality and usefulness, friends' recommendations are preferred even though the recommendations given by the recommender systems have a high novelty factor (Sinha and Swearingen, 2001). Hence, the social relationships should be incorporated into recommender systems. Correspondingly, we make an assumption that users'

* Project supported by the National Natural Science Foundation of China (Nos. 61035004, 61273213, 61072043, and 61305055) and the National Defense Science Foundation of China (No. 9140A15090112JB93180)

final decisions are determined by their preferences and the influences from other users. Specifically, we take the user-item ratings as the users' final decisions. Users' own preferences can be extracted from their profiles, various user generated content, etc. For the influences from other users, we discriminate effects from different relationship information and can consider them together, such as friend relationship in Facebook, follower-followee relationship in Twitter, and trust relationship in Epinions.

The problem is how to design a flexible model for effectively and efficiently combining diverse social information to improve the prediction precision of recommender systems. This problem is nontrivial and poses several important challenges. The first is how to define different factors to describe the respective effects of different information on users' ratings, and how to combine the factors to build a high-quality predictive model. Second, only a small part of items are rated by users, and most of ratings are missing in the user-item matrix. Third, there are thousands, even millions, of users and items in real recommender systems. It is necessary to develop an approach that can scale up well to the real systems.

In this paper, we propose a semi-supervised probabilistic model to address the above challenges. Specifically, we formalize the problem based on the factor graph theory (Kschischang *et al*., 2001). By defining each type of information as a factor, the missing ratings can be predicted. We conducted experiments on two datasets, Douban and Last.fm. Experimental results showed that the proposed method achieves better performance than several state-of-the-art recommendation algorithms. The fact that one can combine different social information for recommendation shows that our proposed model is quite flexible and general.

2 Related work

In this section, we give an overview of two major approaches in recommender systems: (1) traditional recommender techniques, including content-based recommendation and collaborative filtering; (2) social recommender systems, which have drawn a lot of attention recently.

2.1 Traditional recommender systems

Traditional recommender systems are generally based on two strategies, i.e., content-based recommendation (Balabanovic and Shoham, 1997; Lops *et al*., 2011) and collaborative filtering (Breese *et al*., 1998; Koren and Bell, 2011).

In content-based recommendation systems, one tries to recommend items similar to those a given user has liked in history. A profile is created for each user or item to describe its inherent characteristics. For example, a movie profile may include attributes regarding its genre, its director, the participating actors, its box office popularity, etc. User profiles may include their selected items, ratings, demographic information, etc. The profiles help recommender systems associate users with matching items. Of course, content-based strategies require collecting a lot of information to construct user profiles that might be unavailable or difficult to gather. Moreover, content-based recommendation has drawbacks such as the serendipity problem and new user problem.

Collaborative filtering (CF) systems take the known preferences of a group of users to make recommendations or predictions of the unknown preferences for other users. A CF system relies only on users' history behavior, such as previous transactions and product ratings, without constructing explicit user profiles. The correlations between users and interdependencies among products are analyzed to predict new user-item associations. The two primary areas of CF are neighborhood based models and latent factor models:

Neighborhood based models focus mainly on finding the similar users (Breese *et al*., 1998; Jahrer *et al*., 2010) or items (Sarwar *et al*., 2001; Deshpande and Karypis, 2004) for recommendations. User-based approaches predict the ratings of active users based on the ratings found for similar users, while item-based approaches predict the ratings of active users based on the computed information of items similar to those chosen by the active user.

Latent factor models have become an important branch among CF approaches, due to their superiority in terms of accuracy and scalability, as shown in Netflix competition (Koren *et al*., 2009). Latent factor models attempt to explain the ratings by characterizing both items and users for some factors inferred from the rating patterns. Non-negative matrix

factorization (NMF) is one of the famous representatives (Lee and Seung, 1999). Matrix factorization (MF) inspired a group of most well-known latent factor models, e.g., singular value decomposition (SVD) (Kurucz *et al*., 2007) and SVD++ (Koren, 2008). Probabilistic matrix factorization (PMF) was proposed to carry out the rating factorization from a probabilistic view (Salakhutdinov and Mnih, 2008), which leads to the most widely used regularized L_2-norm regression model. Logistic regression was also proposed to learn latent factors (Agarwal and Chen, 2009). The latest developments in recommender systems have explored different criteria to improve user satisfaction based on latent factor models, such as modeling the user choice process (Yang *et al*., 2011) and the marginal net utility (Wang and Zhang, 2011). Because a CF system considers only users' history behavior, it often suffers from data sparseness and cold start problems.

2.2 Social recommender systems

Currently, there is extensive research on how to use social media information to improve recommender systems. In Lu *et al*. (2010), in order to predict review quality, social context information including authors' identities and social networks was incorporated into a unified framework by adding regularization constraints to the text-based predictors. Their experiment results demonstrated that the prediction accuracy of review can be improved by incorporating social contextual information, especially when just sparse training data is available. By combining topic modeling and social network analysis, Mei *et al*. (2008) proposed a method that can leverage the power of both statistical topic models and discrete regularization, and that can be applied to many text mining tasks such as author-topic analysis, spatial text mining, and community discovery. Ma *et al*. (2008; 2011a) incorporated social friendship into their models. They interpreted known ratings as the representation of the users' tastes; when only known ratings and social friendship are considered, known ratings should be the results of the user's tastes and the influence of friends. In their following work (Ma *et al*., 2011b), in order to effectively fuse friends information in matrix factorization, they defined two social regularization terms to constrain the matrix factorization objective function. Shen and Jin (2012) developed a joint personal and social latent factor (PSLF) model for social recommendation. Yang *et al*. (2012a) studied top-k recommendation using a social network. Yang *et al*. (2012b) inferred category-specific social trust circles from available ratings data combined with social network data, and made an effort to develop a circle-based recommendation system. Zhou *et al*. (2012) proposed a kernelized probabilistic matrix factorization (KPMF) model, which can effectively incorporate external side information into matrix factorization. Xin *et al*. (2009) proposed a multiscale continuous conditional random fields (MCCRF) model as a framework for social recommendations. However, the MCCRF model essentially does not use any social information but explores only similar users to generate recommendations. Different from the models in the literature, which combine only one or some types of social information into recommendation systems, in our model we try to incorporate all available information into a unified framework.

3 Problem formalization

In this study, to enhance the prediction precision of recommender systems, we incorporate different social information in social media into a probabilistic model. The user behavior is determined by personal interests and the influences of other users. We divide social information into two categories, user-specific information and relationship-specific information. User-specific information includes user profile, user tags, etc., which reflect the preferences of each user. Relationship-specific information may be a user friend relationship, follower-followee relationship, or trust relationship. Through these directed or undirected relationships other users can affect one's decision. In this study, to be consistent, all social relationships are defined as undirected relationships. In addition, when we predict the users' ratings, some users have rated parts of items. There exist some known ratings, which reflect the historical behavior of the user and are the main information that traditional recommender systems use, so the known ratings need to be considered in our model.

Based on the above analysis, we can formalize the problem studied in this work. Given a set of users U and a set of items V, user-specific information X,

item-specific information Y, user relationship network G, as well as known ratings $\boldsymbol{R}^{\mathrm{L}}$, we conclude all of them to be the set of observations $O=(U; V; X; Y; G; \boldsymbol{R}^{\mathrm{L}})$. Notations are summarized in Table 1. Our purpose is to learn the predictive function to predict unknown ratings. The function can be defined as

$$f: O=(U; V; X; Y; G; \boldsymbol{R}^{\mathrm{L}})\rightarrow \boldsymbol{R}. \quad (1)$$

In this study, by dividing the information that affects user behavior into two categories, two main types of factors are defined. When we consider only the user-specific information, our approach is similar to the content-based recommendation approach. When we consider only the relationship-specific information, our approach can be considered as a recommendation approach incorporating social relationships. Hence, our proposed model has more adaptability.

Table 1 Notations

Symbol	Description
U	The set of users
V	The set of items
N	Number of users
M	Number of items
G	Relationship-specific information
S	Correlation between users and items
X	User-specific information
Y	Item information
$\boldsymbol{R}$	User-item rating matrix
$\boldsymbol{R}_i$	The rating vector of user u_i
$\boldsymbol{R}^{\mathrm{L}}$	Known user-item rating matrix
R_{ij}	The rating that user u_i gives to item v_j

4 Our model

4.1 Basic idea

Based on the factor graph model theory, we formalize both user- and relationship-specific information into a semi-supervised probabilistic model as feature functions. We can formalize the contribution degrees of the two types of information as weights of the feature functions. Solving the probabilistic model includes both estimating the weights of feature functions and inferring unknown ratings in the user-item rating matrix. The objective function in the probabilistic model is a posterior probability distribution of unknown rating variables, given observations.

4.2 Model specification

Given the observation data, we need to build the corresponding model to predict the unknown ratings. Here, we give a simple example (Fig. 1): there are five users (u_1, u_1, …, u_5), five items (v_1, v_2, …, v_5), user social relationships between users (dotted lines), item information (y_1, y_2, …, y_5), user-specific information (x_1, x_2, …, x_5), and their known ratings (solid lines); they are the input of our model. In the model, each user u_i has a rating vector $\boldsymbol{R}_i$, and the entry R_{ij} in $\boldsymbol{R}_i$ represents the rating that u_i gives to v_j. Note that although real ratings are discrete, in order to generate more accurate sorting of items, in this study each predicted rating R_{ij} is defined to be continuous. The ratings are partially known, so our task is to predict

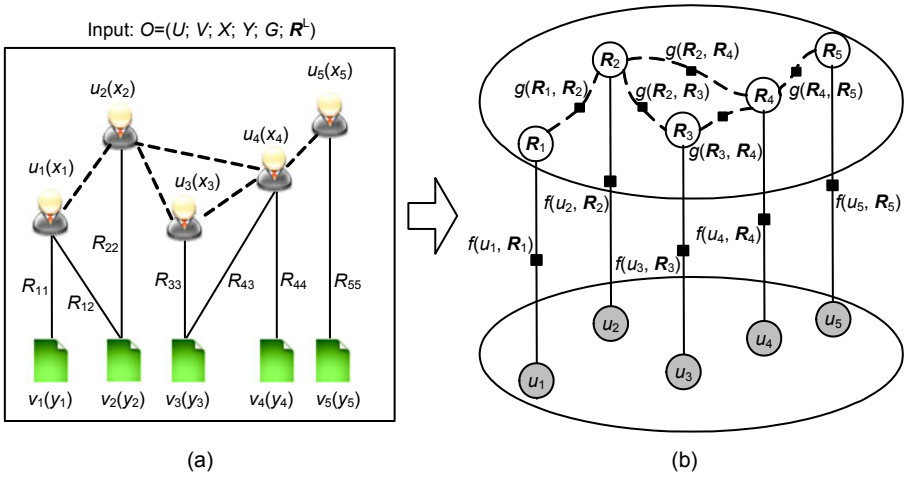

Fig. 1 Graphical representation of our model: (a) a simple example; (b) graphical representation of the model

the missing entries in each rating vector. In our model, we use state factor f and correlation factor g to model the effect of user- and relationship-specific information on user's ratings, respectively. We define the two types of factors as follows:

1. State factor: $f(u_i, \boldsymbol{R}_i)$ denotes the posterior probability of user u_i, given the correlation between user u_i and items V. It reflects the effect of user-specific information on the user's final decision.

2. Correlation factor: $g(\boldsymbol{R}_i, G(\boldsymbol{R}_i))$ represents the correlation between the ratings of different users, where $G(\boldsymbol{R}_i)$ is the set of correlated vectors of other users to $\boldsymbol{R}_i$. It reflects the effect of relationship-specific information on the user's final decision.

Given the observation data G and S, where S represents the correlation between users and items which is generated by users' own preferences and can be obtained from X and Y, the joint distribution over $\boldsymbol{R}$ can be defined as

$$\begin{aligned} P(\boldsymbol{R}\,|\,G,S) &= \prod_{i=1}^{N} f(u_i,\boldsymbol{R}_i)\,g(\boldsymbol{R}_i,G(\boldsymbol{R}_i)) \\ &= \prod_{i=1}^{N}\prod_{j=1}^{M} f(u_i,R_{ij})\,g(R_{ij},G(R_{ij})). \end{aligned} \tag{2}$$

There are different ways to instantiate the two types of factors. In this study, the exponential-linear functions are used. In particular, we define the state factor as

$$f(u_i,R_{ij}) = \frac{1}{Z_{\boldsymbol{\alpha}}}\exp[\boldsymbol{\alpha}^{\mathrm{T}}\cdot\boldsymbol{\Phi}(u_i,R_{ij})], \tag{3}$$

where $\boldsymbol{\alpha}$ is a weighting vector and $\boldsymbol{\Phi}$ is defined as a vector of state factor feature functions. In the same way, the correlation factor can be defined as

$$g(R_{ij},G(R_{ij})) = \frac{1}{Z_{\boldsymbol{\beta}}}\exp\left[\sum_{R_{kj}\in G(R_{ij})}\boldsymbol{\beta}^{\mathrm{T}}\cdot\boldsymbol{\Omega}(R_{ij},R_{kj})\right], \tag{4}$$

where $\boldsymbol{\beta}$ is a weighting vector and $\boldsymbol{\Omega}$ is defined as a vector of correlation factor feature functions. Here we assume that the predicted ratings are continuous, so that users or items can be ranked according to the predicted continuous ratings to generate more accurate recommendation results. $Z_{\boldsymbol{\alpha}}$ and $Z_{\boldsymbol{\beta}}$ are the normalization factors (also called the partition function), defined as

$$Z_{\boldsymbol{\alpha}} = \int_{\boldsymbol{R}}\exp[\boldsymbol{\alpha}^{\mathrm{T}}\cdot\boldsymbol{\Phi}(u_i,R_{ij})]\mathrm{d}\boldsymbol{R}, \tag{5}$$

$$Z_{\boldsymbol{\beta}} = \int_{\boldsymbol{R}}\exp\left[\sum_{R_{kj}\in G(R_{ij})}\boldsymbol{\beta}^{\mathrm{T}}\cdot\boldsymbol{\Omega}(R_{ij},R_{kj})\right]\mathrm{d}\boldsymbol{R}. \tag{6}$$

4.3 Model learning

The task of model learning in our approach is to estimate a parameter configuration $\boldsymbol{\theta}=[\boldsymbol{\alpha}, \boldsymbol{\beta}]$ to maximize the log-likelihood of observation information (known ratings). Here, to simplify the presentation, all factor functions for a rating R_{ij} are concatenated as $\boldsymbol{w}(R_{ij}) = \left(\boldsymbol{\Phi}(u_i,R_{ij})^{\mathrm{T}}, \sum_{R_{kj}}\boldsymbol{\Omega}(R_{ij},R_{kj})^{\mathrm{T}}\right)^{\mathrm{T}}$. The joint probability defined in Eq. (2) can be written as

$$P(\boldsymbol{R}\,|\,G,S) = \frac{1}{Z}\exp\left[\boldsymbol{\theta}^{\mathrm{T}}\sum_{i=1}^{N}\sum_{j=1}^{M}\boldsymbol{w}(R_{ij})\right] = \frac{1}{Z}\exp[\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{W}(R_{ij})], \tag{7}$$

where $\boldsymbol{W}$ is the aggregation of factor functions over all ratings, i.e., $\boldsymbol{W}(R_{ij}) = \sum_i\sum_j \boldsymbol{w}(R_{ij})$. Z is a normalization factor, and can be written as

$$Z = \int_{\boldsymbol{R}}\exp[\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{W}(R_{ij})]\mathrm{d}\boldsymbol{R}. \tag{8}$$

One problem for our model learning is that the ratings are partially known. To deal with this, we use the known ratings to predict the unknown ratings. Here $\boldsymbol{R}-\boldsymbol{R}^{\mathrm{L}}$ represents the entries in rating matrix $\boldsymbol{R}$ except the known ratings $\boldsymbol{R}^{\mathrm{L}}$. So, the log-likelihood objective function $\ell(\boldsymbol{\theta})$ can be defined as

$$\begin{aligned} \ell(\boldsymbol{\theta}) &= \ln P(\boldsymbol{R}^{\mathrm{L}}\,|\,G,S) \\ &= \ln\int_{\boldsymbol{R}-\boldsymbol{R}^{\mathrm{L}}}\frac{1}{Z}\exp[\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{W}(R_{ij})]\mathrm{d}(\boldsymbol{R}-\boldsymbol{R}^{\mathrm{L}}) \\ &= \ln\int_{\boldsymbol{R}-\boldsymbol{R}^{\mathrm{L}}}\exp[\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{W}(R_{ij})]\mathrm{d}(\boldsymbol{R}-\boldsymbol{R}^{\mathrm{L}}) - \ln Z \\ &= \ln\int_{\boldsymbol{R}-\boldsymbol{R}^{\mathrm{L}}}\exp[\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{W}(R_{ij})]\mathrm{d}(\boldsymbol{R}-\boldsymbol{R}^{\mathrm{L}}) \\ &\quad - \ln\int_{\boldsymbol{R}}\exp[\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{W}(R_{ij})]\mathrm{d}\boldsymbol{R}. \end{aligned} \tag{9}$$

Maximize the log-likelihood objective function $\ell(\boldsymbol{\theta})$, i.e.,

$$\boldsymbol{\theta}^{*} = \operatorname{argmax}\ell(\boldsymbol{\theta}). \tag{10}$$

A gradient-based algorithm can be applied to maximize the conditional log-likelihood. Specifically, we first write the gradient of each $\boldsymbol{\theta}$ with regard to the objective function:

$$\begin{aligned}\frac{\partial \ell}{\partial \boldsymbol{\theta}} &= \frac{\partial \ln P(\boldsymbol{R}^{\mathrm{L}} \mid G,S)}{\partial \boldsymbol{\theta}} \\ &= \frac{\int_{\boldsymbol{R}-\boldsymbol{R}^{\mathrm{L}}} \exp[\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{W}(R_{ij})]\cdot\boldsymbol{W}(R_{ij})\mathrm{d}(\boldsymbol{R}-\boldsymbol{R}^{\mathrm{L}})}{\int_{\boldsymbol{R}-\boldsymbol{R}^{\mathrm{L}}} \exp[\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{W}(R_{ij})]\mathrm{d}(\boldsymbol{R}-\boldsymbol{R}^{\mathrm{L}})} \\ &\quad - \frac{\int_{\boldsymbol{R}} \exp[\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{W}(R_{ij})]\cdot\boldsymbol{W}(R_{ij})\mathrm{d}\boldsymbol{R}}{\int_{\boldsymbol{R}} \exp[\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{W}(R_{ij})]\mathrm{d}\boldsymbol{R}} \\ &= E_{P_{\theta}(\boldsymbol{R}|\boldsymbol{R}^{\mathrm{L}},G,S)}(\boldsymbol{W}(R_{ij})) - E_{P_{\theta}(\boldsymbol{R}|G,S)}(\boldsymbol{W}(R_{ij})).\end{aligned} \tag{11}$$

However, we need to compute the expectations under the model distribution, which is difficult due to the normalization factor Z. As the graphical structure can be arbitrary and may contain cycles, the variables are continuous and non-Gaussian, which makes it intractable to compute the second expectation directly in closed form. Here, we use nonparametric belief propagation (NBP) (Sudderth *et al*., 2010) to approximate the gradients. Note that the NBP process needs to be performed twice in each iteration; one time is to estimate the marginal probability $P(\boldsymbol{R}|G,S)$ and the other to estimate $P(\boldsymbol{R}|\boldsymbol{R}^{\mathrm{L}},G,S)$. Finally, using the learned gradient, each parameter can be updated with a learning rate η. The learning algorithm is summarized in Algorithm 1.

Algorithm 1 Parameter estimation

Input: learning rate η

Output: $\boldsymbol{\theta}=[\boldsymbol{\alpha}, \boldsymbol{\beta}]$

Initialize $\boldsymbol{\theta}$

repeat

Calculate $E_{P_{\theta}(\boldsymbol{R}|\boldsymbol{R}^{\mathrm{L}},G,S)}(\boldsymbol{W}(R_{ij}))$ using NBP

Calculate $E_{P_{\theta}(\boldsymbol{R}|G,S)}(\boldsymbol{W}(R_{ij}))$ using NBP

Calculate the gradient of $\boldsymbol{\theta}$ according to Eq. (11)

Update parameter $\boldsymbol{\theta}$ with the learning rate η:

$$\boldsymbol{\theta}_{\mathrm{new}} = \boldsymbol{\theta}_{\mathrm{old}} - \eta\cdot\frac{\partial \ell}{\partial \boldsymbol{\theta}}$$

until convergence

Return $\boldsymbol{\theta}_{\mathrm{new}}$

4.4 Inferring unknown ratings

Now we describe how to infer unknown ratings. Using learned parameters $\boldsymbol{\theta}$, we can predict all unknown ratings by finding a rating configuration which maximizes the joint probability:

$$\boldsymbol{R}^{*}=\operatorname{argmax} P(\boldsymbol{R}|\boldsymbol{R}^{\mathrm{L}}, G, S). \tag{12}$$

Again, we use NBP to calculate the marginal probability of each unknown rating $P(R_{ij}|\boldsymbol{R}^{\mathrm{L}}, G, S)$, and then the value of the unknown ratings can be predicted as the value with the largest marginal probability.

4.5 Distributed learning

Since real recommender systems may contain millions of users and items, it is necessary for the learning algorithm to have high scalability. To deal with this, we adopt a distributed learning algorithm based on the message passing interface (MPI). The learning process can be divided into two steps: (1) computing the gradient of each parameter via NBP; (2) updating all parameters with the gradient-based algorithm. The first step is much more expensive than the second one. In this study a distributed method is proposed to speed up the first step. In our model, relational dependencies between items are not considered, so the distributed method does not decrease the performance of the learning algorithm. In this algorithm the master-slave architecture is adopted; i.e., one master node is used to optimize parameters, and the other slave nodes are used to calculate gradients. In the initial stage, we partition the set of items into P roughly equal subsets, where P denotes the number of slave processors. In each iteration, the master node sends the updated parameters $\boldsymbol{\theta}$ firstly to all slaves. Then slave nodes compute the marginal probabilities by performing NBP on the corresponding subset, then further calculate the gradient of each parameter and send them back to the master node. Lastly, the master gathers all gradients received from different subsets and sums up them, and further updates parameters by performing the gradient descent algorithm.

5 Experiment analysis

We evaluated our model for item recommendation with known ratings. The goal is to predict the missing entries in the user-item rating matrix by exploiting the observed ratings, the underlying user

information, and social relationships. We ran experiments on two datasets, Douban and Last.fm, and compared the prediction results of our model with several state-of-the-art recommendation methods.

5.1 Factor setting

In the experiments, just a state factor and a correlation factor were considered. We used social tagging information to define the state factor and friend relationships to define the correlation factor, respectively. Define $\boldsymbol{T}$ as the vector of all tags. Each entry is one tag. Because each user may give the same tags to different items, we define $\mathbf{UT}_i$ as a numeric vector, with the entry $\mathbf{UT}_i(k)$ in $\mathbf{UT}_i$ being the total number of times of the kth tag in $\boldsymbol{T}$ that user u_i gives, and $\mathbf{IT}_j$ a numeric vector with the entry $\mathbf{IT}_j(k)$ in $\mathbf{IT}_j$ being the total number of times of the kth tag in $\boldsymbol{T}$ that all users apply to item v_j. For example, there are five tags, i.e., $\boldsymbol{T}$=('America film', 'Tom', 'belief', '1994', 'classic'). If user u_i only applies 'Tom' once, and 'classic' three times, then $\mathbf{UT}_i$=(0, 1, 0, 0, 3); if item v_j is tagged by all users with 'America film' 150 times, 'Tom' 200 times, 'belief' 50 times, '1994' 10 times, 'classic' 100 times, then $\mathbf{IT}_j$=(150, 200, 50, 10, 100). To measure the correlation between user u_i and item v_j, we need to normalize them first:

$$\mathbf{NUT}_i=\frac{\mathbf{UT}_i}{\sum_k \mathbf{UT}_i(k)},\quad \mathbf{NIT}_j=\frac{\mathbf{IT}_j}{\sum_k \mathbf{IT}_j(k)}. \tag{13}$$

So, the state factor feature function can be defined as

$$\boldsymbol{\Phi}(u_i,R_{ij})=\frac{\mathbf{NUT}_i\cdot\mathbf{NIT}_j}{|\mathbf{NUT}_i|\cdot|\mathbf{NIT}_j|}. \tag{14}$$

The above definition of the state factor feature has the following intuition: if there exists a high correlation between the tags that user u_i provides and the tags that all users apply to item v_j, the rating R_{ij} that user u_i gives to item v_j is apt to have a high value. For the correlation factor feature function, we define a binary feature function, i.e.,

$$\Omega(R_{ij},R_{kj})=\begin{cases}\dfrac{1}{|\mathrm{NB}(u_i)||\mathrm{NB}(u_k)|}, & e_{ik}=1,\\ 0, & e_{ik}\neq 1,\end{cases} \tag{15}$$

where $\mathrm{NB}(u_i)$ and $\mathrm{NB}(u_k)$ are the sets of neighbors of u_i and u_k, respectively. Intuitively, if users u_i and u_k have a direct friend relationship, the smaller the number of neighbors of users u_i and u_k, the closer the ratings R_{ij} and R_{kj} they give to any item v_j. If there is no friend relationship between the two users, the direct correlation between R_{ij} and R_{kj} does not exist.

5.2 Datasets

The first dataset we used is from Douban (http://www.douban.com). Douban is the largest online Chinese language music, book, and movie database and one of the largest online communities in China. Users can review and assign 5-scale integer ratings (from 1 to 5) to music, books, and movies. It also provides a social networking service similar to Facebook. Users can build friendship with others through their email accounts. Douban is an ideal data source for social recommendation research. Douban provides various interesting groups for users to join. Currently, it has more than 700 groups under the 'Movie' subcategory. We crawled all the users in 50 groups which had the largest numbers of members. Then, according to their friend relationships, we took the users in the maximum connected subgraph and used these users as seeds to further crawl their movie ratings. Finally, we obtained 46 015 unique users, 8246 unique movies, and 1 176 243 movie ratings, and the density of rating was 0.31%. In the social friends network, there were in total 39 438 links between users, and the total number of unique tags was 165 096.

The second dataset we employed for evaluation is Last.fm (http://www.last.fm). Last.fm is a music social network that allows users to create a profile and augment it with the music tracks that they listen to, either from within the website itself or from their own private music collections. We used the Last.fm dataset (http://www.dcs.gla.ac.uk/~jj/data/lastfm dataset.htm) which was collected by Konstas *et al*. (2009). In this dataset, we took tracks as items. There are no clear ratings, and the playcount of every track by each user, i.e., the number of times the track has been listened to, is treated as the rating in the user-item rating matrix. In this study, we removed all tracks that had been listened to by less than 30 users along with their corresponding tags; in the remaining data the playcount ranged from 1 to 6966. Furthermore, in accordance

with the ascending order of the playcount value, we divided all playcounts into 10 sets of the same size. For each playcount in one set, we took an integer from {1, 2, …, 10} as its corresponding rating value in the user-item matrix. Finally, the dataset contained 3148 users, 7458 tracks, 2439 tags, 510125 ratings, and 5616 unique bonds of friendship among the users, and the density of playcounts was 2.17%.

5.3 Metric

We used a popular metric, the root mean square error (RMSE), to measure the prediction quality of our proposed approach in comparison with other collaborative filtering and social recommendation methods. RMSE is defined as

$$\mathrm{RMSE}=\sqrt{\frac{1}{L}\sum_{i,j}(R_{ij}-\hat{R}_{ij})^2}, \tag{16}$$

where $\hat{R}_{ij}$ denotes the rating that user u_i gives to item v_j, R_{ij} denotes the rating that user u_i gives to item v_j as predicted by the method, and L denotes the number of tested ratings.

It is obvious from the above definition that a smaller RMSE value means a better performance.

5.4 Comparisons

In this subsection, to show the effectiveness of our model, we compared the recommendation results of our model with the following methods:

1. NMF: this method was first proposed by Lee and Seung (1999) for image analysis. Recently, it was widely used in collaborative filtering. Only user-item ratings are used for recommendations.

2. PMF: this method was proposed by Salakhutdinov and Mnih (2008). Also, only user-item ratings are used for recommendations.

3. SR (social regularization): this approach was proposed by Ma *et al*. (2011b). It is a state-of-the-art social recommendation algorithm that combines information from both the rating matrix and the social network relationship.

4. KPMF: this approach was proposed by Zhou *et al*. (2012). It outperforms the social regularization algorithm in the experiments in their paper. It incorporates a social network relationship and the rating matrix for recommendation.

5. CFTF (collaborative filtering incorporating social tagging and friendships): this approach was proposed by Konstas *et al*. (2009). It combines information from both social tagging and social network relationship.

For the two datasets we used different training data settings to test the algorithms. For the Douban dataset, we chose 90%, 80%, and 70% of the ratings randomly from the user-item rating matrix as the training data for predicting the remaining 10%, 20%, and 30% of ratings, respectively. While the Last.fm dataset was much denser, we chose 80%, 60%, and 40% as the training data settings. We carried out the random selection five times independently, and showed the average results. In all the experiments, we set the value of η as a trivial value 0.001. Table 2 shows that our method consistently outperformed other methods under all the three settings of both datasets. Compared to NMF, PMF, SR, and KPMF, our method considers more information for recommendation. Compared to CTCF, our method can determine the contribution degrees of different factors automatically by optimizing the model parameters, so it is not surprising that our method achieved better performance. Moreover, the RMSE values of almost all the methods on the Douban dataset were smaller than those on the Last.fm dataset. Several problems may result in this situation. First, in the Last.fm dataset, the process of transforming the playcounts to ratings inevitably generated noises. Second, each user can rate the items on a 5-point integer, while the value of each rating in the transformed Last.fm dataset was a 10-point integer, which led to a larger RMSE. Third, from the Last.fm dataset, we did not know the total number of times of each tag one user gave to all tracks and the total number of times of each tag all users applied to one track. For any tag, the entries in UT and IT in the Last.fm dataset can just take the value 0 or 1.

5.5 Factor contribution analysis

In this subsection, we evaluated the contributions of different factors defined in our model. We separated the correlation factor and the state factor, and evaluated the performance, respectively. Table 3 shows the analytical results of the contribution of different factors.

Table 2 Performance comparisons

Dataset	RMSE					
	NMF	PMF	SR	KPMF	CTCF	Our method
Douban (70%)	1.2146	1.1974	1.1325	1.1179	0.9323	**0.7787**
Douban (80%)	1.1831	1.1773	1.1206	1.1158	0.9056	**0.7746**
Douban (90%)	1.1403	1.1385	1.0973	1.1087	0.8932	**0.7077**
Last.fm (40%)	1.4790	1.4761	1.3531	1.3330	1.3524	**1.1071**
Last.fm (60%)	1.4576	1.4023	1.3367	1.3075	1.2331	**1.0655**
Last.fm (80%)	1.4073	1.3667	1.3265	1.2238	1.1892	**1.0427**

NMF: non-negative matrix factorization; PMF: probabilistic matrix factorization; SR: social regularization; KPMF: kernelized probabilistic matrix factorization; CTCF: collaborative filtering incorporating social tagging and friendships

Table 3 Factor contribution analysis

Dataset	RMSE		
	State factor	Correlation factor	Both factors used
Douban (70%)	1.1872	1.4632	**0.7787**
Douban (80%)	1.1493	1.4456	**0.7746**
Douban (90%)	1.1325	1.4073	**0.7077**
Last.fm (40%)	1.3071	1.5366	**1.1071**
Last.fm (60%)	1.3320	1.5127	**1.0655**
Last.fm (80%)	1.3212	1.4972	**1.0427**

We found that the two types of factors were useful in predicting the missing ratings, but that the performance of using either factor alone was not high. By combining these two types of factors, the performance was significantly improved.

Note that in our experiment we can define more state factors and correlation factors; for example, the similarities between users and those between items can be computed by considering social tags or other information and taken as correlation factors. To simplify the experiment, however, we just considered one state factor and one correlation factor.

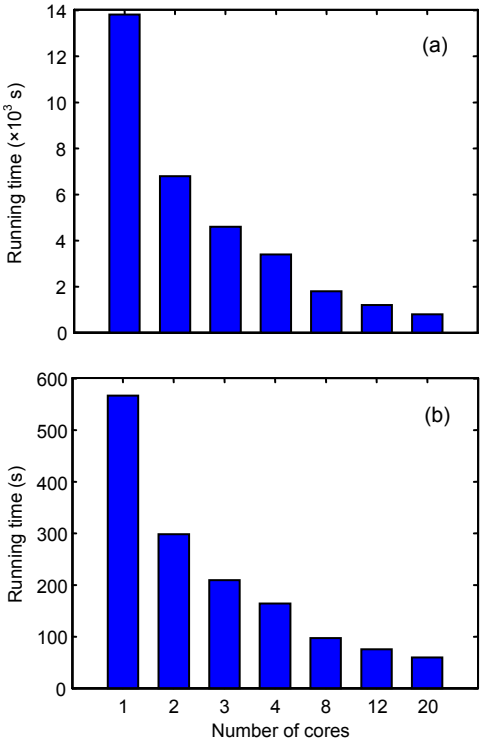

Fig. 2 Running time of one iteration vs. the number of cores: (a) Douban; (b) Last.fm

5.6 Scalability performance

We now performed an analysis to evaluate the scalability performance of our proposed distributed learning algorithm on the Douban dataset. In our experiment, first the state factor of each user and the correlation factor value between users need to be computed using Eqs. (14) and (15), respectively. The computation of the state factors consumed a long period of time, but they were computed only once. Then we performed our distributed learning algorithm. Fig. 2 shows the running time of one iteration in the distributed learning algorithm with different numbers of slave nodes (1, 2, 3, 4, 8, 12, and 20 cores) used. Figs. 2a and 2b describe the average running time of one iteration in all three sampling settings of the Douban dataset and Last.fm dataset, respectively. When the number of cores increased, the speed of the distributed algorithm increased remarkably, but the speedup inevitably decreased. Fig. 3a shows the RMSE results under different numbers of iterations on the Douban dataset when the sampling percentages of the training set were 90%, 80%, and 70%, respectively; Fig. 3b shows the RMSE results under

different numbers of iterations on the Last.fm dataset when the sampling percentages of the training set were 40%, 60%, and 80%, respectively. The algorithm quickly converged after five iterations in the three cases of both datasets. So, we set the maximum number of iterations to 10 and the threshold for the change of θ to 0.001.

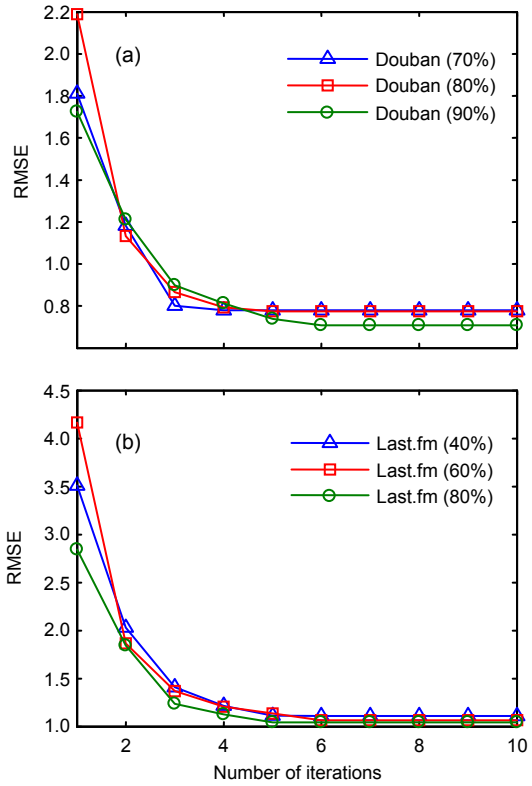

Fig. 3 RMSE under different numbers of iterations: (a) Douban; (b) Last.fm

We implemented all the codes in C++, and all experiments with a server running Windows Server 2003, which has a configuration of Intel Core i5 CPU (2.4 GHz) and 8 GB memory. Moreover, our distributed learning algorithm was conducted on MPI.

6 Conclusions and future work

In this paper, we study how to incorporate different social information into recommendation algorithms to help improve the prediction accuracy of recommender systems. We formulate the problem in a semi-supervised probabilistic framework. Based on the intuition that each user's behavior on the Web should be determined by both the user's preference and the influence of the user's friends, we define two types of feature factors to describe the two types of effects, respectively. The proposed model is quite general since it can combine different information to enhance the recommender system by defining different factors. Experimental results using the Douban and Last.fm datasets demonstrate that our proposed method is promising. Furthermore, a distributed learning algorithm is proposed to scale up our method to handle large datasets. Experimental analysis shows good parallel efficiency of our distributed learning algorithm.

Here we employ only social tagging information to define a state factor and friend relationships between users to define a correlation factor. In the next phase of work, we will define more state factors by considering user profile, item profile, etc. Other social relationships such as follower-followee relationship trust and distrust relationship will be incorporated into our model by defining different correlation factors. Two further questions need to be investigated in our future work: whether different types of information are useful in improving the prediction quality, and whether evaluating their respective contributions will generate better-quality results. Moreover, when incorporating the social relationship information, we ignore the influence diffusion or propagation between users. A more accurate approach is to consider the influence diffusion process between users when we define the correlation factors. This consideration will help alleviate the data sparseness problem and potentially increase the prediction accuracy.

References

Agarwal, D., Chen, B., 2009. Regression-based Latent Factor Models. Proc. 15th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, p.19-28. [doi:10.1145/1557019.1557029]

Balabanovic, M., Shoham, Y., 1997. Content-based, collaborative recommendation. *Commun. ACM*, **40**(3):66-72. [doi:10.1145/245108.245124]

Breese, J.S., Heckerman, D., Kadie, C.M., 1998. Empirical Analysis of Predictive Algorithm for Collaborative Filtering. Proc. 14th Conf. on Uncertainty in Artificial Intelligence, p.43-52.

Deshpande, M., Karypis, G., 2004. Item-based top-*N* recommendation algorithms. *ACM Trans. Inf. Syst.*, **22**(1):143-177. [doi:10.1145/963770.963776]

Jahrer, M., Tuscher, A., Legenstein, R., 2010. Combining Predictions for Accurate Recommender Systems. Proc. 16th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, p.693-702. [doi:10.1145/1835804.1835893]

Konstas, I., Stathopoulos, V., Jose, J.M., 2009. On Social Networks and Collaborative Recommendation. Proc. 32nd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, p.195-202. [doi:10.1145/1571941.1571977]

Koren, Y., 2008. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. Proc. 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.426-434. [doi:10.1145/1401890.1401944]

Koren, Y., Bell, R.M., 2011. Advances in Collaborative Filtering. *In:* Ricci, F., Rokach, L., Shapira, B., *et al*. (Eds.), Introduction to Recommender Systems Handbook. Springer US, p.145-186. [doi:10.1007/978-0-387-85820-3_5]

Koren, Y., Bell, R.M., Volinsky, C., 2009. Matrix factorization techniques for recommender systems. *Computer*, **42**(8): 30-37. [doi:10.1109/MC.2009.263]

Kschischang, F.R., Frey, B.J., Loeliger, H., 2001. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory*, **47**(2):498-519. [doi:10.1109/18.910572]

Kurucz, M., Benczúr, A.A., Csalogány, K., 2007. Methods for Large Scale SVD with Missing Values. Proc. KDD Cup and Workshop, p.31-38.

Lee, D.D., Seung, H.S., 1999. Learning the parts of objects by non-negative matrix factorization. *Nature*, **401**(6755): 788-791. [doi:10.1038/44565]

Lops, P., Gemmis, M.D., Semeraro, G., 2011. Content-Based Recommender Systems: State of the Art and Trends. *In:* Ricci, F., Rokach, L., Shapira, B., *et al*. (Eds.), Introduction to Recommender Systems Handbook. Springer US, p.73-105. [doi:10.1007/978-0-387-85820-3_3]

Lu, Y., Tsaparas, P., Ntoulas, A., Polanyi, L., 2010. Exploiting Social Context for Review Quality Prediction. Proc. 19th Int. Conf. on World Wide Web, p.691-700. [doi:10.1145/1772690.1772761]

Ma, H., Yang, H., Lyu, M.R., King, I., 2008. Sorec: Social Recommendation Using Probabilistic Matrix Factorization. Proc. 17th ACM Conf. on Information and Knowledge Management, p.931-940. [doi:10.1145/1458082.1458205]

Ma, H., King, I., Lyu, M.R., 2011a. Learning to recommend with explicit and implicit social relations. *ACM Trans. Intell. Syst. Technol.*, **2**(3), Article 29. [doi:10.1145/1961189.1961201]

Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I., 2011b. Recommender Systems with Social Regularization. Proc. 4th ACM Int. Conf. on Web Search and Data Mining, p.287-296. [doi:10.1145/1935826.1935877]

Mei, Q., Cai, D., Zhang, D., Zhai, C., 2008. Topic Modeling with Network Regularization. Proc. 17th Int. Conf. on World Wide Web, p.101-110. [doi:10.1145/1367497.1367512]

Salakhutdinov, R., Mnih, A., 2008. Probabilistic matrix factorization. *Adv. Neur. Inf. Process. Syst.*, **20**:1257-1264.

Sarwar, B., Karypis, G., Konstan, J., Riedl, J., 2001. Item-Based Collaborative Filtering Recommendation Algorithms. Proc. 10th Int. Conf. on World Wide Web, p.285-295. [doi:10.1145/371920.372071]

Shen, Y., Jin, R., 2012. Learning Personal + Social Latent Factor Model for Social Recommendation. Proc. 18th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.1303-1311. [doi:10.1145/2339530.2339732]

Sinha, R., Swearingen, K., 2001. Comparing Recommendations Made by Online Systems and Friends. Proc. DELOSNSF Workshop on Personalization and Recommender Systems in Digital Libraries, v.106.

Sudderth, E.B., Ihler, A.T., Isard, M., Freeman, W.T., Willsky, A.S., 2010. Nonparametric belief propagation. *Commun. ACM*, **53**(10):95-103. [doi:10.1145/1831407.1831431]

Wang, J., Zhang, Y., 2011. Utilizing Marginal Net Utility for Recommendation in E-commerce. Proc. 34th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, p.1003-1012. [doi:10.1145/2009916.2010050]

Xin, X., King, I., Deng, H., Lyu, M.R., 2009. A Social Recommendation Framework Based on Multi-scale Continuous Conditional Random Fields. Proc. 18th ACM Conf. on Information and Knowledge Management, p.1247-1256. [doi:10.1145/1645953.1646111]

Yang, S., Long, B., Smola, A.J., Zha, H., Zheng, Z., 2011. Collaborative Competitive Filtering: Learning Recommender Using Context of User Choice. Proc. 34th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, p.295-304. [doi:10.1145/2009916.2009959]

Yang, X., Steck, H., Guo, Y., Liu, Y., 2012a. On Top-*k* Recommendation Using Social Networks. Proc. 6th ACM Conf. on Recommender Systems, p.67-74. [doi:10.1145/2365952.2365969]

Yang, X., Steck, H., Liu, Y., 2012b. Circle-Based Recommendation in Online Social Networks. Proc. 18th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.1267-1275. [doi:10.1145/2339530.2339728]

Zhou, T., Shan, H., Banerjee, A., Sapiro, G., 2012. Kernelized Probabilistic Matrix Factorization: Exploiting Graphs and Side Information. Proc. 12th SIAM Int. Conf. on Data Mining, p.403-414.