Improving the Quality of Recommendations for Users and Items in the Tail of Distribution

LIANG HU and LONGBING CAO, University of Technology, Sydney
JIAN CAO, Shanghai Jiao Tong University
ZHIPING GU, Shanghai Technical Institute of Electronics & Information
GUANDONG XU, University of Technology, Sydney
JIE WANG, Stanford University

Short-head and long-tail distributed data are widely observed in the real world. The same is true of recommender systems (RSs), where a small number of popular items dominate the choices and feedback data while the rest only account for a small amount of feedback. As a result, most RS methods tend to learn user preferences from popular items since they account for most data. However, recent research in e-commerce and marketing has shown that future businesses will obtain greater profit from long-tail selling. Yet, although the number of long-tail items and users is much larger than that of short-head items and users, in reality, the amount of data associated with long-tail items and users is much less. As a result, user preferences tend to be popularity-biased. Furthermore, insufficient data makes long-tail items and users more vulnerable to shilling attack. To improve the quality of recommendations for items and users in the tail of distribution, we propose a coupled regularization approach that consists of two latent factor models: C-HMF, for enhancing credibility, and S-HMF, for emphasizing specialty on user choices. Specifically, the estimates learned from C-HMF and S-HMF recurrently serve as the empirical priors to regularize one another. Such coupled regularization leads to the comprehensive effects of final estimates, which produce more qualitative predictions for both tail users and tail items. To assess the effectiveness of our model, we conduct empirical evaluations on large real-world datasets with various metrics. The results prove that our approach significantly outperforms the compared methods.

CCS Concepts: • **Information systems** → **Information retrieval**; **Retrieval tasks and goals**; *Recommender systems*; • **Computing methodologies** → **Artificial intelligence**; **Machine learning**; *Multi-task learning*

Additional Key Words and Phrases: Recommender systems, long tail, recurrent mutual regularization, multi-objective learning, trust and reputation systems

ACM Reference Format:
Liang Hu, Longbing Cao, Jian Cao, Zhiping Gu, Guandong Xu, and Jie Wang. 2017. Improving the quality of recommendations for users and items in the tail of distribution. ACM Trans. Inf. Syst. 35, 3, Article 25 (June 2017), 37 pages.
DOI: http://dx.doi.org/10.1145/3052769

1. INTRODUCTION

We are leaving the information age and entering the recommendation age [Anderson 2006]. Because of this, recommender systems (RSs) are playing an increasingly important role than ever before. Collaborative filtering (CF) is a core component of modern RSs; it leverages feedback from other users and items to generate recommendations for a target user. However, CF techniques are still challenged by complicated real-world data characteristics [Su and Khoshgoftaar 2009]. On the one hand, some of the challenges arise from the distribution of real-world data in nature. It is known that a lot of real-world data can be observed following a long-tail—a.k.a., power-law—distribution. Due to a lack of sufficient data for most users and items, data sparsity and cold start are two of the most common research issues addressed by the study of RSs. On the other hand, other challenges may be caused by human behavior. For instance, shilling attack is one such typical issue; this refers to some users giving lots of positive feedback for their own items and negative feedback for their competitors' items. Since the basic idea of CF is to predict ratings in terms of the related data associated with other users and items, insufficient data and spam data will obviously deteriorate recommendation results, especially for those users and items in the tail of distribution.

To date, most research has focused on improving the accuracy of RSs. However, simply improving the accuracy by one or two percent will hardly result in a better user experience or a greater business benefit. Here, we give an intuitive interpretation from the long-tail distribution. A long-tail distribution implies skewed data that has a short head and a long tail, that is, a small number of popular items in the head part, which account for most of the data, whereas the large number of items in the tail only account for a small amount of data. Here, we use the experimental *Rich Epinions Dataset* (RED) [Meyffret et al. 2012] as a demonstration. This dataset was crawled from the well-known online review website epinions.com, which contains a total of 1,127,673 reviews given by 113,629 users on 317,755 items. Each review contains a user rating, and the density of this dataset is only 0.003%. The left- and right-hand sides of Figure 1 depict, respectively, the long-tail distribution for items and for users. We find that only a few items and users in the short head have sufficient ratings while a large number of items and users in the long tail have less than ten ratings each. Such a skewed data distribution causes RSs to learn users' preferences largely from popular items because these items account for the majority of the data. As a result, the improvement of recommendations is largely determined by popular items. However, such improvement for popular items is trivial because popular items are likely already known by most users who can make the decision to choose them or not. Moreover, Anderson's well-known research suggests that future businesses will obtain more profit from long-tail selling [Anderson 2006]. Motivated by these observations, we focus on improving the quality of recommendations for tail users and items, which we believe will be a great benefit for both business profits and the user's experience.

1.1. Challenges of Tail Users and Items

In recent years, a lot of CF techniques have been developed [Su and Khoshgoftaar 2009], where k nearest neighbor (kNN) [Candillier et al. 2008] and matrix factorization (MF) [Koren et al. 2009] are two examples. In particular, kNN is a representative, memory-based approach while MF is a representative, model-based approach. The simple form

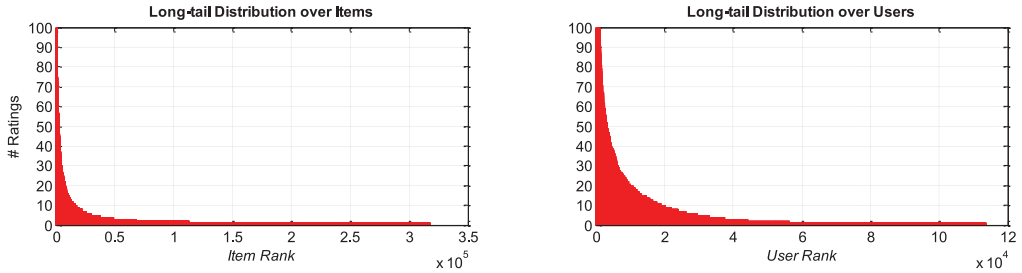

Fig. 1. Items (left) and users (right) are ranked by the number of their ratings (truncated from 0 to 100) on Rich Epinions Dataset; they are both clearly distributed with short heads and long tails.

of user-based kNN can be given by:

$$\hat{Y}_{i,j} = \frac{\sum_{n \in \Gamma_i} w_{i,n} Y_{n,j}}{\sum_{n \in \Gamma_i} w_{i,n}} \tag{1}$$

where $\hat{Y}_{i,j}$ is the predictive rating of user i on item j; $Y_{n,j}$ is the observation on j from a neighbor of i; $\mathbf{\Gamma}_i$ denotes the neighbor set of user i; and $w_{i,n}$ is the weight between user i and user n. Typically, $w_{i,n}$ can be computed by Pearson correlation, cosine, or Jaccard similarity [Candillier et al. 2008].

A probabilistic matrix factorization (PMF) [Salakhutdinov and Mnih 2008b] is a typical model to illustrate the MF approach from the probabilistic view. Given a data matrix $\boldsymbol{Y} \in \mathbb{R}^{N \times M}$ with the index of each observed choice $(i, j) \in \boldsymbol{O}$ on N users and M items, we can obtain the following distributions with the Gaussian latent factors $\boldsymbol{U}_i$ of users and $\boldsymbol{V}_j$ of items:

$$P(\boldsymbol{U}_i) = N\big(\boldsymbol{U}_i | \boldsymbol{0}, \sigma_U^2 \boldsymbol{I}\big) \qquad P\big(\boldsymbol{V}_j\big) = N\big(\boldsymbol{V}_j | \boldsymbol{0}, \sigma_V^2 \boldsymbol{I}\big) \tag{2}$$

$$P(Y_{ij} | \boldsymbol{U}_i, \boldsymbol{V}_j) = N\big(Y_{ij} | \boldsymbol{U}_i^{\mathrm{T}} \boldsymbol{V}_j, \sigma^2\big) \tag{3}$$

$$P\left(\boldsymbol{U}, \boldsymbol{V} | \boldsymbol{Y}\right) \propto P\left(\boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V}\right) = \prod_{ij \in \boldsymbol{O}} P(Y_{ij} | \boldsymbol{U}_i, \boldsymbol{V}_j) \prod_i P(\boldsymbol{U}_i) \prod_j P(\boldsymbol{V}_j) \tag{4}$$

where $\boldsymbol{U} = [\boldsymbol{U}_1, \ldots, \boldsymbol{U}_N]$ is the user factor matrix; $\boldsymbol{V} = [\boldsymbol{V}_1, \ldots, \boldsymbol{V}_M]$ is the item factor matrix; and $\sigma_U^2, \sigma_V^2, \sigma^2$ are the variance parameters of the Gaussian distributions. We learn the user factors and the item factors through a maximum *a posteriori* (MAP) estimate. According to the Bayesian theorem, we have the posterior $P(\boldsymbol{U}, \boldsymbol{V} | \boldsymbol{Y}) \propto P(\boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{V})$ given in Equation (4). The following objective function can then be obtained by minimizing the negative log-posterior. Without loss of generality, we easily obtain the classic objective of an MF model [Koren et al. 2009; Salakhutdinov and Mnih 2008b], when we set $\sigma^2 = 1$ and denote $\lambda = \sigma_U^{-2} = \sigma_V^{-2}$ as the regularization parameter:

$$\boldsymbol{J} = \underset{\boldsymbol{U}, \boldsymbol{V}}{argmin} \frac{1}{2} \left[\sum_{ij \in \boldsymbol{O}} \left(Y_{ij} - \boldsymbol{U}_i^{\mathrm{T}} \boldsymbol{V}_j\right)^2 + \lambda \left(\left\| \boldsymbol{U}_F^2 \right\| + \left\| \boldsymbol{V}_F^2 \right\| \right) \right] \tag{5}$$

We can easily write the partial derivative $\partial \boldsymbol{J} / \partial \boldsymbol{V}_j$ w.r.t. each $\boldsymbol{V}_j$. The optimization w.r.t. $\boldsymbol{V}_j$ is convex when $\boldsymbol{U}$ is fixed. A close-form update equation for $\boldsymbol{V}_j$ can be obtained by

setting $\partial \boldsymbol{J}/\partial \boldsymbol{V}_j$ to zero [Salakhutdinov and Mnih 2008b]:

$$\boldsymbol{V}_j \leftarrow \left(\lambda \boldsymbol{I} + \sum_{i \in \boldsymbol{O}_j} \boldsymbol{U}_i \boldsymbol{U}_i^{\mathrm{T}}\right)^{-1} \sum_{i \in \boldsymbol{O}_j} Y_{ij} \boldsymbol{U}_i \tag{6}$$

where $\boldsymbol{O}_j$ indexes those users who have chosen item j. Similarly, the optimization w.r.t. $\boldsymbol{U}_i$ is convex when $\boldsymbol{V}$ is fixed, and thus, we can obtain:

$$\boldsymbol{U}_i \leftarrow \left(\lambda \boldsymbol{I} + \sum_{j \in \boldsymbol{O}_i} \boldsymbol{V}_j \boldsymbol{V}_j^{\mathrm{T}}\right)^{-1} \sum_{j \in \boldsymbol{O}_i} Y_{ij} \boldsymbol{V}_j \tag{7}$$

where $\boldsymbol{O}_i$ indexes the items chosen by user i.

So far, we have briefly reviewed the kNN and MF models. Due to the skewness of the distributions of the users and items (cf. Figure 1), the data pulled from long-tail users and items is much sparser than that of short-head users and items. As a result, the kNN and MF models are more vulnerable to the following challenges than long-tail users and items.

Popularity Bias. Given any two users, their choices tend to overlap more with popular items but less so with long-tail items. Hence, the neighbor set in kNN is largely constructed from popular items. Furthermore, note that the data is very sparse in the tail, i.e., each tail item is chosen by very few users. As a result, a user often cannot find any feedback on those long-tail items from neighbors, so a prediction is unavailable (cf. Equation (1)), which creates a situation where those items will never be recommended. As for popular items, they tend to have sufficient feedback, so kNN can more easily make predictions and recommend them. Although MF does not directly depend on neighbor data, it still suffers from the tail's data-sparsity challenge. From Equation (6) and Equation (7), we find that the estimates of item factors and user-factors are largely determined by the amount of observed data w.r.t. item j and user i. For items and users in the tail, the amount of observable data is small, so the quality of the estimates for them are naturally poorer than with short-head items and users. Accordingly, the predictions for short-head items and users are much more accurate than those for long-tail items and users.

Cold Start. The long-tail distribution implies a number of users are cold start in nature. Cold-start users usually have provided little feedback on some popular items, or sometimes no feedback at all. Given a cold-start user, kNN does not have sufficient data to find suitable neighbors. As a result, the prediction results for long-tail items or users is poor. From Equation (7), we find that the factors, $\boldsymbol{U}_i$, differentiate user preferences according to the feedback they have provided on different items. However, long-tail users have provided very little feedback on popular items so the learned factors of these users tend to be similar; for this reason, they are not able to clearly represent personal preferences. In extreme cases, we find that neither kNN nor MF can work, cf. Equations (1) and (7), when we have only cold-start users without any feedback data.

Shilling Attack. As mentioned, a shilling attack refers to a group of spam users intentionally providing fake feedback, e.g., much higher or lower ratings than a true rating to bias the ratings and the recommendations for them. Intuitively, short-head items are well known, and users either actively or passively learn information about them from many sources. For this reason, short-head items are less affected by shilling attack. In comparison, information on unpopular items is limited, and they are largely known by recommendations. Thus, these items can suffer more easily from shilling

attack. Moreover, the number of ratings on head items is much more than on tail items; as a result, it is much easier to attack tail items by imputing a few fake ratings. For example, given a head item that has 1,000 ratings and a tail item that has 5 ratings, and where both have an average score of 4, if a shilling attack gives five low ratings with a score of 1 on both items, the average rating of this head item is still close to 4, but the average rating of the tail item is reduced to 2.5.

Since kNN depends on the neighbors' feedback to construct a prediction, it can suffer greatly from a shilling attack on tail items due to the large proportion of fake feedback. As for the MF method, each item factor vector, $\boldsymbol{V}_j$, determines how this item would be preferred by users. If a shilling attack is conducted on a tail item j, $\boldsymbol{V}_j$ is learned largely only from fake feedback (cf. Equation (6)). As a result, the prediction for a user's preference for item j is biased by the fake $\boldsymbol{V}_j$.

1.2. Our Proposal

Users' choices for popular items are largely influenced by others. For example, seeing a new movie with friends may not really reflect a particular user's subjective preference. On the other hand, choices for long-tail items can better reflect a user's taste since they are rarely due to the influence of others. To tackle the popularity-bias challenge, we need to construct a model that can emphasize the choices of long-tail items in order to learn users' special preferences. However, only emphasizing long-tail items is not enough because, as mentioned, long-tail items suffer more easily from shilling attack. Therefore, we also need to construct a model that can weigh the credibility of each piece of feedback. Moreover, an efficient way to deal with the cold-start challenge is to borrow information from other relevant users. We argue that high-quality, relevant users should be reputable and, thus, trusted. In summary, we need to design an approach that jointly models both the objective to emphasize the *specialty* of choices and the objective to assess the *credibility* of the feedback for each choice.

Heteroscedastic Matrix Factorization. In Equation (3), the variance parameter, σ^2, does not vary with different observations $\{Y_{ij}\}$, which is so-called "homoscedasticity." Now, if we model each observation Y_{ij} with different variance, σ_{ij}^2, as demonstrated in Equation (8), then these observations are assumed to be "heteroscedastic." By minimizing the negative log-form of Equation (4), we immediately obtain the following objective (Equation (9)) where the weighted squared loss $w_{ij}(Y_{ij} - \boldsymbol{U}_i^{\mathrm{T}}\boldsymbol{V}_j)^2$ is the log-form of Equation (8) and $w_{ij} = \sigma_{ij}^{-2}$ serves as the weight to penalize the loss of fitting Y_{ij}. As a result, we call this variant MF model heteroscedastic MF (HMF). From a probabilistic view, the variance parameter σ_{ij}^2 controls the confidence level [Hu et al. 2014]. Specifically, a smaller σ_{ij}^2 implies higher confidence and less uncertainty of the observation Y_{ij}, i.e., a large w_{ij} is applied to more tightly fitting Y_{ij}.

$$P(Y_{ij}|\boldsymbol{U}_i, \boldsymbol{V}_j) = N\big(Y_{ij}|\boldsymbol{U}_i^{\mathrm{T}}\boldsymbol{V}_j, \sigma_{ij}^2\big) \tag{8}$$

$$J = \underset{\boldsymbol{U},\boldsymbol{V}}{argmin}\ \frac{1}{2}\left[\underbrace{\sum\nolimits_{ij} w_{ij}\big(Y_{ij} - \boldsymbol{U}_i^{\mathrm{T}}\boldsymbol{V}_j\big)^2}_{weighted\ loss} + \lambda\big(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2\big)\right] \tag{9}$$

Heteroscedastic Modeling. In order to emphasize the *specialty* of users' choices with long-tail items, we differentiate each user choice in terms of heteroscedastic modeling. Specifically, we model the variance parameter σ_{ij}^2 by a variance function $f^S(\cdot)$ to score

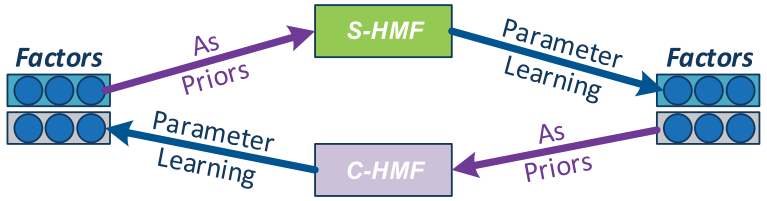

Fig. 2. A recurrent mutual regularization process couples S-HMF and C-HMF using the user and item-factors learned from one another as the empirical priors to couple the objectives *specialty* and *credibility*.

the *specialty* of this choice. As a result, we obtain specialty-specific heteroscedastic MF (S-HMF), which more tightly fits the users' choices for long-tail items.

On the other hand, we need to model the *credibility* of users' feedback. Technically, the parameters should be estimated by tightly fitting the more credible feedback while loosely fitting the less credible feedback. Hence, we can construct another HMF model using Equation (8), where the variance parameter σ_{ij}^2 is modeled by a variance function $f^C(\cdot)$, which scores the *credibility* of each review. We call this type of HMF a *credibility-specific HMF (C-HMF)*.

Coupling Objectives using Empirical Priors. So far, we have presented S-HMF and C-HMF, which correspond to two independent objectives for optimization. However, we need to consider both objectives w.r.t. *specialty* and *credibility jointly* when learning user preference, as stated previously. From the view of Bayesian modeling, MAP trades off the estimation of parameters between prior and likelihood. That is, the estimates are very close to the given prior when little data is available. In PMF, the priors of user factors and item factors are modeled by uninformative priors, i.e., zero-means as illustrated by Equation (2). If we provide some informative priors, $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$, pertaining to the user factors and the items factors as Equation (10), then we obtain the objective function illustrated by Equation (11). In this objective, we find that the estimates of $\boldsymbol{U}_i$ and $\boldsymbol{V}_j$ are regularized by the given priors $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$. As a result, the estimates of user factors and items factors tend to shrink towards the given informative priors.

$$P(\boldsymbol{U}_i) = N\big(\boldsymbol{U}_i|\boldsymbol{\mu}_U, \sigma_U^2\boldsymbol{I}\big) \qquad P(\boldsymbol{V}_j) = N(\boldsymbol{V}_j|\boldsymbol{\mu}_V, \sigma_V^2\boldsymbol{I}) \tag{10}$$

$$J = \underset{\boldsymbol{U},\boldsymbol{V}}{argmin}\,\frac{1}{2}\left[\underbrace{\textstyle\sum_{ij} w_{ij}\big(Y_{ij} - \boldsymbol{U}_i^{\mathrm{T}}\boldsymbol{V}_j\big)^2}_{weighted\ loss} + \underbrace{\lambda_U \textstyle\sum_i \|\boldsymbol{U}_i - \boldsymbol{\mu}_i\|_2^2 + \lambda_V \sum_j \|\boldsymbol{V}_j - \boldsymbol{\mu}_j\|_2^2}_{regularization}\right] \tag{11}$$

The user factors and the item-factors learned from C-HMF target the objective of *credibility*, so they can serve as good empirical priors for modeling the user factors and the item factors of S-HMF. Symmetrically, the user factors and the item factors learned from S-HMF target the objective of *specialty*, which contains information on the users' intrinsic preferences. As a result, the user factors and the item factors learned from S-HMF are good empirical priors for modeling the user factors and the item factors of C-HMF. Thus, as shown in Figure 2, it forms a mutual regularization process that couples S-HMF and C-HMF using the empirical priors learned from each other recurrently. Therefore, we call this coupled model for multi-objective optimization the *recurrent mutual regularization model (RMRM)*.

Moreover, we need to borrow information from other users' preferences to deal with cold-start users. Since user factors represent the features of user preference, we can construct the prior using user factors from a group of relevant users. Specifically, we

design a sophisticated way to combine these Gaussian-distributed user factor vectors, namely, by using the Product of Gaussian Experts (PoGE) [Hinton 2002; Williams and Agakov 2002]. In fact, such a PoGE-prior can be regarded as playing the role of social regularization [Ma et al. 2011]. This will be discussed further in the following sections.

1.3. Contributions

In this article, we address the challenges of recommendations for long-tail items and users. The main contributions of our work are summarized as follows:

—We present the potential requirements to improve recommendations for users and items in the tail of distributions, which will significantly improve business profits and improve a user's experience.
—We show the vulnerabilities of current approaches for tail items and users as they pertain to the challenges of popularity bias, cold start, and shilling attack. As a result, we establish a pair of coupled objectives to jointly emphasize the *specialty* of choices and assess the *credibility* of feedback.
—We design a recurrent mutual regularization process to couple the objectives modeled by S-HMF and C-HMF. To implement the recurrent mutual regularization process for RMRM, we design a scalable algorithm based on the variational Bayesian method to efficiently learn its parameters.
—We conduct empirical evaluations on two real-world data sets. Based on various metrics, the overall results prove that our approach significantly outperforms the compared methods.
—Although we focus mainly on recommendation problems in this article, the proposed approach could potentially be applied to many other areas. RMRM provides a general framework to couple multiple objectives and to learn the comprehensive latent features regularized by empirical priors.

2. RELATED WORK

Basically, our work aims to improve the accuracy of recommendations for users and items in the tail by considering both *specialty* and *credibility*. Note that some recent research targets long-tail recommendations from different perspectives to improve certain other metrics, such as *diversity* and *serendipity* [Herlocker et al. 2004; Vargas and Castells 2011; Yin et al. 2012]. The focus of this research and its goals is quite different from our own work, so our objectives should not be confused with theirs.

2.1. Technologies in Recommender Systems

Memory-based approaches, such as kNN, are the origins of CF systems [Su and Khoshgoftaar 2009], and they have been applied successfully in real-world commercial systems [Sarwar et al. 2001]. In general, kNN can be subdivided into two categories: user-based nearest neighbor; and item-based nearest neighbor [Su and Khoshgoftaar 2009]. However, these approaches are not suitable when the data is sparse. They are not able to deal with the challenges of long-tail items and users.

With the rapid development of machine learning, model-based approaches have become more and more popular in recent years. Here, matrix factorization (MF) models have gained dominance in the field of recommendation, as they have shown their superiority to neighborhood-based techniques, ultimately winning the Netflix Prize competition [Koren et al. 2009]. The basic idea of MF methods is to fit the user-item rating matrix using low-rank approximations, and then to use these results for prediction. Many MF methods have been proposed, including probabilistic MF (PMF) [Salakhutdinov and Mnih 2008b] and maximum-margin MF (MMMF) [Srebro et al. 2005]. To achieve better performance, they need to carefully tune the hyperparameters

for MF. Hence, researchers have proposed Bayesian PMF (BPMF) [Salakhutdinov and Mnih 2008a] and variational Bayesian MF (VBMF) [Lim and Teh 2007; Shan and Banerjee 2010] to learn the hyperparameters.

Apart from the MF approach, some other models have also achieved success in this field. For example, in the literature, choice modeling [Train 2003] is strongly related to the recommendation problem, and Hu et al. [2014] proposed a latent-feature-based Bayesian heteroscedastic choice model (BHCM) to represent heterogeneities between users and items. Additionally, with the prevalence of deep learning techniques [Bengio et al. 2013], restricted Boltzmann machines (RBM) have also been applied in RSs [Georgiev and Nakov 2013]. This was reported to have achieved comparable performance to MF in the Netflix Prize competition [Salakhutdinov et al. 2007].

Most current recommender systems are built on explicit feedback, e.g., ratings, to differentiate users' preferences. However, explicit feedback is not always available in the real world while implicit feedback, e.g., click logs, can be obtained more easily. Implicit feedback is often represented by binary values, that is, 1 for observed choices and 0 for others [Hu et al. 2014; Pan et al. 2008]. These unobserved choices have zero values, but they do not mean true negative instances. Therefore, a strategy that is often used to assign a larger confidence level to the observed choices to represent the high certainty of users' explicit likes while a much smaller confidence level is assigned to unobserved choices to represent the small certainty of dislike [Hu et al. 2014; Hu et al. 2008; Pan et al. 2008]. For instance, Bayesian personalized ranking (BPR) [Rendle et al. 2009] assumes that users show a stronger like for their chosen items than for unobserved ones. Hence, the preference ordering relation can be constructed for each pair of items. As a result, BPR learns the utility of choosing an item from the ordering relationships.

2.2. Dealing with Data Sparsity in the Long Tail

The long tail was popularized by Anderson in 2004, who reported that Amazon, Apple, and Yahoo! apply this strategy to realize significant profits when selling items in the tail [Anderson 2006]. However, there is sparse data for long-tail items and users, which greatly decreases the quality of recommendations. Park and Tuzhilin [2008] observed this difficulty when recommending long-tail items with very few ratings. Thus, they proposed to split the whole item set into head and tail parts, and then to group the tail items into clusters. As a result, the clusters of the tail items are treated as virtual items, which have relatively more ratings than just a single item. Levy and Bosteels [2010] then studied music recommendations in the long tail using a conventional, item-based CF approach. However, these methods cannot work well when the data is too sparse to find similar items.

To tackle the data-sparsity challenge, one often needs to incorporate additional information. It is expected that the additional side information about the users and the items would be beneficial if it were to be incorporated into a model. Thus, Bayesian matrix factorization with side information (BMFSI) [Porteous et al. 2010] is extended from BPMF. It incorporates side information via linear regression. That is, the data is modeled as a combination of MF terms with user and item latent factors and regression against the side information of users and items. Further, the regression-based latent factor model (RLFM) [Agarwal and Chen 2009] uses a different strategy to incorporate side information. It assumes that the user latent factor matrix is generated from the provided features (i.e., side information) of users via regression while the item latent factor matrix is generated from the provided features of items via regression. Following this, both the user and the item latent factor matrices are used as MF. However, side information is not always available due to privacy and security. In addition, these

methods do not consider the credibility of the data and the side information, so they can easily suffer from shilling attack for long-tail items.

2.3. Trust-Aware Recommender Systems

Spam data can be found everywhere on the Internet, e.g., e-commerce and social networking sites. Hence, trust and reputation systems (TRS) [Jøsang and Ismail 2002; Jøsang et al. 2008] have been designed specifically to foster trusted behavior in these domains. In recent years, with increasing attention placed on RS, researchers in the TRS area have proposed to incorporate trust into RS [Jøsang et al. 2013], where a trust score between two users serves as the weight to conduct the conventional neighbor-based method. However, the neighbor-based method has its limitations when the data is very sparse.

People in the RS area are also aware of the importance of trust, and they have designed many MF-based methods. SoRec [Ma et al. 2008] incorporates trust networks into RS where joint factorization is conducted over two matrices: *user-item* ratings and *user-user* trust relationships. In recent years, researchers have utilized social relations, i.e., trust relations, to perform regularization for users. SocialMF [Jamali and Ester 2010] and SoReg [Ma et al. 2011] are two representative models to regularize the user factor vector of a target user via the user factor vectors of their trusters. Such regularization plays the role of borrowing the preferences of the trusters to deal with the cold-start issue.

In this article, we also incorporate trust information into our framework. On the one hand, we employ Bayesian reputation modeling to assess the reputation of users in order to weight the credibility of their feedback. On the other hand, we place a PoGE-prior that has been constructed via high-reputation trusters in order to better regularize user factors learning as SocialMF and SoReg. As a result, our model has the advantages of both, which enables it to better deal with the challenges of cold start and shilling attack.

3. PRELIMINARIES

3.1. Notations

Before commencing a detailed discussion, we summarize the frequently used notations used in this article and their meanings in Table I to simplify the presentation in the rest of the article.

3.2. Reputation Modeling

One of the key components in our approach is modeling the reputation of users and the credibility of their feedback. Intuitively, the reviews from high-reputation users tend to be more trusted and helpful to other users, and thus, they more credibly discuss the real features of items. In other words, the reputation of a user is highly relevant to the credibility of her feedback. Bayesian reputation systems [Jøsang and Quattrociocchi 2009] have been proposed to model reputation from a probabilistic perspective, so this can be integrated easily into our framework. In particular, in this work, we employ the beta reputation model [Jøsang and Ismail 2002] to obtain the helpfulness scores for user reviews.

3.2.1. Beta Reputation Model. Let $\boldsymbol{e} \stackrel{\text{def}}{=} \{r, s\}$ denote the evidence that contains r positive feedback and s negative feedback w.r.t. a target entity. The probability of evidence $\boldsymbol{e}$ can be described by a group of Bernoulli events, which follows a binomial distribution:

$$P(\boldsymbol{e}|p) = \binom{r+s}{r} p^r (1-p)^s \tag{12}$$

Table I. Summary of Frequently Used Notations in This Article

Symbol	Description
i	$i \in \{1, \ldots, N\}$ is used to index a user
j	$j \in \{1, \ldots, M\}$ is used to index an item
$\boldsymbol{U}_i$	$\boldsymbol{U} = [\boldsymbol{U}_1, \ldots, \boldsymbol{U}_N]$ is the user factor matrix, where $\boldsymbol{U}_i$ is the user factor vector of user i
$\boldsymbol{V}_j$	$\boldsymbol{V} = [\boldsymbol{V}_1, \ldots, \boldsymbol{V}_M]$ is the item factor matrix, where $\boldsymbol{V}_j$ is the item factor vector of item j
Y_{ij}	$\boldsymbol{Y}$ is the data matrix, and $Y_{ij} \in \boldsymbol{Y}$ is an entry with the index (i, j)
$\boldsymbol{O}_i$	$\boldsymbol{O}$ is the index set of all modeled data points,$\boldsymbol{O}_i$ is the index set of data w.r.t. user i
$\boldsymbol{O}_j$	$\boldsymbol{O}_j$ is the index set of data w.r.t. item j
σ_{ij}^2	σ_{ij}^2 is the variance parameter of observation $\boldsymbol{Y}_{ij}$
$\boldsymbol{\mu}_i$	$\boldsymbol{\mu}_i$ is the empirical prior placed on the user factors $\boldsymbol{U}_i$
$\boldsymbol{\mu}_j$	$\boldsymbol{\mu}_j$ is the empirical prior placed on the item factors $\boldsymbol{V}_j$
w_{ij}	$\boldsymbol{W}$ is the weight matrix, and $w_{ij} \in \boldsymbol{W}$ is the weight to scale the loss of fitting Y_{ij}
φ_i	φ_i is the reputation score of user i
ω_{ij}	ω_{ij} is the credibility score on a user review
η_{ij}	η_{ij} is the specialty score on a user choice
$\mathcal{N}_i^K$	$\mathcal{N}_i^K$ is the top-K neighbors of user i
$\mathcal{N}^R$	$\mathcal{N}^R$ denotes the top-R high-reputation experts in the system
^{S}	Superscript to indicate S-HMF-related model parameters
^{C}	Superscript to indicate C-HMF-related model parameters
$\|\boldsymbol{x}\|_2$	The 2-norm of a vector $\boldsymbol{x}$
$diag(\boldsymbol{x})$	Generate a diagonal matrix using a vector $\boldsymbol{x}$
$PoGE$	Product of Gaussian experts
$.*$	Element-wise product (MATLAB-style)
$.^2$	Element-wise square (MATLAB-style)
$.^{-1}$	Element-wise inverse (MATLAB-style)

Then, we can obtain the following definition of a reputation function given the beta-prior with the probability density function (pdf) defined by a gamma function $\Gamma(\cdot)$:

$$Beta\,(p|\alpha, \beta) = \frac{\Gamma\,(\alpha + \beta)}{\Gamma\,(\alpha)\,\Gamma\,(\beta)} p^{\alpha-1}\,(1-p)^{\beta-1}$$

Definition 1 (*Reputation Function*). [Jøsang and Ismail 2002]:

$$\varphi(\boldsymbol{e}) \stackrel{\text{def}}{=} \int P\,(\boldsymbol{e}|p)\,Beta\,(p|\alpha, \beta)\,dp = Beta\,(r+\alpha, s+\beta)\,.$$

Obviously, the reputation function is defined as the posterior of the beta distribution, where the hyperparameters α and β can be thought of as a certain amount of pseudo-positive and -negative feedback—in practice, this is often set $\alpha = \beta = 1$. Next, we can obtain the expectation of this reputation function, i.e., the mean of $Beta\,(r+\alpha, s+\beta)$:

$$\mathcal{R}(\boldsymbol{e}) \stackrel{\text{def}}{=} \mathbb{E}\,[\varphi\,(\boldsymbol{e})] = \frac{r+\alpha}{r+\alpha+s+\beta} \tag{13}$$

We find that $\mathcal{R}\,(\boldsymbol{e})$ is bounded within (0, 1), and that it approaches the upper bound 1 only if the user has a large amount of positive feedback. Clearly, we can employ such a beta reputation model to assess the reputation of a user by the score $\mathcal{R}(\boldsymbol{e})$ in RSs.

3.2.2. Data for Modeling Reputation. For most online shopping and review websites, a user's review of an item consists of a rating and a free message. In order to find helpful reviews and to display them on the first page, some of the most well-known

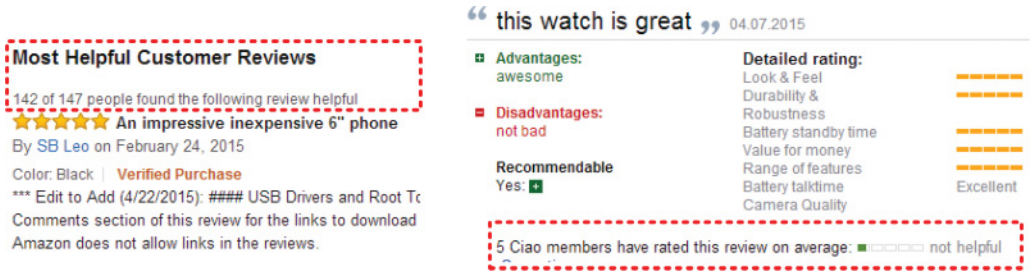

Fig. 3. The screen snapshots of reviews from Amazon.com (left) and Ciao.com (right), where the red, dotted boxes show the helpfulness score.

websites, e.g., Amazon.com,[1] Epinions.com,[2] Ciao.com,[3] have designed a scoring system to evaluate the helpfulness of each review.

Figure 3 demonstrates two screen snapshots of reviews found on Amazon.com and Ciao.com. These websites have integrated several algorithms to score the helpfulness of each review in terms of other users' feedback or experts' judgments of this review. As a result, each review is generally associated with a multilevel helpfulness score, from *Not Helpful* to *Most Helpful*, as shown in Figure 3. Obviously, we can employ the beta reputation model to assess the reputation of each user: if a user gives a lot of reviews that mostly receive high helpfulness scores, then this user tends to be a high-reputation user. On the contrary, the reputation score will be low if a spam user gives a lot of fake reviews. The mathematical representation of the reputation score will be discussed along with our model in the following subsections.

3.3. Explicit and Implicit Rating

Rating data is typical feedback that represents the preferences of users. Typically, rating data can be divided into two categories: explicit and implicit.

Explicit Rating. The multilevel rating scores, e.g., five-star ratings, can explicitly differentiate user preferences, so they are typical explicit rating data. Therefore, we only model observed ratings, while the remaining entries are treated as missing. From the HMF view, we have no information on these missing entries to tell us whether users have liked the items or not, so we assign positive confidence, $c_{ij} > 0$, to the observed ratings and zeros to the remaining ones.

$$w_{ij} = \begin{cases} c_{ij} & (i,j) \text{ indexes an observation} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

If we set all $c_{ij} = 1$, then we obtain a binary-weight matrix $\boldsymbol{w}$ [Acar et al. 2010; Srebro and Jaakkola 2003]. Using this $\boldsymbol{w}$ in Equation (14), we can immediately obtain the traditional unweighted MF objective, as shown in Equation (5), from the objective of HMF. In this case, the index set $\boldsymbol{O} = \{(i,j)|w_{ij} > 0\}$ only consists of observed entries.

Implicit Rating. In the real world, explicit ratings are not always provided by users, but implicit rating data, such as purchase records and number of clicks, can be obtained more easily. This implicit rating data is usually modeled as a unary preference because the blank entries do not necessarily indicate user dislike, but, instead, are a result of the users' lack of awareness [Herlocker et al. 2004]. Hence, we can assign a higher

[1]http://www.amazon.com.
[2]http://www.epinions.com.
[3]http://www.ciao.com.

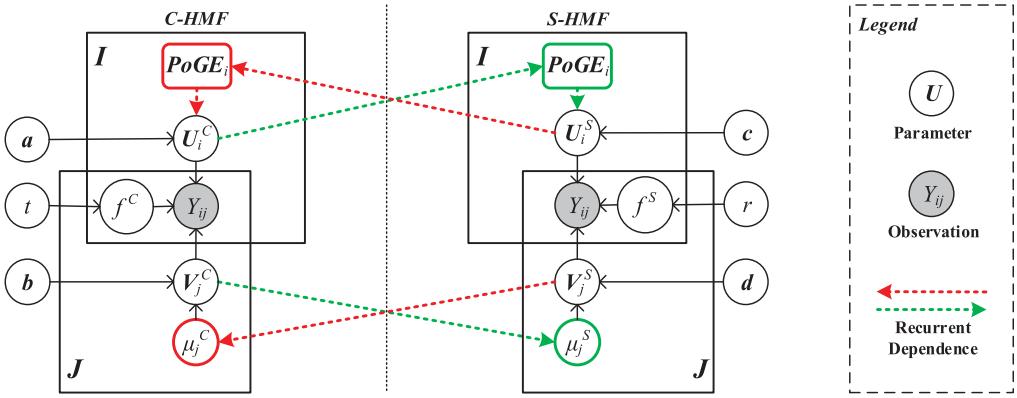

Fig. 4. The graphical representation of the RMRM framework, where S-HMF and C-HMF are recurrently regularized by the empirical priors, induced from one another.

confidence level to observed entries and a much lower confidence level to blank entries [Hu et al. 2014; Hu et al. 2008]. Recall that the confidence level is associated with the variance parameter, i.e., the inverse of weight (cf. Equation (14)), when a Gaussian distribution is assumed. As a result, the weighting strategy [Hu et al. 2008; Pan et al. 2008] of implicit ratings is often established as follows:

$$w_{ij} = \begin{cases} c_{ij} & (i,j) \text{indexes an observation} \\ \epsilon & \text{otherwise} \end{cases} \tag{15}$$

where ϵ is a small constant to denote the low confidence representing users' likes or dislikes for blank entries while $c_{ij} > \epsilon$ denotes relatively higher confidence representing users' likes of observed entries. In this case, we need to model both likes and dislikes so the index set $\boldsymbol{O}$ consists of all entries of the data matrix.

4. MODEL AND LEARNING

4.1. Overview of RMRM

To implement more reliable recommendations for tail users and tail items, we propose to model two coupled objectives for joint optimization, namely, the *specialty* of user choices and the *credibility* of user feedback. To achieve this goal, we design a recurrent mutual regularization model (RMRM) to couple these two objectives together.

4.1.1. The Framework. As illustrated in Figure 4, the objective of *specialty* is modeled by S-HMF (the right model shown in Figure 4) while the objective of *credibility* is modeled by C-HMF (the left model shown in Figure 4). RMRM couples these two objective models in terms of the empirical priors induced from one another.

The C-HMF focuses on modeling the credibility of each user review. This is implemented using two means. First, C-HMF assigns different levels of confidence, i.e., variance, for each observation, Y_{ij}, where the variance is modeled by a variance function $f^C(\cdot)$, which is devised based on the Bayesian reputation model as presented in Section 3.2. As a result, the estimation of the item-factors is more dependent on credible feedback. Second, a PoGE-prior is imposed on the user factors of each user, which plays the role of regularizing user behavior in terms of relevant, high-reputation experts. Here, such a PoGE-prior can regularize both the preference learning of cold-start users and the behavior of spam users. Therefore, the item factor vectors $\{\boldsymbol{V}_j^C\}$ learned from C-HMF represents more authentic features of items than those learned from classic MF

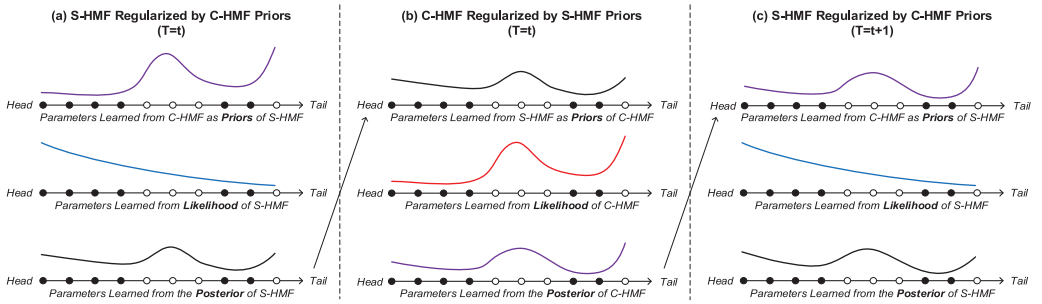

Fig. 5. The geometric illustration of the recurrent mutual regularization process, where the estimates of S-HMF and C-HMF are recurrently regularized by the empirical priors induced from one another.

models. At the same time, the user factor vectors $\{\boldsymbol{U}_i^C\}$ of tail users contain knowledge from relevant experts.

The S-HMF focuses on emphasizing the specialty of choices. The choices of tail items are much less influenced by others, thus they better reflect personal preferences. In S-HMF, the variance function $f^S(\cdot)$ assigns greater confidence to the choice of items in a deeper tail. As a result, S-HMF tends to fit the observations of tail items more tightly than those of head items. Therefore, the user factor vectors $\{\boldsymbol{U}_i^S\}$ that are estimated from C-HMF can better reflect users' personal preferences than those learned using classic MF methods.

As discussed in previous sections, long-tail items and users with little data are more easily affected by shilling attack and cold-start issues, which leads to unreliable estimates $\{\boldsymbol{V}_j^S\}$ and $\{\boldsymbol{U}_i^S\}$ learned from S-HMF. According to Bayesian probabilistic modeling, a prior plays an important role when there is limited data. Therefore, $\{\boldsymbol{V}_j^C\}$ and $\{\boldsymbol{U}_i^C\}$ learned from the credibility-oriented objective model, i.e., C-HMF, are good empirical priors to regularize S-HMF to relieve both shilling attack and cold start. In turn, $\{\boldsymbol{U}_i^S\}$ learned from S-HMF are refined user features so they can serve as the empirical priors for C-HMF in order to deal with popularity bias. Therefore, we designed a RMRM framework that consists of the recurrent dependencies between C-HMF and S-HMF to handle these challenges.

4.1.2. Geometric Illustration. To give an intuitive understanding of the working mechanism of RMRM, we provide the geometric illustration depicted in Figure 5. Here, we demonstrate the data fitting process of RMRM from the perspective of a given user, as well as a similar process that can be conducted from an item perspective. The axes arrange items (denoted as small circles) according to their popularity. More specifically, those items where the user provides credible feedback are marked with solid circles whereas hollow circles denote items receiving less credible feedback. The colored lines indicate fitting curves; the closer the curve is to a circle indicates the tighter the parameters to fit the choice of the corresponding item.

Figure 5(a) depicts S-HMF, which is regularized by the parameters learned from C-HMF. Given a user i, the top fitting curve of Figure 5(a) reflects the parameters learned from C-HMF, which tend to tightly fit user i's choices with credible feedback whereas they are loosely fit with others without credible feedback. The middle fitting curve of Figure 5(a) reflects the parameters learned directly by maximizing the heteroscedastic likelihood of the choices of user i, i.e., minimizing the weighted loss of fitting user i's choices (cf. Equation (14)). When the parameters learned from C-HMF are employed as empirical priors for S-HMF, we obtain a regularized S-HMF model, as depicted in the right part of RMRM. The parameters can be estimated by maximizing the posterior, i.e.,

minimizing the objective given in Equation (14), where the regularization term brings estimates closer to the given priors. As a result, the bottom fitting curve of Figure 5 represents the regularization results of the parameters, which more aggressively fit those choices in the tail with a high degree of credibility.

In turn, the parameters learned from this regularized S-HMF serve as the empirical priors to regularize C-HMF, as shown in the top curve of Figure 5(b). The parameters learned directly by maximizing the heteroscedastic likelihood of C-HMF tend to fit more tightly with the choices with credible feedback, as shown by the middle curve of Figure 5(b). When the empirical priors are imposed for regularization, the parameters learned from the posterior of C-HMF contain the information of the specialty of choices from the priors. Therefore, the bottom fitting curve shown in Figure 5(b) tends to fit more tightly the choices of tail items than the one produced by the maximum heteroscedastic likelihood estimation.

Now, let us move to the next iteration, as shown in Figure 5(c). As in the previous iteration, the parameters learned from the regularized C-HMF serve as the empirical priors to regularize S-HMF. As a result, the coupled recurrent regularizing process of RMRM converges the parameters to the region that represents the specialty of user choices and, simultaneously, enhances its credibility.

4.2. Learning Regularized C-HMF Model

Given observations $\{Y_{ij}|(i,j) \in \boldsymbol{O}\}$, we can obtain the probabilistic model according to the graphical representation of C-HMF, as shown in the left part of Figure 4:

$$P(\boldsymbol{U}_i^C) = PoGE(\boldsymbol{U}_i^C|\{\boldsymbol{\mu}_n^C, \varphi_n\}_{n\in T_i}) \tag{16}$$

$$P(\boldsymbol{V}_j^C) = N(\boldsymbol{V}_j^C|\boldsymbol{\mu}_j^C, diag[\boldsymbol{b}]) \tag{17}$$

$$P(Y_{ij}|\boldsymbol{U}_i^C, \boldsymbol{V}_j^C) = N(Y_{ij}|\boldsymbol{U}_i^{C\mathrm{T}}\boldsymbol{V}_j^C, t\omega_{ij}^{-1}) \tag{18}$$

where $\boldsymbol{\mu}_n^C$ and $\boldsymbol{\mu}_j^C$ are the user and the item factor vectors induced from, $\boldsymbol{\mu}_n^S$ and $\boldsymbol{\mu}_j^S$, i.e. the counterparts in S-HMF. The details of using PoGE to construct the prior on $\boldsymbol{U}_i^C$ will be discussed later in this section. $diag[\boldsymbol{b}]$ stands for diagonal variance matrices of Gaussian priors. $t\omega_{ij}^{-1}$ as a whole denotes the variance of likelihood, where ω_{ij} is a confidence score obtained by heteroscedastic modeling w.r.t. credibility of feedback and t is a scale parameter to be learned.

4.2.1. Heteroscedastic Modeling on Credibility. The key component of C-HMF is to model the credibility of feedback on the items that a user has chosen. Intuitively, users with higher reputations tend to give more credible feedback. Therefore, we can employ the reputation model presented in Section 3.2.1 to access the reputation of each user.

Reputation Modeling. As demonstrated in Section 3.2.2, each review of a particular item is associated with a helpfulness score. Typically, a five-level score set, e.g., $\boldsymbol{h} = \{Not\ Helpful, Somewhat\ Helpful, Helpful, Very\ Helpful, Most\ Helpful\}$, is often applied to measure the helpfulness of a review. Here, we extend the beta reputation model (cf. Section 3.2.1) to assess the reputation of each user. Intuitively, if a user gives a lot of reviews that mostly receive high helpfulness scores, then this user tends to be a high-reputation user.

As the helpfulness scores for user reviews are not binary feedback, i.e., positive or negative, as presented in Section 3.2.2, they cannot directly serve as evidence. However, five-level helpfulness scores are very suitable to be represented as a typical

fuzzy set [Jøsang et al. 2008]. First, we can assign values $\boldsymbol{h} = \{0, 1, 2, 3, 4\}$ to the corresponding five-level helpfulness scores. Then, the membership functions of helpful $(+)$ and unhelpful $(-)$ can be given as follows:

$$\begin{cases} \mu_+(h) = \dfrac{h+\alpha}{h_{MAX}+\alpha} \\ \mu_-(h) = 1 - \mu_+(h) \end{cases} \tag{19}$$

where h_{MAX} is the maximum score in $\boldsymbol{h}$, e.g., $h_{MAX} = 4$ in the five-level score set above, and $\alpha \geq 0$ is a smooth parameter that bounds the degree of membership in $[\alpha/(h_{MAX}+\alpha), 1]$, e.g., the *NotHelpful* score has the smallest helpfulness $\mu_+(0)$ for a review. Then, we represent the evidence $\boldsymbol{e}_i$ for user i through all their helpfulness scores $\boldsymbol{h}_i$ as follows:

$$\boldsymbol{e}_i \stackrel{\text{def}}{=} \{\langle \mu_+(h_n), \mu_-(h_n)\rangle | h_n \in \boldsymbol{h}_i\}$$

As a result, we can still use Equation (12) to denote the probability of evidence $\boldsymbol{e}_i$, where we have r positive feedback where $r = \sum \mu_+(h_n)$, and s negative feedback where $s = \sum \mu_-(h_n)$. Then, we define the reputation score of a user based on Equation (13):

Definition 2 (*Reputation Score*). Given the helpfulness scores $\boldsymbol{h}_i$ of a user i, the reputation score of this user is defined by:

$$\varphi_i = \mathcal{R}(\boldsymbol{e}_i | \boldsymbol{h}_i) \stackrel{\text{def}}{=} \frac{r+\alpha}{r+s+\alpha+\beta} \tag{20}$$

In practice, we can set $\beta > \alpha$ *a priori*. That is, we assign a relatively low score, $\varphi_i = \alpha/(\alpha+\beta) < 0.5$, to a new user without any observed helpful ratings, because spam users often create a new account when conducting an attack to avoid being tracked by the system. Obviously, φ_i arrives at the upper bound only if a user receives a lot of high helpfulness scores for their reviews. This implies that high-reputation users are also experienced users. On the contrary, φ_i becomes lower if a user always gives false reviews.

Credibility Scoring. We assign the feedback credibility for an item choice in terms of two scores: the reputation of a user (a global score), and the helpfulness of the review (a local score). Thus, we obtain the following:

$$\omega_{ij} \stackrel{\text{def}}{=} \begin{cases} \varphi_i \mu_+(h_{ij}) + \epsilon & (i,j) \text{ is an observed entry} \\ \epsilon & \text{otherwise} \end{cases} \tag{21}$$

That is, the observation is associated with a high credibility score only if a high-reputation user gives a helpful review. In particular, we set $\epsilon = 0$ for explicit rating data while ϵ is set to a small constant for implicit rating data (cf. Section 3.3).

Since a higher credibility score means a higher confidence of that item choice, the variance function of a feedback can be given by $f^C(Y_{ij}) = t\omega_{ij}^{-1}$ (recall that lower variance means higher confidence), where t is a scale parameter to learn.

4.2.2. PoGE-Prior. In particular, we use PoGE to construct the prior for each user in order to incorporate the knowledge of a set of experts indexed by $\boldsymbol{T}_i$.

$$PoGE\big(\boldsymbol{U}_i^C | \{\boldsymbol{\mu}_n^C, w_n\}_{n \in \boldsymbol{T}_i}\big) = \prod_{n \in \boldsymbol{T}_i} N\big(\boldsymbol{U}_i^C | \boldsymbol{\mu}_n^C, diag\big[\hat{w}_n^{-1}\boldsymbol{a}\big]\big) \tag{22}$$

where w_n is a weight parameter. In general, PoE (Product of Experts) [Hinton 2002] has an intractable form. Fortunately, the product of Gaussian densities has a closed

form, that is, a new Gaussian density [Williams and Agakov 2002]. Therefore, we can obtain the following Gaussian distribution from Equation (22):

$$PoGE\big(\boldsymbol{U}_i^C \big| \{\boldsymbol{\mu}_n^C, w_n\}_{n\in\boldsymbol{T}_i}\big) = N\big(\boldsymbol{U}_i^C | \hat{\boldsymbol{\mu}}_i^C, diag\big[\hat{w}_i^{-1}\boldsymbol{a}\big]\big) \tag{23}$$

$$where\ \hat{\boldsymbol{\mu}}_i^C = \frac{\sum_{n\in\boldsymbol{T}_i} w_n \boldsymbol{\mu}_n^C}{\hat{w}_i}\ with\ \hat{w}_i = \sum_{n\in\boldsymbol{T}_i} w_n$$

Obviously, the mean parameter, $\boldsymbol{\mu}_i^C$, of the PoGE distribution is a weighted average of the user factor vectors of all related experts.

In this article, we construct the related expert set as follows:

$$\boldsymbol{T}_i = i \cup \boldsymbol{\mathcal{N}}_i^K \cup \boldsymbol{\mathcal{N}}^R \cup \boldsymbol{\mu_0} \tag{24}$$

In Equation (24), i stands for the target user itself. $\boldsymbol{\mathcal{N}}_i^K$ is the top-K neighbors of user i; the neighbors could be a set of users with an explicit relationship with i, e.g., trusters [Ma et al. 2008] or followers [Yang et al. 2011]; they can also be constructed from the data [Koren 2010] if no explicit relation is available. $\boldsymbol{\mathcal{N}}^R$ denotes the top-R high-reputation experts in the system. Moreover, $\boldsymbol{\mu_0}$ is an optional expert with a zero-mean Gaussian prior to avoid overfitting. As illustrated in Equation (16), we use the reputation score φ_n (cf. Equation (20)) of a user as the weight w_n of an expert in PoGE (cf. Equation (22)). By taking the log-form of PoGE over this expert set $\boldsymbol{T}_i$, we easily obtain the following summation form:

$$\begin{aligned}
&\log PoGE\big(\boldsymbol{U}_i^C \big| \{\boldsymbol{\mu}_n^C, \varphi_n\}_{n\in\boldsymbol{T}_i}\big) \qquad (25)\\
&= \log N\big(\boldsymbol{U}_i^C \big| \boldsymbol{\mu}_i^C, diag\big[\varphi_i^{-1}\boldsymbol{a}\big]\big) + \sum_{k\in\boldsymbol{\mathcal{N}}_i^K} \log N\big(\boldsymbol{U}_i^C \big| \boldsymbol{\mu}_k^C, diag\big[\varphi_k^{-1}\boldsymbol{a}\big]\big)\\
&\quad + \sum_{r\in\boldsymbol{\mathcal{N}}^R} \log N\big(\boldsymbol{U}_i^C \big| \boldsymbol{\mu}_r^C, diag\big[\varphi_r^{-1}\boldsymbol{a}\big]\big) + \log N\big(\boldsymbol{U}_i^C \big| \boldsymbol{0}, diag\big[\varphi_0^{-1}\boldsymbol{a}\big]\big)\\
&= \underbrace{\varphi_i \big\| \boldsymbol{a}^{-1} .* \big(\boldsymbol{U}_i^C - \boldsymbol{\mu}_i^C\big)\big\|_2^2}_{Self\text{-}based\ regulariztion} + \underbrace{\sum\nolimits_{k\in\boldsymbol{\mathcal{N}}_i^K} \varphi_k \big\| \boldsymbol{a}^{-1} .* \big(\boldsymbol{U}_i^C - \boldsymbol{\mu}_i^C\big)\big\|_2^2}_{neighbor\text{-}based\ regularization}\\
&\quad + \underbrace{\sum\nolimits_{r\in\boldsymbol{\mathcal{N}}^R} \varphi_r \big\| \boldsymbol{a}^{-1}\big(\boldsymbol{U}_i^C - \boldsymbol{\mu}_i^C\big)\big\|_2^2}_{expert\text{-}based\ regularization} + \underbrace{\varphi_0 \big\| \boldsymbol{a}^{-1} .* \boldsymbol{U}_i^C \big\|_2^2}_{complexity\text{-}regularization} + \underbrace{T(\cdot)}_{terms\ irrelevant\ to\ regularization}
\end{aligned}$$

From the above equation, we find that $\boldsymbol{U}_i^C$ is respectively regularized by four types of experts as specified in $\boldsymbol{T}_i$. In the first term, $\boldsymbol{\mu}_i^C$ is the user factor vector of the target user itself so it serves for *self-based regularization*. Since the majority of users are tail users with limited data, it is useful to borrow information from their neighbors. As a result, the user factor vectors $\{\boldsymbol{\mu}_k^C\}$ from i's neighbors $\boldsymbol{\mathcal{N}}_i^K$ are employed for *neighbor-based regularization*. Moreover, we involve a set of high-reputation experts $\boldsymbol{\mathcal{N}}^R$ in the system to conduct *expert-based regularization* because effective *self-based regularization* and *neighbor-based regularization* are often not available, e.g., a fully cold-start user who has no data available and no neighbors or a spam user who only links other spam users as his neighbors. The last regularization term is simply the most frequently used L_2-norm regularizer when $\boldsymbol{a}$ is set $\boldsymbol{1}$, which penalizes the complexity to prevent overfitting. Due to the equivalence between Equation (22) and Equation (23), Equation (25) can be

reformed to Equation (26):

$$\log PoGE\left(\boldsymbol{U}_i^C \middle| \left\{\boldsymbol{\mu}_n^C, \varphi_n\right\}_{n\in T_i}\right) = \log N\left(\boldsymbol{U}_i^C \middle| \hat{\boldsymbol{\mu}}_i^C, diag\left[\hat{\varphi}_i^{-1}\boldsymbol{a}\right]\right) = \hat{\varphi}_i \left\|\boldsymbol{a}^{-1}.^*\left(\boldsymbol{U}_i^C - \hat{\boldsymbol{\mu}}_i^C\right)\right\|_2^2 + T(\cdot)$$
$$where\ \hat{\boldsymbol{\mu}}_i^C = \frac{\sum_{n\in T_i}\varphi_n \boldsymbol{\mu}_n^C}{\hat{\varphi}_i} with\ \hat{\varphi}_i = \sum_{n\in T_i}\varphi_n \tag{26}$$

From the perspective of Equation (25), φ_n controls the penalty of loss for fitting $\boldsymbol{\mu}_n^C$, i.e. a higher reputation expert has a larger regularization effect. From the perspective of Equation (26), the empirical prior mean $\boldsymbol{\mu}_i^C$ is a weighted average user factor vector over $\boldsymbol{T}_i$, so $\boldsymbol{\mu}_i^C$ receives more contributions from higher reputation experts with a larger φ_n. Note that φ_0 is not a reputation score but a common regularization parameter as λ in Equation (9), and it can be determined by usual regularization parameter selection methods, such as cross-validation.

4.2.3. Parameter Learning. We can obtain the marginal log-likelihood by integrating $\boldsymbol{U}^C, \boldsymbol{V}^C$ from the joint distribution:

$$\log P(\boldsymbol{Y}) = \log \int P(\boldsymbol{Y}, \boldsymbol{U}^C, \boldsymbol{V}^C) d\boldsymbol{U}^C d\boldsymbol{V}^C$$
$$where\ P(\boldsymbol{Y}, \boldsymbol{U}^C, \boldsymbol{V}^C) = \prod_{ij\in \boldsymbol{O}} P\left(Y_{ij} \middle| \boldsymbol{U}_i^C, \boldsymbol{V}_j^C\right) \prod_i P\left(\boldsymbol{U}_i^C\right) \prod_j P\left(\boldsymbol{V}_j^C\right) \tag{27}$$

However, the computation of Equation (27) is generally intractable. To enable it to run efficiently on large-scale data and the precise learning parameters for our model, we use the variational Bayesian (VB) method, which provides a good balance between efficiency and accuracy in learning latent features [Kim and Choi 2013; Lim and Teh 2007; Shan and Banerjee 2010]. Now, if we let $Q(\boldsymbol{U}^C, \boldsymbol{V}^C)$ be the variational distribution, we can then obtain the lower bound by applying Jensen's inequality [Shan and Banerjee 2010].

$$\log P(\boldsymbol{Y}) \geq \int Q(\boldsymbol{U}^C, \boldsymbol{V}^C) \log \frac{P(\boldsymbol{Y}, \boldsymbol{U}^C, \boldsymbol{V}^C)}{Q(\boldsymbol{U}^C, \boldsymbol{V}^C)} d\boldsymbol{U}^C d\boldsymbol{V}^C \equiv \mathcal{L}(Q) \tag{28}$$

The lower bound $\mathcal{L}(Q)$ can be rewritten using the expectation conditional on $Q(\boldsymbol{U}^C, \boldsymbol{V}^C)$ from Equation (28), and it becomes tight only when $Q(\boldsymbol{U}^C, \boldsymbol{V}^C) = P(\boldsymbol{U}^C, \boldsymbol{V}^C | \boldsymbol{Y})$.

$$\mathcal{L}(Q) \equiv \mathbb{E}_Q[\log P(\boldsymbol{Y}|\boldsymbol{U}^C, \boldsymbol{V}^C) + \log P(\boldsymbol{U}^C) + \log P(\boldsymbol{V}^C)] + H[Q(\boldsymbol{U}^C, \boldsymbol{V}^C)] \tag{29}$$

Generally, it usually assumes $Q(\boldsymbol{U}^C, \boldsymbol{V}^C)$ has a factorial form [Kim and Choi 2013; Lim and Teh 2007; Shan and Banerjee 2010]:

$$Q(\boldsymbol{U}^C, \boldsymbol{V}^C) = Q(\boldsymbol{U}^C) Q(\boldsymbol{V}^C) = \prod_i Q\left(\boldsymbol{U}_i^C\right) \prod_j Q\left(\boldsymbol{V}_j^C\right) \tag{30}$$

Here, $Q(\boldsymbol{U}_i^C)$ and $Q(\boldsymbol{V}_j^C)$ are variational Gaussian distributions with diagonal variance matrices:

$$Q\left(\boldsymbol{U}_i^C\right) = N\left(\boldsymbol{U}_i^C \middle| \boldsymbol{u}_i^C, diag\left[\boldsymbol{\lambda}_i^C\right]\right) \qquad Q\left(\boldsymbol{V}_j^C\right) = N\left(\boldsymbol{V}_j^C \middle| \boldsymbol{v}_j^C, diag\left[\boldsymbol{\gamma}_j^C\right]\right) \tag{31}$$

Then, we can write Equation (29) in the following form by using Equations (16), (17), (18), and (30):

$$
\begin{aligned}
\mathcal{L}(Q) &= \sum_{ij\in \boldsymbol{O}} \mathbb{E}_{Q(\boldsymbol{U}_i^C)Q(\boldsymbol{V}_j^C)}\big[\log P\big(Y_{ij}|\boldsymbol{U}_i^C, \boldsymbol{V}_j^C\big)\big] + \sum_i \mathbb{E}_{Q(\boldsymbol{U}_i^C)} \log P\big(\boldsymbol{U}_i^C\big) + H\big[Q\big(\boldsymbol{U}_i^C\big)\big] \qquad (32)\\
&\quad + \sum_j \mathbb{E}_{Q(\boldsymbol{V}_j^C)} \log P\big(\boldsymbol{V}_j^C\big) + H\big[Q\big(\boldsymbol{V}_j^C\big)\big]\\
&= \sum_{ij\in \boldsymbol{O}} \mathbb{E}_{Q(\boldsymbol{U}_i^C)Q(\boldsymbol{V}_j^C)}\big[\log N\big(Y_{ij}|\boldsymbol{U}_i^{C\mathrm{T}}\boldsymbol{V}_j^C, t\omega_{ij}^{-1}\big)\big]\\
&\quad + \sum_i \mathbb{E}_{Q(\boldsymbol{U}_i^C)}\big[\log N\big(\boldsymbol{U}_i^C|\boldsymbol{\mu}_i^C, diag\big[\hat{\omega}_i^{-1}\boldsymbol{a}\big]\big)\big] + H\big[N\big(\boldsymbol{U}_i^C|\boldsymbol{u}_i^C, diag\big[\boldsymbol{\lambda}_i^C\big]\big)\big]\\
&\quad + \sum_j \mathbb{E}_{Q(\boldsymbol{V}_j^C)}\big[\log N\big(\boldsymbol{V}_j^C|\boldsymbol{\mu}_j^C, diag[\boldsymbol{b}]\big)\big] + H\big[N\big(\boldsymbol{V}_j^C|\boldsymbol{v}_j^C, diag\big[\boldsymbol{\gamma}_j^C\big]\big)\big]\\
&= \frac{1}{2}\left(\sum_{ij\in \boldsymbol{O}} \log 2\pi t^{-1}\omega_{ij} - t^{-1}\omega_{ij}\big[\big(\boldsymbol{Y}_{i,j} - \boldsymbol{u}_i^{C\mathrm{T}}\boldsymbol{v}_j^C\big).^2 + \big(\boldsymbol{u}_i^{C}.^2\big)^{\mathrm{T}}\boldsymbol{\gamma}_j^C + \boldsymbol{\lambda}_i^{C\mathrm{T}}\big(\boldsymbol{v}_j^{C}.^2 + \boldsymbol{\gamma}_j^C\big)\big]\right)\\
&\quad - \frac{1}{2}\left(\sum_i \log\big\|2\pi\hat{\varphi}_i\boldsymbol{a}.^{-1}\big\|_2 + \big[\big(\boldsymbol{u}_i^C - \boldsymbol{\mu}_i^C\big).^2 + \boldsymbol{\lambda}_i^C\big]^{\mathrm{T}}\hat{\varphi}_i\boldsymbol{a}.^{-1} - \log\big\|2\pi e\boldsymbol{\lambda}_i^C\big\|_2\right)\\
&\quad - \frac{1}{2}\left(\sum_i \log\big\|2\pi\boldsymbol{b}.^{-1}\big\|_2 + \big[\big(\boldsymbol{v}_i^C - \boldsymbol{\mu}_i^C\big).^2 + \boldsymbol{\gamma}_i^C\big]^{\mathrm{T}}\boldsymbol{b}.^{-1} - \log\big\|2\pi e\boldsymbol{\gamma}_i^C\big\|_2\right)
\end{aligned}
$$

Let us denote $\{\bar{\boldsymbol{U}}^C, \boldsymbol{\Lambda}^C, \bar{\boldsymbol{V}}^C, \boldsymbol{\Gamma}^C\}$ as the variational parameters and $\{\boldsymbol{a}, \boldsymbol{b}, t\}$ as the model parameters where $\bar{\boldsymbol{U}}^C = [\boldsymbol{u}_i^C]_{1\le i\le N}$ stands for a matrix consisting of mean vectors and $\boldsymbol{\Lambda}^C = [\boldsymbol{\lambda}_i^C]_{1\le i\le N}$ denotes a matrix consisting of the variance vectors of $Q(\boldsymbol{U}^C)$, and $\bar{\boldsymbol{V}}^C$, $\boldsymbol{\Gamma}^C$ are defined similarly w.r.t. $Q(\boldsymbol{V}^C)$. To maximize $\mathcal{L}(Q)$, we can use coordinate ascend, i.e., iteratively optimizing $\mathcal{L}(Q)$ by searching for a solution for one parameter at a time and fixing the others. Table II summarizes the updating scheme for each parameter.

4.2.4. The Tricks for Complexity Reduction. The data matrix $\boldsymbol{Y}$ and its corresponding weight matrix $\boldsymbol{W} = t^{-1}\boldsymbol{\omega}$ are very sparse as they pertain to explicit rating data, where non-zero entries in these two matrices are associated with observed ratings. Due to the factorial variational distribution $Q(\boldsymbol{U}^C, \boldsymbol{V}^C)$, the parameter updating scheme of Table II is naturally parallelizable. The updating scheme in Table II can be implemented in the same way as that used by Kim and Choi [2013] who designed a scalable parameter updating scheme for variational Bayesian MF. Accordingly, the time complexity is $\mathrm{O}(3K[\sum_i |\boldsymbol{W}_{i,:}^{\ge 0}| + \sum_j |\boldsymbol{W}_{:,j}^{\ge 0}|]) = \mathrm{O}(6K|\boldsymbol{O}|)$ as illustrated in Kim and Choi [2013], where $|\boldsymbol{W}_{i,:}^{\ge 0}|$ equals the number of observed ratings for user i, $|\boldsymbol{W}_{:,j}^{\ge 0}|$ equals the number of observed ratings for item j, and $|\boldsymbol{O}|$ is the total number of observed ratings. In practice, the length of the latent factor vector, K, is small, and in our experiments, it yields good results for $K \le 10$. Normally, the data density, $s = |\boldsymbol{O}|/NM$, of most explicit rating data sets is very small, i.e., large sparsity, in the real world, usually $s \le 0.01\%$ (e.g., the RED dataset). Therefore, this updating scheme is executed very efficiently.

In the case of implicit rating data, the blank entries in data matrix $\boldsymbol{Y}$ are also modeled as implicit feedback [Hu et al. 2014; Hu et al. 2008]. Accordingly, in this case, the weight matrix $\boldsymbol{W}$ is a full matrix, i.e., $|\boldsymbol{O}| = |\boldsymbol{W}^{\ge 0}|$, having the space complexity $\mathrm{O}(NM)$ (cf.

Table II. Parameter Updating Scheme for C-HMF

In the following equations, we denote $W_{ij} = t^{-1}\omega_{ij}$

—Update parameters $\{\boldsymbol{u}_i^C, \boldsymbol{\lambda}_i^C\}$ of distribution $Q(\boldsymbol{U}_i^C)$ in parallel, for each i:

$$\boldsymbol{u}_i^C \leftarrow \boldsymbol{\Psi}_i[\bar{\boldsymbol{V}}^C diag(\boldsymbol{W}_{i,:})\boldsymbol{Y}_{i,:}^{\mathrm{T}} + \hat{\varphi}_i \boldsymbol{a}_\cdot^{-1} .* \boldsymbol{\mu}_i^C] \tag{33}$$

$$where\ \boldsymbol{\Psi}_i^{-1} = diag(\hat{\varphi}_i \boldsymbol{a}_\cdot^{-1}) + \bar{\boldsymbol{V}}^C diag(\boldsymbol{W}_{i,:})\bar{\boldsymbol{V}}^{C\mathrm{T}} + diag(\boldsymbol{\Gamma}^C \boldsymbol{W}_{i,:}^{\mathrm{T}})$$

$$\boldsymbol{\lambda}_i^C \leftarrow (\bar{\boldsymbol{V}}^C{}_\cdot^{.2} + \boldsymbol{\Gamma}^C)\boldsymbol{W}_{i,:}^{\mathrm{T}} + \hat{\varphi}_i^{-1}\boldsymbol{a}$$

—Update parameters $\{\boldsymbol{v}_j^C, \boldsymbol{\gamma}_j^C\}$ of the distribution $Q(\boldsymbol{V}_j^C)$ in parallel, for each j:

$$\boldsymbol{v}_j^C \leftarrow \boldsymbol{\Psi}_j(\bar{\boldsymbol{U}}^C diag(\boldsymbol{W}_{:,j})\boldsymbol{Y}_{:,j} + \boldsymbol{b}_\cdot^{-1} .* \boldsymbol{\mu}_j^C) \tag{34}$$

$$where\ \boldsymbol{\Psi}_j^{-1} = diag(\boldsymbol{b}_\cdot^{-1}) + \bar{\boldsymbol{U}}^C diag(\boldsymbol{W}_{:,j})\bar{\boldsymbol{U}}^{C\mathrm{T}} + diag(\boldsymbol{\Lambda}^C \boldsymbol{W}_{:,j})$$

$$\boldsymbol{\gamma}_j^C \leftarrow (\bar{\boldsymbol{U}}^C{}_\cdot^{.2} + \boldsymbol{\Lambda}^C)\boldsymbol{W}_{:,j} + \boldsymbol{b}$$

—Update model parameters $\{\boldsymbol{a}, \boldsymbol{b}, t\}$:

$$\boldsymbol{a} \leftarrow \frac{\sum_i \hat{\varphi}_i\left[(\boldsymbol{u}_i^C - \boldsymbol{\mu}_i^C)_\cdot^{.2} + \boldsymbol{\lambda}_i^C\right]}{N}$$

$$\boldsymbol{b} \leftarrow \frac{\sum_j\left[(\boldsymbol{v}_j^C - \boldsymbol{\mu}_j^C)_\cdot^{.2} + \boldsymbol{\gamma}_j^C\right]}{M}$$

$$t \leftarrow \frac{\sum_{ij\in\boldsymbol{O}} \omega_{ij}\left[(\boldsymbol{Y}_{i,j} - \boldsymbol{u}_i^{C\mathrm{T}}\boldsymbol{v}_j^C)_\cdot^{.2} + (\boldsymbol{u}_i^C{}_\cdot^{.2})^{\mathrm{T}}\boldsymbol{\gamma}_j^C + (\boldsymbol{v}_j^C{}_\cdot^{.2} + \boldsymbol{\gamma}_j^C)^{\mathrm{T}}\boldsymbol{\lambda}_i^C\right]}{|\boldsymbol{O}|}$$

Equation (15) and Equation (21)). Normally, it is impractical to load such a full matrix $\boldsymbol{W}$ into memory. From the analysis above, the time for running this updating scheme on implicit rating data is $1/s$ (i.e., often more than 10,000) times slower than running it on explicit rating data. To improve the running performance on the implicit rating data, we can apply the following trick to reduce the complexity. If we let $\tilde{\boldsymbol{W}}_{i,:} = \boldsymbol{W}_{i,:} - c$ and $c = t^{-1}\epsilon$, we write each row of $\boldsymbol{W}$ as $\boldsymbol{W}_{i,:} = \tilde{\boldsymbol{W}}_{i,:} + c$. According to Equation (21), it is easy to see that $\tilde{\boldsymbol{W}}_{i,:}$ only has non-zero entries on observed ratings. Now, let us take updating for $\{\boldsymbol{u}_i^C, \boldsymbol{\lambda}_i^C\}$ as an example. In Equation (33), $\bar{\boldsymbol{V}}^C diag(\boldsymbol{W}_{i,:})\bar{\boldsymbol{V}}^{C\mathrm{T}}$ can be rewritten as $\bar{\boldsymbol{V}}^C diag(\tilde{\boldsymbol{W}}_{i,:})\bar{\boldsymbol{V}}^{C\mathrm{T}} + c\bar{\boldsymbol{V}}^C\bar{\boldsymbol{V}}^{C\mathrm{T}}$; obviously, the term $c\bar{\boldsymbol{V}}^C\bar{\boldsymbol{V}}^{C\mathrm{T}}$ is not dependent on the user index i, so it can be pre-computed in time at most $O(K^2M)$ and less than $O(KM)$ using parallel multiplication. Similarly, $\bar{\boldsymbol{V}}^C diag(\boldsymbol{W}_{i,:})\boldsymbol{Y}_{i,:}^{\mathrm{T}}$ can be written as $\bar{\boldsymbol{V}}^C diag(\tilde{\boldsymbol{W}}_{i,:})\boldsymbol{Y}_{i,:}^{\mathrm{T}} + c\bar{\boldsymbol{V}}^C\boldsymbol{Y}_{i,:}^{\mathrm{T}}$, where the term $c\bar{\boldsymbol{V}}^C\boldsymbol{Y}_{i,:}^{\mathrm{T}}$ can be computed in time less than $O(KM)$ due to the sparse $\boldsymbol{Y}_{i,:}$. Moreover, $\boldsymbol{\Gamma}^C\boldsymbol{W}_{i,:}^{\mathrm{T}}$ can be written as $\boldsymbol{\Gamma}^C\tilde{\boldsymbol{W}}_{i,:}^{\mathrm{T}} + c\boldsymbol{\Gamma}^C\boldsymbol{1}$, where the term $c\boldsymbol{\Gamma}^C\boldsymbol{1}$ can be computed in less time than $O(KM)$ because $\boldsymbol{\Gamma}^C\boldsymbol{1}$ is equivalent to summing $\boldsymbol{\Gamma}^C$ by rows. Using the same trick, the additional time in parallel computing $\{\boldsymbol{\lambda}_i^C\}$ is also $O(KM)$. Therefore, the overall additional time cost is $O(4KM)$ when learning $\{\boldsymbol{u}_i^C, \boldsymbol{\lambda}_i^C\}$ in this parallel fashion. When applying this trick to updating $\{\boldsymbol{v}_j^C, \boldsymbol{\gamma}_j^C\}$, the overall additional time cost is $O(4KN)$. Moreover, we can compute t by summing over

i in a parallel way, where the additional time cost is also O(KM) since $\boldsymbol{\omega}_{i,:} = t\boldsymbol{W}_{i,:}$. As a result, the overall additional time cost for implicit rating data is O($K[M+N]$), so the whole time complexity is O($6K|\tilde{\boldsymbol{W}}^{\geq 0}|$) + O($4KM$) + O($4KN$) + O($K[M+N]$) = O($6K|\tilde{\boldsymbol{W}}^{\geq 0}| + 5K[M+N]$) < O($6K[|\tilde{\boldsymbol{W}}^{\geq 0}| + M + N]$). Normally, $M + N \ll |\tilde{\boldsymbol{W}}^{\geq 0}|$, so O($6K[|\tilde{\boldsymbol{W}}^{\geq 0}| + M + N]$) is within the same order as O($6K|\tilde{\boldsymbol{W}}^{\geq 0}|$).

By applying this trick, updating the equations depends on the sparse weight matrix $\tilde{\boldsymbol{W}}$ instead of the full matrix $\boldsymbol{W}$, so the space complexity to store $\tilde{\boldsymbol{W}}$ is the same as the sparse weight matrix in the case of explicit rating data. Therefore, it can be concluded that the time and space complexities of the learning parameters on the implicit rating data is a little higher than those on the explicit rating data, but still in the same order.

4.3. Learning Regularized S-HMF Model

When the parameter set, $\{\bar{\boldsymbol{U}}^C, \boldsymbol{\Lambda}^C, \bar{\boldsymbol{V}}^C, \boldsymbol{\Gamma}^C\}$, is learned from C-HMF, we obtain the distribution of $\boldsymbol{U}_i^C$ and $\boldsymbol{V}_j^C$ approximated by the variational distributions $Q(\boldsymbol{U}_i^C)$ and $Q(\boldsymbol{V}_j^C)$. Therefore, we can sample $\boldsymbol{\mu}_i^S \sim Q(\boldsymbol{U}_i^C)$, $\boldsymbol{\mu}_j^S \sim Q(\boldsymbol{V}_j^C)$ as the means of empirical prior distributions for S-HMF. To avoid unnecessary sampling noise, the expectations, $\boldsymbol{\mu}_i^S = \mathbb{E}[Q(\boldsymbol{U}_i^C)] = \boldsymbol{u}_i^C$ and $\boldsymbol{\mu}_j^S = \mathbb{E}[Q(\boldsymbol{V}_j^C)] = \boldsymbol{v}_j^C$ are often used as the means of empirical prior distributions. As a result, we can write the probabilistic model of S-HMF, shown in the right part of RMRM in Figure 4 as follows:

$$P(\boldsymbol{U}_i^S) = PoGE\left(\boldsymbol{U}_i^S \middle| \{\boldsymbol{\mu}_n^S, \varphi_n\}_{n \in \boldsymbol{T}_i}\right) \tag{35}$$

$$P(\boldsymbol{V}_j^S) = N(\boldsymbol{V}_j^S | \boldsymbol{\mu}_j^S, diag[\boldsymbol{d}]) \tag{36}$$

$$P(Y_{ij} | \boldsymbol{U}_i^S, \boldsymbol{V}_j^S) = N(Y_{ij} | \boldsymbol{U}_i^{S\mathrm{T}} \boldsymbol{V}_j^S, r\eta_{ij}^{-1}) \tag{37}$$

where $diag[\boldsymbol{c}]$ and $diag[\boldsymbol{d}]$ are diagonal covariance matrices. Additionally, $r\eta_{ij}^{-1}$ denotes the variance of likelihood, where η_{ij} is a novelty score given by the variance model and r is a scale parameter to be learned.

4.3.1. Heteroscedastic Modeling on Specialty. As discussed previously, popular items tend to be widely known by users and have more interaction, so both the choices of and the feedback for these items may largely be influenced by others, whereas tail items tend to be chosen more independently, thus the choices of these items can better reflect the personal preferences of users [Vargas and Castells 2011]. As a result, we model the specialty of user choices on the basis of the popularity of items.

Specialty Modeling. If we denote the probability of choosing item j as θ_j, then we have the multinomial distribution over all the items, where $\Gamma(\cdot)$ is the gamma function and $|\boldsymbol{O}_j|$ denotes the number of observed choices of item j:

$$Mult(\{|\boldsymbol{O}_j|\} | \boldsymbol{\theta}) = \frac{\Gamma\left(\sum_j |\boldsymbol{O}_j| + 1\right)}{\prod_j \Gamma(|\boldsymbol{O}_j| + 1)} \prod_j \theta_j^{|\boldsymbol{O}_j|}$$

Moreover, we place a symmetric Dirichlet-prior, $Dir(\boldsymbol{\theta}|\alpha)$ on $\boldsymbol{\theta}$, where the hyper-parameter α can be interpreted as the number of pre-given, pseudo-choices of each item. Then, we can obtain the posterior for all observations:

$$Dir(\{|\boldsymbol{O}_j| + \alpha\}) \propto \int Mult(\{|\boldsymbol{O}_j|\} | \boldsymbol{\theta}) Dir(\boldsymbol{\theta}|\alpha) d\boldsymbol{\theta}$$

The expectation of this posterior on choosing item j is:

$$\mathbb{E}_{|\boldsymbol{O}_j|}[Dir(\{|\boldsymbol{O}_j|+\alpha\})] = \frac{|\boldsymbol{O}_j|+\alpha}{\sum_j(|\boldsymbol{O}_j|+\alpha)} \equiv \bar{p}(j|\alpha) \tag{38}$$

where $\bar{p}(j|\alpha)$ is the smoothed version of the probability of choosing item j to avoid zero probability—a.k.a. Laplace smoothing—of the new items or the items with uncounted choice in a given dataset. In information theory, self-information is a measure of the information content associated with an event in a probability space. Here, a choice is such an event. As analyzed previously, choices on tail items can reflect users' special preferences, i.e., these choices contain more information content. As a result, we give the following definition of specialty of choice in terms of self-information:

Definition 3 (*Specialty of Choice*). Given all observed choices, the specialty of a choice on an item j is measured by self-information:

$$\psi_j = -\log \bar{p}(j|\alpha) \tag{39}$$

Specialty Score. We assign the credibility of feedback on a choice in terms of two scores: the reputation of a user (a global score), and the helpfulness of the review (a local score), thus we obtain:

$$\eta_{ij} \stackrel{\text{def}}{=} \begin{cases} \psi_j + \epsilon & (i,j) \text{ is an observed choice} \\ \epsilon & \text{otherwise} \end{cases} \tag{40}$$

That is, the observation is associated with a high credibility score only if a high-reputation user gives a helpful review. In particular, we set $\epsilon = 0$ for explicit rating data while ϵ is set to a small constant for implicit rating data (cf. Section 3.3).

Since a higher credibility score means a higher level of confidence in that choice, the variance function of a piece of feedback can be given by $f^S(Y_{ij}) = r\eta_{ij}^{-1}$ (note that lower variance means higher confidence), where t is a scale parameter to be learned.

4.3.2. Parameter Learning. Similar to the derivation of VB on C-HMF, we can easily obtain the lower bound of marginal log-likelihood of S-HMF:

$$\log P(\boldsymbol{Y}) \geq \int Q(\boldsymbol{U}^S, \boldsymbol{V}^S) \log \frac{P(\boldsymbol{Y}, \boldsymbol{U}^S, \boldsymbol{V}^S)}{Q(\boldsymbol{U}^S, \boldsymbol{V}^S)} d\boldsymbol{U}^S d\boldsymbol{V}^S \equiv \mathcal{L}(Q) \tag{41}$$

where $Q(\boldsymbol{U}^S, \boldsymbol{V}^S) = \prod_i Q(\boldsymbol{U}_i^S) \prod_j Q(\boldsymbol{V}_j^S) = \prod_i N(\boldsymbol{U}_i^S | \boldsymbol{u}_i^S, diag[\boldsymbol{\lambda}_i]) \prod_j N(\boldsymbol{V}_j^S | \boldsymbol{v}_j^S, diag[\boldsymbol{\gamma}_j])$ is a factorized variational Gaussian distribution. The parameter updating scheme is given in Table III; here, the variational parameters $\{\bar{\boldsymbol{U}}^S, \boldsymbol{\Lambda}^S, \bar{\boldsymbol{V}}^S, \boldsymbol{\Gamma}^S\}$ and the model parameters $\{\boldsymbol{c}, \boldsymbol{d}, r\}$ are updated in turn to maximize $\mathcal{L}(Q)$, where $\bar{\boldsymbol{U}}^S = [\boldsymbol{u}_i^S]_{1 \leq i \leq N}$ denotes a matrix consisting of mean vectors and $\boldsymbol{\Lambda}^S = [\boldsymbol{\lambda}_i^S]_{1 \leq i \leq N}$ denotes a matrix consisting of variance vectors w.r.t. $Q(\boldsymbol{U}^S)$, and where $\bar{\boldsymbol{V}}^S$, $\boldsymbol{\Gamma}^S$ are defined similarly w.r.t. $Q(\boldsymbol{V}^S)$. With the same trick as that applied to C-HMF, we can efficiently implement this parameter updating scheme on the implicit rating data.

After the parameters of S-HMF are learned, we can either sample $\boldsymbol{\mu}_i^S \sim Q(\boldsymbol{U}_i^S)$ or use the expectations of the variational Gaussian distribution, $\boldsymbol{\mu}_i^S = \mathbb{E}[Q(\boldsymbol{U}_i^S)] = \boldsymbol{u}_i^S$, to construct the PoGE-based empirical priors for the coupled model, C-HMF (cf. Equation (22)).

4.4. Algorithm and Prediction

So far, we have presented the details of RMRM and the parameter learning schemes w.r.t. C-HMF and S-HMF, respectively. Algorithm 1 summarizes the whole learning process with recurrent regularization in terms of coupled empirical priors.

Table III. Parameter Updating Scheme for S-HMF

In the following equations, we denote $W_{ij} = r^{-1}\eta_{ij}$.

—Update parameters $\{\boldsymbol{u}_i^S, \boldsymbol{\lambda}_i^S\}$ of the distribution $Q(\boldsymbol{U}_i^S)$ in parallel, for each i:

$$\boldsymbol{u}_i^S \leftarrow \boldsymbol{\Psi}_i\left[\bar{\boldsymbol{V}}^S diag(\boldsymbol{W}_{i,:})\boldsymbol{Y}_{i,:}^{\mathrm{T}} + \hat{\varphi}_i \boldsymbol{c}_{\cdot}^{-1} .* \boldsymbol{\mu}_i^S\right] \tag{42}$$

$$where\ \boldsymbol{\Psi}_i^{-1} = diag[\hat{\varphi}_i \boldsymbol{c}_{\cdot}^{-1}] + \bar{\boldsymbol{V}}^S diag(\boldsymbol{W}_{i,:})\bar{\boldsymbol{V}}^{S\mathrm{T}} + diag\left(\boldsymbol{\Gamma}^S \boldsymbol{W}_{i,:}^{\mathrm{T}}\right)$$

$$\boldsymbol{\lambda}_i^S \leftarrow (\bar{\boldsymbol{V}}^S{}_{\cdot}^{.2} + \boldsymbol{\Gamma}^S)\boldsymbol{W}_{i,:}^{\mathrm{T}} + \hat{\varphi}_i^{-1} c$$

—Update parameters $\{\boldsymbol{v}_j^S, \boldsymbol{\gamma}_j^S\}$ of the distribution $Q(\boldsymbol{V}_j^S)$ in parallel, for each j:

$$\boldsymbol{v}_j^S \leftarrow \boldsymbol{\Psi}_j\left(\bar{\boldsymbol{U}}^S diag\left(\boldsymbol{W}_{:,j}\right)\boldsymbol{Y}_{:,j} + \boldsymbol{d}_{\cdot}^{-1} .* \boldsymbol{\mu}_j^S\right) \tag{43}$$

$$where\ \boldsymbol{\Psi}_j^{-1} = diag(\boldsymbol{d}_{\cdot}^{-1}) + \bar{\boldsymbol{U}}^S diag(\boldsymbol{W}_{:,j})\bar{\boldsymbol{U}}^{S\mathrm{T}} + diag(\boldsymbol{\Lambda}^S \boldsymbol{W}_{:,j})$$

$$\boldsymbol{\gamma}_j^S \leftarrow (\bar{\boldsymbol{U}}^S{}_{\cdot}^{.2} + \boldsymbol{\Lambda}^S)\boldsymbol{W}_{:,j} + \boldsymbol{d}$$

—Update model parameters $\{\boldsymbol{c}, \boldsymbol{d}, r\}$:

$$\boldsymbol{c} \leftarrow \frac{\sum_i \hat{\varphi}_i\left[\left(\boldsymbol{u}_i^S - \boldsymbol{\mu}_i^S\right)_{\cdot}^{.2} + \boldsymbol{\lambda}_i^S\right]}{N}$$

$$\boldsymbol{d} \leftarrow \frac{\sum_j \left[\left(\boldsymbol{v}_j^S - \boldsymbol{\mu}_j^S\right)_{\cdot}^{.2} + \boldsymbol{\gamma}_j^S\right]}{M}$$

$$r \leftarrow \frac{\sum_{ij\in \boldsymbol{O}} \eta_{ij}\left[\left(\boldsymbol{Y}_{i,j} - \boldsymbol{u}_i^{S\mathrm{T}}\boldsymbol{v}_j^S\right)_{\cdot}^{.2} + \left(\boldsymbol{u}_i^S{}_{\cdot}^{.2}\right)^{\mathrm{T}}\boldsymbol{\gamma}_j^S + \left(\boldsymbol{v}_j^S{}_{\cdot}^{.2} + \boldsymbol{\gamma}_j^S\right)^{\mathrm{T}}\boldsymbol{\lambda}_i^S\right]}{|\boldsymbol{O}|}$$

In Algorithm 1, we run k-step variational updating for both C-HMF (cf. Line 6) and S-HMF (cf. Line 9). In practice, this works well with a small k (less than 10). This type of updating strategy can be viewed as k-step, mean-field, contrastive divergence [Welling and Hinton 2002], which has proved its effectiveness theoretically.

Moreover, we use PoGE to approximate the distribution of user factors of those who are fully cold-start users without any feedback (cf. Lines 7 and 10). Since there is no data available for a fully cold-start user to update his/her user factors, we post-update them using the updated user factors from their mostly related trusters when the sub-iterations of C-HMF and S-HMF are finished.

Prediction. After the parameters of RMRM are learned, we obtain the regularized estimates $\{\boldsymbol{U}^C, \boldsymbol{V}^C\}$ for C-HMF and $\{\boldsymbol{U}^S, \boldsymbol{V}^S\}$ for S-HMF. We can predict the missing entries of the *user-item* matrix using the MF reconstruction form using these estimates. According to the variational approximation, we have $\boldsymbol{U}_i^C \sim Q(\boldsymbol{u}_i^C, \boldsymbol{\lambda}_i^C)$ and $\boldsymbol{V}_j^C \sim Q(\boldsymbol{v}_j^C, \boldsymbol{\gamma}_j^C)$; the means of $\boldsymbol{U}_i^C$ and $\boldsymbol{V}_j^C$ are just $\boldsymbol{u}_i^C$ and $\boldsymbol{v}_j^C$ which can be obtained from Equation (33) and Equation (34). Therefore, we can reconstruct the value of entry (i, j) as follows:

$$\hat{Y}_{ij} = \mathbb{E}\left[P\left(\boldsymbol{U}_i^{C\mathrm{T}}\boldsymbol{V}_j^C\right)\right] \approx \mathbb{E}\left[Q\left(\boldsymbol{U}_i^C\right)\right]^{\mathrm{T}}\mathbb{E}\left[Q\left(\boldsymbol{V}_j^C\right)\right] = \boldsymbol{u}_i^{C\mathrm{T}}\boldsymbol{v}_j^C \tag{44}$$

Similarly, we can reconstruct the value of entry (i, j) using $\boldsymbol{U}_i^S$ and $\boldsymbol{V}_j^S$:

$$\hat{Y}_{ij} = \mathbb{E}\left[P\left(\boldsymbol{U}_i^{S\mathrm{T}}\boldsymbol{V}_j^S\right)\right] \approx \mathbb{E}\left[Q\left(\boldsymbol{U}_i^S\right)\right]^{\mathrm{T}}\mathbb{E}\left[Q\left(\boldsymbol{V}_j^S\right)\right] = \boldsymbol{u}_i^{S\mathrm{T}}\boldsymbol{v}_j^S \tag{45}$$

ALGORITHM 1: Parameter Learning for RMRM

***Pre-computing*:**

Compute credibility score ω_{ij} for each entry using Equation (21);
Compute specialty score η_{ij} for each entry using Equation (40);

***Model Learning*:**

$it \leftarrow 0$
while $it \leq$ MAX_ITERATION
- Learning C-HMF:
Construct empirical priors via Equations (16, 17) using $\{\boldsymbol{\mu}_i^S, \boldsymbol{\mu}_j^S\}$ from S-HMF;
Run k-step parameter updating as Table II;
For each fully cold-start user c

$$\boldsymbol{U}_c^C = \mathbb{E}[PoGE(\{\boldsymbol{U}_t^C, \varphi_t\}_{t \in \mathcal{N}_i^K \cup \mathcal{N}^R})];$$

- Learning S-HMF:
Construct empirical priors via Equations (35, 36) using $\{\boldsymbol{\mu}_i^C, \boldsymbol{\mu}_j^C\}$ from C-HMF;
Run k-step parameter updating as Table III;
For each fully cold-start user c

$$\boldsymbol{U}_c^S = \mathbb{E}[PoGE(\{\boldsymbol{U}_t^S, \varphi_t\}_{t \in \mathcal{N}_i^K \cup \mathcal{N}^R})];$$

-Checking Convergence:
If the performance over validation set is not improved in some consecutive iterations, then ***break***;
$it \leftarrow it + 1$;
end

Whether to choose the prediction result from Equation (44) or Equation (45) is dependent on specific data sets. In general, the prediction results from Equation (45) place more emphasis on the personal taste for specific choices, so-called RMRM-S, whereas the prediction result from Equation (44) may achieve better performance in a system with a large amount of spam feedback, so-called RMRM-C. In practice, we choose one of these dependent on a real-world environment.

5. DISCUSSION

So far, we have presented RMRM and the corresponding learning algorithm. In fact, the idea and the methods adopted by RMRM have direct connections with other methods. Hence, we discuss these connections in this section.

5.1. Social Regularization from PoGE Perspective

In recent years, one prevalent approach of recommender systems has been to incorporate social relationships for regularization [Jamali and Ester 2010; Ma et al. 2011]. This method is built on the basic idea that users' preferences are mostly influenced by others with the strongest social relationships, typically, their trusters. In general, the social regularization on a user i often leads to the following two forms of the regularization term, and we denote them as SR1 [Ma et al. 2011] and SR2 [Jamali and Ester 2010; Ma et al. 2011], respectively.

$$SR1: \qquad \lambda_1 \left\| \boldsymbol{U}_i - \frac{\sum_{t \in \boldsymbol{T}_i} s_{it} \boldsymbol{U}_t^2}{\sum_{t \in \boldsymbol{T}_i} s_{it}} \right\|_2^2$$

$$SR2: \qquad \lambda_2 \sum_{t \in \boldsymbol{T}_i} s_{it} \|\boldsymbol{U}_i - \boldsymbol{U}_t\|_2^2$$

where $\boldsymbol{T}_i$ denotes i's truster set, s_{it} is the strength or similarity between i and t [Ma et al. 2011], or where we can simply use $s_{it} = 1$ to denote an observed link [Jamali and Ester 2010].

It is interesting to find that, in fact, both SR1 and SR2 are identical from the PoGE perspective, as both of them actually correspond to the same PoGE-prior. Now, let us set up the PoGE-prior for user i as follows:

$$PoGE(\boldsymbol{U}_i|\{\boldsymbol{U}_t, s_{it}\}_{t\in\boldsymbol{T}_i}) = \prod_{t\in\boldsymbol{T}_i} N\left(\boldsymbol{U}_i|\boldsymbol{U}_t, \lambda_2^{-1}s_{it}^{-1}\boldsymbol{I}\right) \tag{46}$$

Obviously, we can obtain SR2 by taking the negative log-form of Equation (46). According to Equation (23), we can obtain the following equivalent form from Equation (46):

$$PoGE(\boldsymbol{U}_i|\{\boldsymbol{U}_t, s_{it}\}_{t\in\boldsymbol{T}_i}) = N\left(\boldsymbol{U}_i \left| \frac{\sum_{t\in\boldsymbol{T}_i} s_{it}\boldsymbol{U}_t}{\sum_{t\in\boldsymbol{T}_i} s_{it}}, \lambda_1^{-1}\boldsymbol{I}\right.\right) \tag{47}$$

where $\lambda_1 = \lambda_2 \sum_{t\in\boldsymbol{T}_i} s_{it}$. By taking the negative log-form of Equation (47), we immediately obtain SR1. Therefore, SR1 and SR2 are actually derived from the same PoGE-prior, so we prove the identity between them. In fact, by using SR1 and SR2, the evaluation results are very close [Ma et al. 2011]. The small difference is probably caused by the settings of the regularization parameters λ_1 and λ_2 and by the random initialization of the parameters.

5.2. Multi-Objective Optimization

RMRM consists of two main components, where C-HMF models user choices by emphasizing credibility and S-HMF models user choices by emphasizing specialty. Each component leads to an objective for optimization, so RMRM can be viewed partially as a multi-objective optimization (MOO) [Deb 2014] problem. However, the conventional MOO problem often has two independent objectives, thus it needs to obtain solutions using higher-level information, whereas the two objectives of RMRM are coupled by the empirical priors induced from each other. In fact, the two objectives of RMRM are constructed from the same data, and we use a recurrent algorithm to learn the parameters that are regularized by the empirical priors induced from each other objective model. Therefore, RMRM is a variant case of MOO.

In general, the optimal solution of MOO is not unique, and it often uses a genetic algorithm to search the solution space [Deb 2014]. The recurrent learning algorithm of RMRM induces new empirical priors in each iteration, and S-HMF and C-HMF are reset using the new priors, which leads to new objectives for optimization, cf. Equation (28) and Equation (41). Hence, an iteration of RMRM corresponds to a generation of a genetic algorithm to search for the optimal solution. Taking Equation (41) as an example, the new objective may find better estimates of the parameters, provided that we have learned better priors $P(\boldsymbol{U}_i^S)$ and $P(\boldsymbol{V}_j^S)$, leading to better $P(\boldsymbol{Y}|\boldsymbol{U}^S, \boldsymbol{V}^S)$. As a result, the marginal likelihood $P(\boldsymbol{Y})$ is improved (cf. Equation (41)). Moreover, the new empirical priors from the peer model can help to find a better optimal solution in the next iteration. In comparison, the objective function of a single-objective model, such as MF, does not change with iterations so they more easily become stuck in local minima.

6. EXPERIMENTS

We conduct empirical evaluations using two real-world datasets that cover the cases of, respectively, explicit rating data and implicit rating data. We compare RMRM with a set of state-of-the-art methods gauged by various metrics. The overall results prove that our approach significantly outperforms all the compared methods.

6.1. A Comparison of the State-of-the-Art Methods

In the following experiments, a group of state-of-the-art methods are employed for comparison; some are used for explicit rating data and others for implicit rating data.

—*PMF* [Salakhutdinov and Mnih 2008b]. The conventional probabilistic MF model learns the factors of users and items from a rating matrix without taking additional information into account.
—*Trust-kNN* [Jøsang et al. 2013]. This method takes the top-k, high-reputational trusters of a user as the neighbors, and then predicts the user's rating of an item by averaging the available neighbors' ratings for that item.
—*SoRec* [Ma et al. 2008]. This method jointly models the trust-link matrix and a user-item rating matrix, which shares user factors to propagate the interaction between two matrices.
—*SoReg* [Ma et al. 2011]. This method utilizes the trust relationships to construct the regularizer to learn user factors.
—*SocialMF* [Jamali and Ester 2010]. This method is very similar to SoReg. The main difference lies in the setting of similarities for trusters (cf. Section 5.1).
—*MF-IR* [Hu et al. 2008; Pan et al. 2008]: This is a zero-mean, regularized, MF model, which is able to deal with implicit rating data.
—*SoRec-IR, SoReg-IR, SocialMF-IR*: The original versions of SoRec, SoReg, and SocialMF were designed for learning preferences from explicit rating data. To enable them to deal with implicit rating data, we extend them using weight modeling, as in *MF-IR* (cf. Equation (14)).
—*C-HMF*: One of the main components of RMRM, as presented in Section 4.2, is to enhance credibility-based modeling. Moreover, we use zero-mean regularization since the single model does not have the empirical priors learned through S-HMF.
—*S-HMF*: One of the main components of RMRM, as presented in Section 4.3, is to enhance specialty-based modeling. Moreover, we use zero-mean regularization since the single model does not have the empirical priors learned through C-HMF.
—*RMRM*: RMRM is the main model proposed in this article. Since C-HMF and S-HMF can model both explicit rating data and implicit rating data in a unified way, RMRM naturally has an advantage. In particular, we use RMRM-C to denote the prediction results generated using Equation (44), and RMRM-S to denote the prediction result generated using Equation (45).

6.2. Evaluation Metric

In the following experiments, we use rating metrics to evaluate the performance of the explicit rating data while using ranking metrics to assess the performance of implicit rating data.

6.2.1. Rating Metrics. To measure the accuracy of rating prediction, we utilize the most widely used evaluation metrics, namely, mean absolute error (MAE) [Herlocker et al. 2004].

$$MAE = \frac{\sum_{ij \in \boldsymbol{Ts}} abs(Y_{ij} - \hat{Y}_{ij})}{|\boldsymbol{Ts}|}$$

where Y_{ij} denotes a true rating in the testing set $\boldsymbol{Ts}$ and $\hat{Y}_{ij}$ is the predicted rating.

6.2.2. Ranking Metrics. The common way to assess the performance of prediction on implicit feedback data is to measure whether relevant items are placed in the top positions of a recommendation list. Therefore, information retrieval metrics are often employed to evaluate the ranking performance of recommender systems. Here, $rel(k) = 1$ if the item at position k is relevant, and $rel(k) = 0$ otherwise.

Table IV. Statistics of the Epinions Dataset

# users: 39,902	# items: 63,027
# trust links: 43,8965	# trusters/users: 11
max # of trusters: 1,713	# users with zero truster: 14,202
# ratings: 734,441	density: 0.029%
# ratings/users: 18	# ratings/items: 11
max # ratings of user: 1,809	max # ratings of item: 2,112

—*recall@K*. This considers the fraction of relevant items for all N relevant items:

$$recall@K = \frac{\sum_{k=1}^{K} rel(k)}{N}$$

—*precision@K*. This considers the fraction of relevant items for top K recommended items:

$$precision@K = \frac{\sum_{k=1}^{K} rel(k)}{K}$$

—*AP@K*. Average precision (AP) is the average result over *precision*@1∼K, which is defined as:

$$AP@K = \frac{\sum_{k=1}^{K} rel(k) \times precision@k}{min(K, N)}$$

—*nDCG@K*. Normalized discounted cumulative gain (nDCG) [Burges et al. 2005] is a measure of ranking quality, which places greater emphasis on relevant items:

$$nDCG@K = \frac{DCG@K}{IDCG@K}$$

where IDCG means ideal DCG, and where we have:

$$DCG@K = \sum_{k=1}^{K} \frac{2^{rel(k)} - 1}{\log_2(k+1)}, \quad IDCG@K = \sum_{k=1}^{K} \frac{1}{\log_2(k+1)}$$

6.3. Explicit Rating Data Evaluation

6.3.1. Data Preparation. We construct a truncated dataset from the RED dataset [Meyffret et al. 2012], as mentioned in the introduction by filtering both users and items with fewer than three ratings. This is because no data will be available for training if a user or an item only accounts for one or two ratings that are held out for testing. In addition, we need at least two testing items for a user in order to evaluate the accuracy of the ranking for these items. The statistics of this evaluation dataset are illustrated in Table IV.

Figure 6 demonstrates the long-tail distributions for the number of ratings w.r.t. items and users. We find that a large number of both items and users in the tail have very few ratings. Therefore, this dataset is suitable to evaluate the performance of recommendations for users and items in the tail of distributions. The hyperparameters of the compared methods are tuned by cross-validation. Here, we find that the length of the latent factor vector can produce good results with this dataset by setting $K = 5$.

Figure 7 illustrates the distributions of the number of helpful scores w.r.t. items and users in this evaluation dataset. We find that they have similar long-tail distributions with those in Figure 6. This is a natural phenomenon because helpful scores are based on reviews—more reviews tend to receive more helpful scores. Thus, these helpful scores are used for the reputation model. In this experiment, we set $\alpha = 1$ and $\beta = 3$

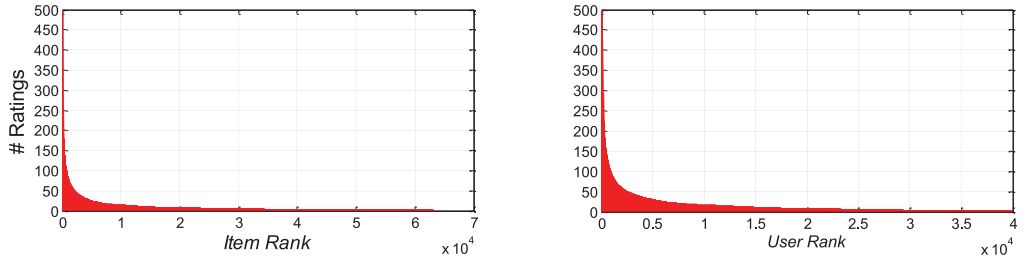

Fig. 6. Long-tail distributions for the number of ratings of items and users (truncated from 0 to 500).

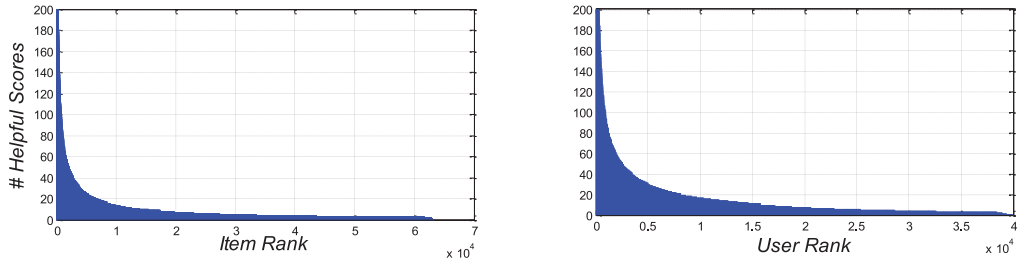

Fig. 7. The distributions for the number of helpful scores w.r.t. items and users (truncated from 0 to 200).

in Equation (20) to compute the reputation scores. That is, the initial reputation score is 0.25 for new users.

6.3.2. Prediction of Long-Tail Distributed Items. Improving the prediction performance of long-tail items would obviously bring more business profit to a company by precisely targeting a specific group of users. To evaluate the prediction performance of long-tail items, we randomly hold out 20% of the data from the evaluation dataset as the ground truths for testing, denoted as $\boldsymbol{Ts}_{20}$. Then, as shown below, we split $\boldsymbol{Ts}_{20}$ into four parts according to the popularity of the items so that we can compare the performance of different methods using both short-head items and long-tail items.

—*Most Popular.* The items in the headmost 5% of the distribution, as shown in the left-hand image of Figure 6.
—*Less Popular.* The items in the 5∼20% interval of the distribution.
—*Shallow Tail.* The items in the 20∼50% interval of the distribution.
—*Deep Tail.* The items in the endmost 50% of the distribution.

We evaluate the MAE of all comparative methods of these four parts of the distribution. Note that the data becomes extremely sparse in the deep tail, and, as a result, Trust-kNN is barely effective, as all of the neighbors tend never to rate the testing items. In such a case, we simply predict the rangs of an item as $Mean+\varepsilon$, where $Mean$ denotes the mean rating for all items, and ε is a small random value, following standard Gaussian distribution.

Figure 8 reports the results of the comparison of all methods for the four parts of the distribution. We find the performance of Trust-kNN decreases when the data become sparser since the tail items are rarely rated, which results in the random prediction mentioned above. Obviously, such neighborhood-based methods have a limitation when conducting recommendations in the long tail. We find that PMF outperforms Trust-kNN, as it does not need to search the neighborhood; instead, similarity is implicitly represented by latent factors. However, PMF suffers from the three aforementioned typical issues in long-tail recommendations. As illustrated by the four cases shown in

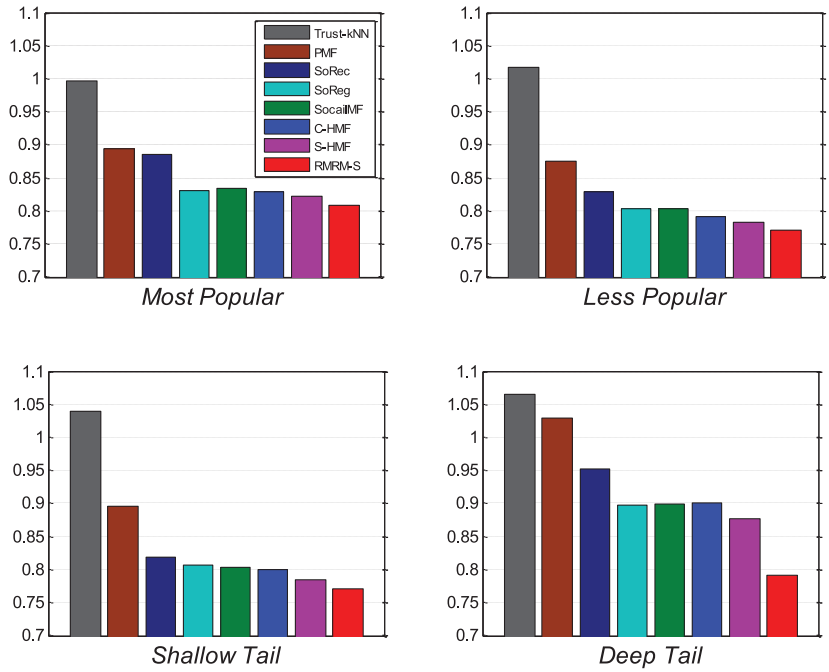

Fig. 8. MAEs of rating prediction for the long-tail item distribution.

Figure 8, we find that PMF achieves a relatively higher accuracy in the cases of *Most Popular* and *Less Popular*, but that the performance becomes worse when the available ratings for items become fewer, especially in the case of *Deep Tail*. In comparison, C-HMF improves the ability to alleviate shilling attacks, and it enables user preference for the long-tail items that are to be targeted in terms of heteroscedasticity modeling. As a result, C-HMF significantly outperforms PMF.

The remaining models involve trust relationships as the secondary information aspect, which addresses data insufficiency in the tail of distribution. Comparing SoRec with PMF, we find that the involved truster relationships are helpful to improve the accuracy of long-tail items. However, both the rating matrix and the trust matrix convey heterogeneous information, but SoRec cannot find a best trade-off point for all users. To overcome this deficiency, SoReg and SocialMF incorporate the context of trusters to regularize user factor learning. The results prove that SoReg is more effective than SoRec. In particular, RMRM-S is selected in this experiment since we would like to more aggressively emphasize users' special preferences over tail items. From the results, we easily find that RMRM-S achieves the best performance for all four cases. Note that the performance of *Deep Tail* is even better than *Most Popular*, which demonstrates that our model is able to better learn users' preferences from the long tail. Moreover, the deviations of the MAEs in the four cases are small. Such stable performance over the whole distribution may be attributed to the fusion of reliability and novelty, brought about by the coupled recurrent regularization. Therefore, we can conclude that RMRM-S is the most accurate model for recommending long-tail items.

6.3.3. Prediction on Long-Tail Distributed Users. Accurate recommendations for long-tail users can significantly improve users' experiences and users' retention rates. In the next experiment, we conduct an evaluation on the testing set $\boldsymbol{Ts}_{20}$. As in the previous

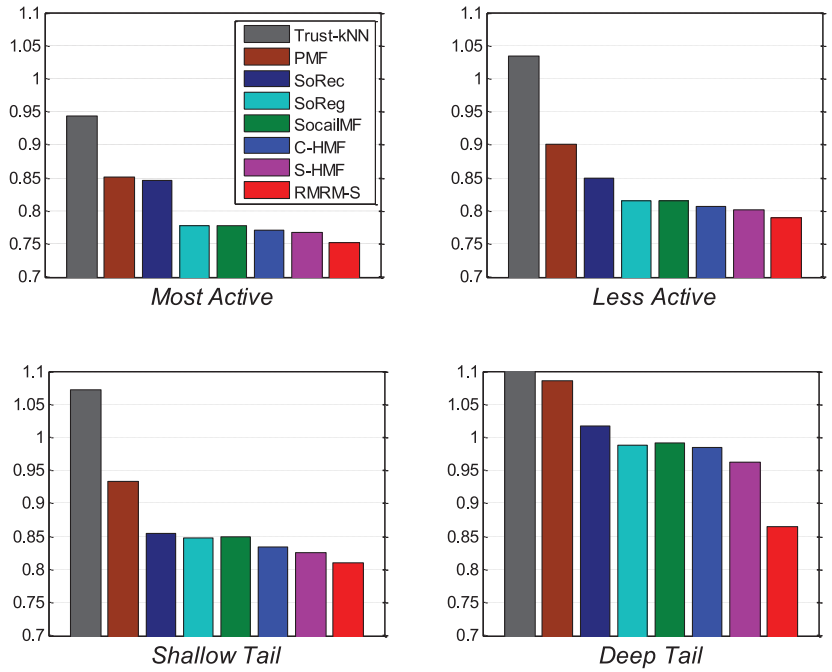

Fig. 9. MAEs of rating prediction for the long-tail user distribution.

experiment, we split $\boldsymbol{Ts}_{20}$ into four parts according to the activity of the users to compare the performance of both the short-head and long-tail users.

—*Most Active.* The users in the headmost 5% of the distribution, as shown in the right-hand image of Figure 6.
—*Less Active.* The users in the 5∼20% interval of the distribution.
—*Shallow Tail.* The users in the 20∼50% interval of the distribution.
—*Deep Tail.* The endmost 50% users of the distribution of the distribution.

Figure 9 shows the comparative results of all of the methods for the long-tail user distribution. We observe similar results to those in the previous experiment. Actually, conducting accurate predictions for deep-tail users is more difficult than for deep-tail items because almost all deep-tail users have both few ratings and few trust relationships. For those models that do not use trust relationships, S-HMF achieves the best performance since the heteroscedasticity modeling of user choices enables it to learn users' personal preferences better.

In particular, we found that more than one-third of the users have no links, as illustrated in Table IV. Consequently, SoRec cannot obtain secondary information for these users due to the lack of links in the trust matrix. Similarly, no truster is available to conduct regularization for SoReg and SocialMF. As a result, these methods cannot learn user factors when there is no trust link available for a cold-start user. To overcome this deficiency, RMRM-S incorporates top-N high-reputation experts into the system, cf. Equation (24). Hence, RMRM-S can still conduct regularization, even when no direct trusters are available. Since RMRM-S takes the advantages of C-HMF, S-HMF, and SoReg, it results in a significant improvement in recommendations for long-tail users.

6.3.4. Impact of the Number of Involved Trusters. The previous experiments show that borrowing knowledge from trusters can be very helpful to address the challenges of

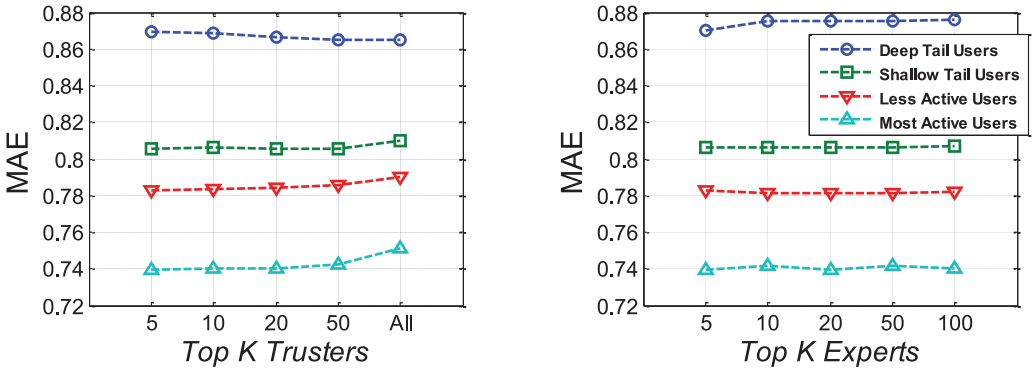

Fig. 10. MAEs varying the numbers of involved (a) user trusters, and (b) system experts.

recommendations for long-tail items. In RMRM, the truster set consists of two parts: the trusters that a user actively follows, and the experts with the highest reputation in the system, cf. Equation (24). We next illustrate the impact of the number of user trusters and the number of system trusters, respectively. In this experiment, we use the same testing set as in the previous experiments.

MAEs for Different Numbers of Involved User Trusters. We fix five high-reputation system experts and vary the number of top-K trusters, where $K \in \{5, 10, 20, 50, All\}$, to compare the performances. Figure 10(a) displays the results when the number of trusters changes. We find that increasing the number of trusters improves the performance of tail users. This is because they account for very little data, so there is a need to incorporate more trusters for regularization. In comparison, we find that the performance of head users is not improved, and even becomes worse when K increases. This can be attributed to the fact that head users account for sufficient data, which enables RSs to learn their preferences without borrowing information from others. Moreover, we find that involving too many trusters does not improve performance. It can thus be interpreted that the priors from too many trusters over-regularize the user preferences learning.

MAEs for Different Number of Involved System Experts. We fix the top five trusters of each user, and vary the number of top-K high-reputation system experts, where $K \in \{5, 10, 20, 50, 100\}$. From Figure 10(b), we find that the performance is very close under different K, becoming a little worse when K reaches 100. That is, it involves too many experts, which may over-regularize the user factors learning. As a result, we only need to involve a small group of system experts in practice.

6.3.5. Shilling Attack Simulation. In this experiment, we attempted to test the robustness of each model in a shilling attack environment. To simulate such an environment, we created 1,000 virtual spam users to conduct the attack, and we respectively selected 100 items from the head (0%∼20%) and the tail (20%∼100%) as the attack targets. In this experiment, we conducted nuke attack in the case of the average attack model [Burke et al. 2015]. More specifically, we first randomly selected the 50 most popular items from the head of distribution to serve as the filler item set [Burke et al. 2015]. Before conducting the attack, we assigned each item in the filter item set with the mean rating of that item for each spam user. As a result, we built a fake profile for these spam users who have average preferences that are similar to most users. Then, we simulated the nuke attack on each target item by injecting fifty minimum ratings,

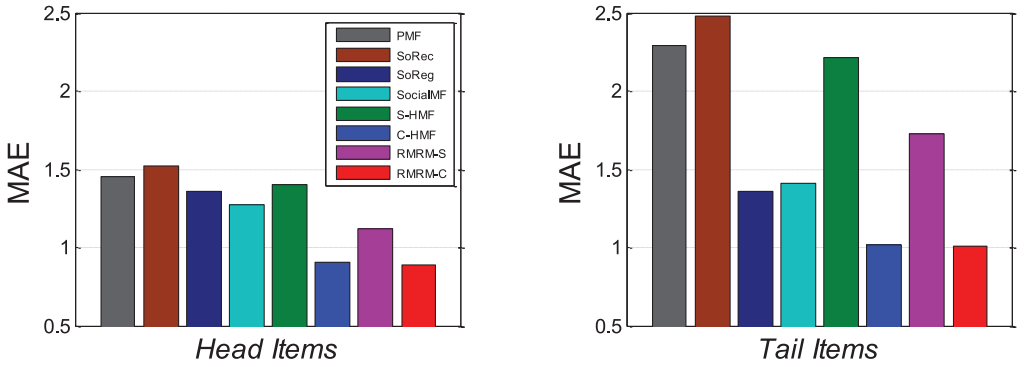

Fig. 11. MAEs for head items and tail items with shilling attack.

Table V. Statistics of Apps for Android Dataset

# users: 234,347	# Apps: 24,141
# installations: 1,274,896	density: 0.023%
# installations/users: 5.44	# installations/items: 52.81
max # installations of user: 565	max # installations of item: 11,801

i.e., 1, from fifty out of 1,000 spam users by random selection. Thus, we constructed a *user-item* rating matrix with fake ratings and spam users.

We retrained all the comparison models on this attacked rating matrix and then made predictions. Figure 11 illustrates the prediction results for the head items (left) and the tail items (right). Obviously, the MAEs of the head items are lower overall than those of the tail items, which reveals the fact that the tail items are more easily biased by fake ratings due to the few ratings they receive. PMF achieves poor performance because it is completely based on the ratings for each user without incorporating any other information, whereas SoReg and SocialMF are more robust to shilling attack due to the regularization from trusters or experts. In comparison, SoRec does not achieve comparable performance with SoReg and SocialMF, which illustrates that the impact from the fake rating matrix overwhelms that from the trust-link matrix, especially when the trust-link matrix is very sparse. RMRM-S achieves better performance than that of the single S-HMF model because S-HMF in RMRM-S is regularized by the empirical priors from C-HMF. Finally, we find that C-HMF and RMRM-C achieve much better performance than other models. In particular, the results of C-HMF and RMRM-C do not become worse as do the other models in the case of tail items, which proves that the heteroscedastic modeling for credibility is a very effective way to defend against shilling attack.

6.4. Implicit Rating Data Evaluation

6.4.1. Data Preparation. With the popularity of mobile phones, millions of apps have been published online, covering all aspects of daily life, including food, shopping, sports, games, and so on. Popular apps (head items) are known by most users, so recommending unpopular apps (tail items) to users is a more meaningful task. Here, we use a publicly available data set of apps for Android from Amazon [McAuley et al. 2015] to evaluate all the compared methods. Since the installation history is always available in the app store, for our experiment, we take the installation record as the implicit rating (with the observed installation of an item as 1). From the raw data, we remove users who have less than three installations and items that have less than four installations. The statistics of this evaluation dataset are illustrated in Table V.

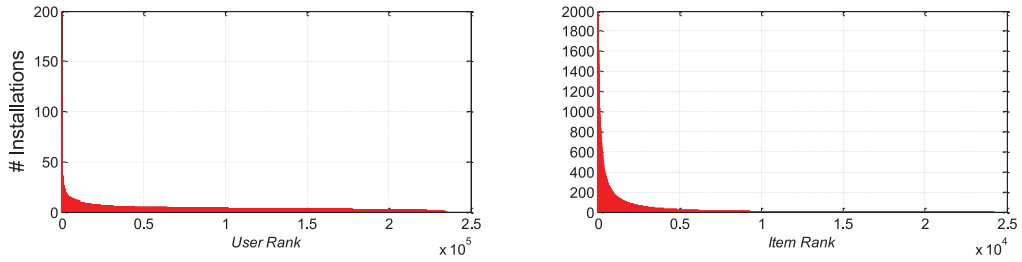

Fig. 12. Long-tail distributions over the number installations w.r.t. users and items (truncated).

Figure 12 shows the distributions of the number of installations w.r.t. users and items of this evaluation set. We see that the number of installations w.r.t. both items and users have obvious long-tail distributions. The hyperparameters of all the compared methods have been tuned by cross-validation. We find that the length of the latent factor vector can produce good results with this dataset by setting $K = 50$. Moreover, we set $\alpha = 1, \beta = 1$ for the reputation score, defined by Equation (20).

This dataset does not provide explicit relationships between the users. Intuitively, the number of choices of common apps between two users can be used to measure their similarity. Moreover, the choices on tail items better reflect user preferences. Therefore, we find the top-K neighbors of user i by ranking the weighted sum $\{\sum_{v \in \boldsymbol{O}(i,j)} \psi_v | j \neq i\}$ for all users except i, where ψ_v is defined by Equation (39) and $\boldsymbol{O}(i, j)$ is the set of common apps between i and j in the training set.

6.4.2. Evaluation of Tail Items Recommendation. For a real-world recommender system, generating an accurate list of attractive items for each user is more meaningful than accurately predicting ratings because, of course, the final goal of recommender systems is to find items desired by different users. In this experiment, we randomly hold out 20% of the observations from each item as the testing set, denoted as $\boldsymbol{Ts}_{20}$, and use the remainder for the training set. As in the previous experiments, we split $\boldsymbol{Ts}_{20}$ into four item groups according to the number of installations, namely *Most Popular*, *Less Popular*, *Shallow Tail*, and *Deep Tail*.

Table VI reports the mean AP@10, AP@20, nDCG@10, and nDCG@20 for testing the items in each group. In the case of *Most Popular*, the results from all models are relative close; this is due to the sufficient data of the head items. Overall, RMRM models achieve better performance than the other models, which proves that RMRM can better capture user preferences for tail items. In particular, RMRM-S and S-HMF achieve better performance than the other models in the cases of *Most Popular* and *Less Popular*, whereas RMRM-C and C-HMF outperform the others in the cases of *Shallow Tail* and *Deep Tail*. This reflects the fact that the apps in the tail are known by very few people, so that their installation and corresponding feedback mainly come from two types of users: (a) users who really have interest in these apps (i.e., valuable feedback), and (b) Internet marketers (i.e., valueless feedback). Accordingly, the designs of RMRM-C and C-HMF emphasize the feedback from the former and de-emphasize the feedback from the latter. Furthermore, RMRM-C incorporates the empirical priors from S-HMF for regularization, thus it achieves the best performance for recommending items in the tail.

Figure 13 depicts the recall@20∼50 curves for all compared models for recommending tail items (*Shallow Tail* and *Deep Tail*). Similar to the performance shown in Table VI, RMRM-based methods outperform the other approaches. Therefore, we find that the curve of RMRM-C is above all the other models with obvious margins, which, again,

Table VI. Mean AP@5, AP@10, nDCG@10, and nDCG@20 of Item Recommendations

	Most Popular				*Less Popular*			
Method	**AP@10**	**AP@20**	**nDCG@10**	**nDCG@20**	**AP@10**	**AP@20**	**nDCG@10**	**nDCG@20**
MF-IR	0.0135	0.0144	0.0160	0.0195	0.0112	0.0121	0.0187	0.0222
SoRec-IR	0.0135	0.0140	0.0162	0.0182	0.0111	0.0119	0.0180	0.0216
SoReg-IR	0.0133	0.0141	0.0163	0.0191	0.0119	0.0128	0.0191	0.0231
SocialMF-IR	0.0144	0.0150	0.0177	0.0200	0.0119	0.0128	0.0195	0.0232
S-HMF	0.0149	0.0156	0.0184	0.0211	0.0123	0.0131	0.0199	0.0237
C-HMF	0.0126	0.0131	0.0157	0.0174	0.0107	0.0115	0.0175	0.0208
RMRM-S	**0.0153**	**0.0159**	**0.0187**	**0.0212**	**0.0125**	**0.0132**	**0.0202**	**0.0239**
RMRM-C	0.0131	0.0136	0.0161	0.0180	0.0109	0.0116	0.0182	0.0213

	Shallow Tail				*Deep Tail*			
Method	**AP@10**	**AP@20**	**nDCG@10**	**nDCG@20**	**AP@10**	**AP@20**	**nDCG@10**	**nDCG@20**
MF-IR	0.0108	0.0096	0.0305	0.0331	0.0120	0.0077	0.0362	0.0342
SoRec-IR	0.0108	0.0095	0.0302	0.0328	0.0120	0.0076	0.0364	0.0338
SoReg-IR	0.0116	0.0103	0.0322	0.0349	0.0134	0.0086	0.0408	0.0383
SocialMF-IR	0.0111	0.0099	0.0315	0.0342	0.0128	0.0082	0.0396	0.0369
S-HMF	0.0129	0.0110	0.0355	0.0373	0.0160	0.0101	0.0475	0.0428
C-HMF	0.0171	0.0140	0.0438	0.0438	0.0238	0.0151	0.0654	0.0578
RMRM-S	0.0130	0.0110	0.0356	0.0374	0.0165	0.0105	0.0485	0.0448
RMRM-C	**0.0175**	**0.0142**	**0.0453**	**0.0445**	**0.0240**	**0.0154**	**0.0659**	**0.0592**

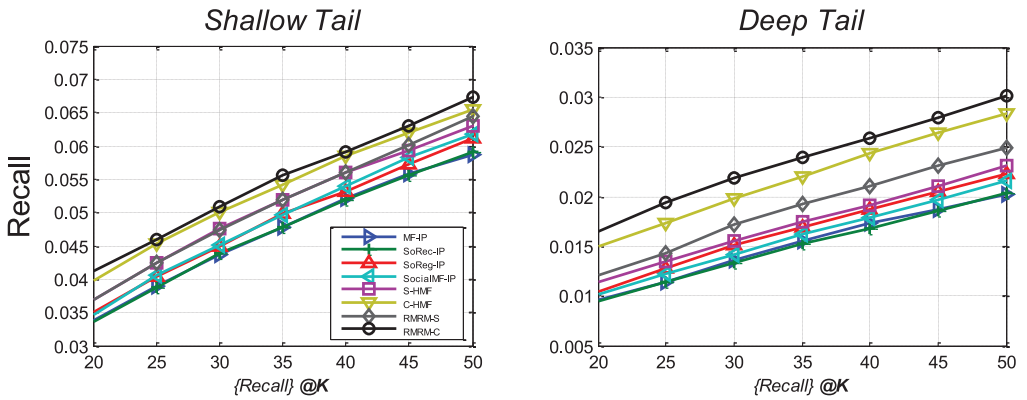

Fig. 13. Recall@20∼50 of tail-item recommendations for users.

proves that RMRM-C can better capture users' special preferences and provide more robust protection against shilling attack.

6.4.3. Evaluation of Tail-Users' Recommendations. For a company, finding an accurate list of potential users to deliver the information about their apps can reduce large promotion costs. In this experiment, as before, we randomly hold out 20% of the observations from the users as the testing set, denoted as $\boldsymbol{Ts}_{20}$, and use the remainder as the training set. In the same way as the previous experiments, we split $\boldsymbol{Ts}_{20}$ into four user groups, i.e., *Most Active*, *Less Active*, *Shallow Tail*, and *Deep Tail*, according to the number of apps that a user has installed.

Table VII reports the mean AP@10, AP@20, nDCG@10, and nDCG@20 when testing the users in each group. It is easily observed that RMRM-based models are superior to

Table VII. Mean AP@5, AP@10, nDCG@10 and nDCG@20 of User Recommendation

	Most Active				*Less Active*			
Method	**AP@10**	**AP@20**	**nDCG@10**	**nDCG@20**	**AP@10**	**AP@20**	**nDCG@10**	**nDCG@20**
MF-IR	0.0516	0.0555	0.0693	0.0834	0.0484	0.0522	0.0646	0.0787
SoRec-IR	0.0515	0.0553	0.0693	0.0832	0.0472	0.0510	0.0632	0.0768
SoReg-IR	0.0572	0.0614	0.0762	0.0917	0.0516	0.0555	0.0692	0.0835
SocialMF-IR	0.0567	0.0609	0.0762	0.0916	0.0511	0.0551	0.0690	0.0836
S-HMF	0.0608	0.0650	0.0805	0.0959	0.0554	0.0594	0.0735	0.0882
C-HMF	0.0600	0.0640	0.0790	0.0936	0.0556	0.0594	0.0734	0.0874
RMRM-S	**0.0614**	**0.0655**	**0.0820**	**0.0970**	0.0551	0.0593	0.0737	**0.0893**
RMRM-C	0.0607	0.0648	0.0800	0.0950	**0.0564**	**0.0604**	**0.0744**	0.0891

	Shallow Tail				*Deep Tail*			
Method	**AP@10**	**AP@20**	**nDCG@10**	**nDCG@20**	**AP@10**	**AP@20**	**nDCG@10**	**nDCG@20**
MF-IR	0.0414	0.0455	0.0637	0.0793	0.0313	0.0352	0.0631	0.0802
SoRec-IR	0.0419	0.0459	0.0644	0.0798	0.0312	0.0351	0.0631	0.0800
SoReg-IR	0.0459	0.0502	0.0711	0.0876	0.0347	0.0390	0.0699	0.0881
SocialMF-IR	0.0466	0.0508	0.0713	0.0876	0.0355	0.0400	0.0711	0.0899
S-HMF	0.0503	0.0546	0.0764	0.0930	0.0389	0.0433	0.0768	0.0950
C-HMF	0.0520	0.0559	0.0779	0.0943	0.0420	0.0466	0.0832	0.1025
RMRM-S	0.0500	0.0545	0.0763	0.0932	0.0384	0.0430	0.0774	0.0964
RMRM-C	**0.0522**	**0.0563**	**0.0786**	**0.0946**	**0.0421**	**0.0468**	**0.0833**	**0.1030**

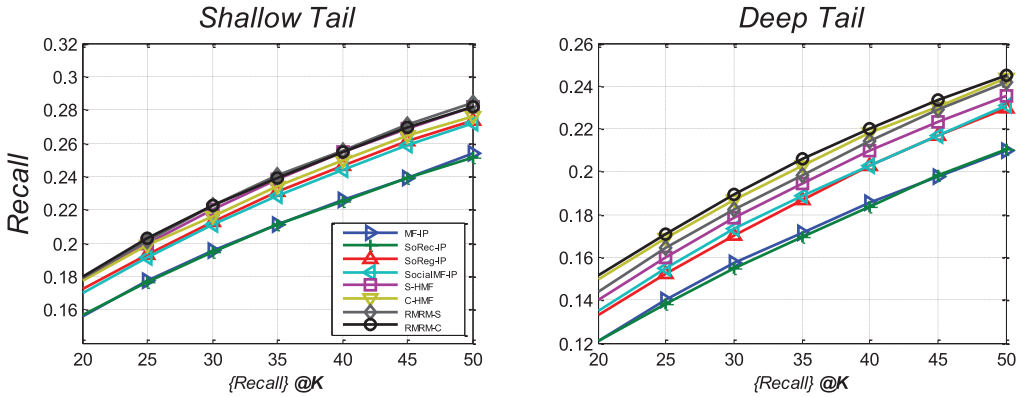

Fig. 14. Recall@20∼50 of item recommendations for tail users.

the other models, and that RMRM-S achieves the best performance in the case of *Most Active* while RMRM-C shows its advantage in the other cases. Both heteroscedasticity modeling on credibility and regularization with coupled empirical priors enable RMRM-C to capture the preferences of tail users more precisely, thus RMRM-C more effectively recommends attractive apps to tail users.

Figure 14 shows the recall@20∼50 curves of all of the compared models when recommending items for tail users (*Shallow Tail* and *Deep Tail*). Similar to the prediction performance for tail users, as shown in Table VII, the recall curves of the RMRM approach are above those of other models, which proves that the features learned from RMRM can more reliably represent the traits of items and the personal preferences of users.

7. CONCLUSION

In this paper, we address the challenges of improving the recommendations of items and for users in long-tail distributions, and analyze the ineffectiveness of current approaches. As a result, we propose RMRM, which consists of two coupled components, namely, C-HMF which emphasizes the credibility of ratings and S-HMF which emphasizes the specialty of choices, where the parameters of C-HMF and S-HMF are regularized in terms of the empirical priors induced from each other. The empirical evaluations of two real-world datasets illustrate that RMRM is capable of conducting more reliable predictions than the other compared methods, especially for both items and users in the tail of distributions.

In fact, RMRM provides a general framework for learning latent features that are regularized by multi-objective empirical priors. Therefore, RMRM and its extension could be applied in many areas outside of recommender systems, such as computer vision, audio processing, and multimedia clustering, all of which depend largely on the MF technique and thus could benefit from multi-objective regularization.

REFERENCES

E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Morup. 2010. Scalable tensor factorizations with missing data. In *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 701–712.

D. Agarwal and B. C. Chen. 2009. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, 19–28.

C. Anderson. 2006. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hachette Digital, Inc.

Y. Bengio, A. Courville, and P. Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 1798–1828.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*. ACM, New York, 89–96.

R. Burke, M. P. O'Mahony, and N. J. Hurley. 2015. Robust collaborative recommendation. In *Recommender Systems Handbook*, F. Ricci, L. Rokach, and B. Shapira (Eds.). Springer, Boston, MA, 961–995.

L. Candillier, F. Meyer, and F. Fessant. 2008. Designing specific weighted similarity measures to improve collaborative filtering systems. In *Advances in Data Mining. Medical Applications, E-Commerce, Marketing, and Theoretical Aspects.* Springer, 242–255.

K. Deb. 2014. Multi-objective optimization. In *Search Methodologies.* Springer, 403–449.

K. Georgiev and P. Nakov. 2013. A non-iid framework for collaborative filtering with restricted Boltzmann machines. In *Proceedings of the 30th International Conference on Machine Learning*. JMLR: W&CP.

J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22, 5–53.

G. E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation* 14, 1771–1800.

L. Hu, W. Cao, J. Cao, G. Xu, L. Cao, and Z. Hu. 2014. Bayesian heteroskedastic choice modeling on non-identically distributed linkages. In *Proceedings of the 2014 IEEE International Conference on Data Mining (ICDM)*. 851–856.

Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*. 263–272.

M. Jamali and M. Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, New York, 135–142.

A. Jøsang, G. Guo, M. Pini, F. Santini, and Y. Xu. 2013. Combining recommender and reputation systems to produce better online advice. In *Modeling Decisions for Artificial Intelligence*, V. Torra, Y. Narukawa, G. Navarro-Arribas, and D. Meg (Eds.). Springer, Berlin, 126–138.

A. Jøsang and R. Ismail. 2002. The beta reputation system. In *Proceedings of the 15th Bled Electronic Commerce Conference*. 41–55.

A. Jøsang, X. Luo, and X. Chen. 2008. Continuous ratings in discrete Bayesian reputation systems. In *Trust Management II.* Springer, 151–166.

A. Jøsang and W. Quattrociocchi. 2009. Advanced features in Bayesian reputation systems. In *Trust, Privacy and Security in Digital Business*, S. Fischer-Hübner, C. Lambrinoudakis, and G. Pernul (Eds.). Springer, Berlin, 105–114.

Y. D. Kim and S. Choi. 2013. Scalable variational Bayesian matrix factorization In *Proceedings of the 1st Workshop on Large-Scale Recommender Systems*.

Y. Koren. 2010. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data* 4, 1–24.

Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 30–37.

M. Levy and K. Bosteels. 2010. Music recommendation and the long tail. In *Proceedings of the 1st Workshop on Music Recommendation and Discovery (WOMRAD)*. ACM RecSys.

Y. J. Lim and Y. W. Teh. 2007. Variational Bayesian approach to movie rating prediction. In *Proceedings of the KDD Cup and Workshop*. ACM, New York, 15–27.

H. Ma, H. Yang, M. R. Lyu, and I. King. 2008. Sorec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. ACM, New York, 1458205, 931–940.

H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. 2011. Recommender systems with social regularization. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*. ACM, New York, 1935877, 287–296.

J. McAuley, R. Pandey, and J. Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, 2783381, 785–794.

S. Meyffret, E. Guillot, L. Médini, and F. Laforest. 2012. RED: A Rich Epinions Dataset for Recommender Systems. Research Report. LIRIS.

R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. 2008. One-class collaborative filtering. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE, 502–511.

Y. J. Park and A. Tuzhilin. 2008. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM Conference on Recommender Systems*. ACM, New York, 1454012, 11–18.

I. Porteous, A. U. Asuncion, and M. Welling. 2010. Bayesian matrix factorization with side information and Dirichlet process mixtures. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI'10)*. AAAI Press, Atlanta, Georgia, USA.

S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 1795167, 452–461.

R. Salakhutdinov and A. Mnih. 2008a. Bayesian probabilistic matrix factorization using Markov-chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, New York, 1390267, 880–887.

R. Salakhutdinov and A. Mnih. 2008b. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems* (2008b), 1257–1264.

R. Salakhutdinov, A. Mnih, and G. Hinton. 2007. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*. ACM, New York, 1273596, 791–798.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*. ACM, New York, 372071, 285–295.

H. Shan and A. Banerjee. 2010. Generalized probabilistic matrix factorizations for collaborative filtering. In *Proceedings of the 2010 IEEE 10th International Conference on Data Mining (ICDM)*. 1025–1030.

N. Srebro and T. Jaakkola. 2003. Weighted low-rank approximations. In *Proceedings of the 20th International Conference on Machine Learning*. 720.

N. Srebro, J. D. M. Rennie, and T. Jaakkola. 2005. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems* (2005), 1329–1336.

X. Su and T. M. Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence 2009*, 19.

K. Train. 2003. *Discrete Choice Methods with Simulation*. Cambridge University Press.

S. Vargas and P. Castells. 2011. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the 5th ACM Conference on Recommender Systems* (2011). ACM, New York, 109–116.

M. Welling and G. Hinton. 2002. A new learning algorithm for mean field boltzmann machines. In *Artificial Neural Networks—Icann 2002*, J. Dorronsoro (Ed.). Springer, Berlin, 351–357.

C. K. Williams and F. V. Agakov. 2002. Products of gaussians and probabilistic minor component analysis. *Neural Computation* 14, 1169–1182.

S. H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha. 2011. Like like alike: Joint friendship and interest propagation in social networks. In *Proceedings of the 20th International Conference on World Wide Web*. ACM, New York, 1963481, 537–546.

H. Yin, B. Cui, J. Li, J. Yao, and C. Chen. 2012. Challenging the long tail recommendation. *Proc. VLDB Endow.* 5, 896–907.

Received January 2016; revised August 2016; accepted December 2016