MFPR: A Personalized Ranking Recommendation with Multiple Feedback

CHUAN SHI, JIAN LIU, YIDING ZHANG, and BINBIN HU, Beijing University of Posts and Telecommunications
SHENGHUA LIU, Institute of Computing Technology, Chinese Academy of Sciences
PHILIP S. YU, University of Illinois at Chicago

Recently, recommender systems have played an important role in improving web user experiences and increasing profits. Recommender systems exploit users' behavioral history (i.e., feedback on items) to build models. The feedback usually includes explicit feedback (e.g., ratings) and implicit feedback (e.g., browsing history, click logs), which are both useful for improving recommendations. However, as far as we are concerned, no existing works have integrated both explicit and multiple implicit feedback simultaneously. Therefore, we propose a unified and flexible model, named Multiple Feedback-based Personalized Ranking (MFPR), to make full use of multiple feedback, which uses a personalized ranking framework. To train model MFPR, we design an algorithm to generate ordered item pairs as labeled data, with consideration of both rating scores and multiple implicit feedback. Extensive experiments on two real-world datasets validate the effectiveness of the MFPR model. With the integration of multiple feedback, MFPR significantly improves recommendation performance.

CCS Concepts: • **Computing methodologies → Machine learning**; **Machine learning algorithms**;

Additional Key Words and Phrases: Recommender system, multiple feedback, bayesian personalized ranking

ACM Reference format:
Chuan Shi, Jian Liu, Yiding Zhang, Binbin Hu, Shenghua Liu, and Philip S. Yu. 2018. MFPR: A Personalized Ranking Recommendation with Multiple Feedback. *ACM Trans. Soc. Comput.* 1, 2, Article 7 (June 2018), 22 pages.
https://doi.org/10.1145/3216368

1 INTRODUCTION

To alleviate the information overload problem, recommender systems have been proposed to help users find items of interest through utilizing the user-item interaction in formation and/or content

This work is supported in part by the National Natural Science Foundation of China (No. 61772082, No. 61375058, and No. 61772498), the National Key Research and Development Program of China (No. 2017YFB0803304), and the Beijing Municipal Natural Science Foundation (No. 4182043 and No. 4172059). This work is supported in part by NSF through grants IIS-1526499, IIS-1763325, CNS-1626432, and NSFC 61672313.
Authors' addresses: C. Shi, J. Liu, Y. Zhang, and B. Hu, Beijing Key Lab of Intelligent Telecommunication Software and Multimedia, Beijing University of Posts and Telecommunications, No. 10 Xi Tu Cheng Road, Beijing, China; emails: shichuan@bupt.edu.cn, fullback@yeah.net, zyd@bupt.edu.cn, and hubinbin@bupt.edu.cn; S. Liu (corresponding author), CAS Key Laboratory of Network Data Science & Technology, Institute of Computing Technology Chinese Academy of Sciences, No.6 Kexueyuan South Road Zhongguancun, Beijing, China; email: liushenghua@ict.ac.cn; P. S. Yu, Department of Computer Science, University of Illinois at Chicago, 851 South Morgan Street, Chicago, USA; email: psyu@uic.edu.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 ACM 2469-7818/2018/06-ART7 $15.00
https://doi.org/10.1145/3216368

information associated with users and items. Recommender systems have attracted much attention from multiple disciplines, and many techniques have been proposed to build recommender systems. The interaction information (i.e., feedback) between users and items is widely exploited to build recommendation models.

The feedback data in recommender systems usually come in the form of explicit or implicit feedback [10]. Explicit feedback is the interaction information that directly expresses user preferences for items, such as the rating information. Implicit feedback is the interaction information that indirectly reflects users' opinions and can imply user preferences [20]. Figure 1 shows an example of multiple feedback in Douban Book. The rating (1–5 scores) is the explicit feedback that directly reflects user preferences. There are two types of implicit feedback, which also imply user preference. The term "wish" means that the user wishes to read the book but has not begun yet; "reading" means reading the book. We can see that explicit feedback (i.e., rating) quantifies users' preferences, which is critical for recommendation, while implicit feedback is also an important complement. We know that in real applications, explicit feedback is usually scarce, but implicit feedback is usually abundant. Although one kind of implicit feedback may be weak and indirectly relfect user preferences, the aggregation of this type of feedback provides important hints about user preferences.

Many methods exploit feedback information to build recommender models. Figure 2 shows how these methods utilize this information. As shown in Figure 2(a), traditional collaborative filtering usually utilizes explicit feedback information (i.e., ratings) [11, 14, 26]. Since implicit feedback information is widely and cheaply available, some studies began to use implicit feedback in recent years. Some works considered using a single type of implicit feedback [12, 21, 25] (see Figure 2(b)), and Fortes and Manzato [5] began to combine several types of implicit feedback with a simple ensemble approach (see Figure 2(c)). In addition, SVD++ [14] is designed to combine rating information and only one type of implicit feedback for improving rating prediction, as shown in Figure 2(d). Unfortunately, all these works do not simultaneously utilize comprehensive feedback information in recommender systems.

In this article, we propose to solve the personalized ranking problem by integrating multiple feedback, as shown in Figure 2(e). For convenience, multiple feedback means one type of explicit feedback and multiple types of implicit feedback in the following sections. In many review web sites, such as Yelp and Dianping, users are required to give a rating score (i.e., explicit feedback) to an object, and they can also have other interactions with objects, such as "checking in" and "viewing." Obviously, our problem setting is a general framework for utilizing feedback information, and existing problems are special cases of our problem setting. In addition, many recommendation algorithms predict users' rating scores of items and then calculate the RMSE criteria between predicted and true values to evaluate their effectiveness (e.g., SVD, SVD++). Actually, the rating is just one way to express user preferences. Users are usually more concerned with the order than the rating score of items. Therefore, from the recommendation perspective, predicting the rank of an item is more straightforward and meaningful than predict rating scores. Thus, in this work, we focus on developing a personalized ranking model that integrates multiple feedback. Although many methods have been proposed to utilize feedback, these models are usually designed for special problem settings, and they cannot be directly applied in a multiple-feedback setting.

Integration of multiple feedback faces two challenges. (1) Design a unified ranking model integrating multiple feedback. To make the best use of this feedback information, we need to design an effective mechanism to handle relations between explicit and implicit feedback as well as relations among implicit feedback. (2) Generate training samples. As a ranking method, we need to generate preference pairs or lists for training. However, there are multiple types of feedback. It is not a trivial task to utilize this feedback to generate the training data.

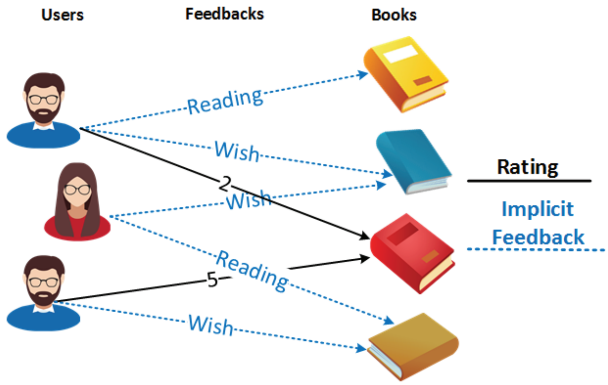


Fig. 1. An example of multiple feedback between users and books in Douban Book.

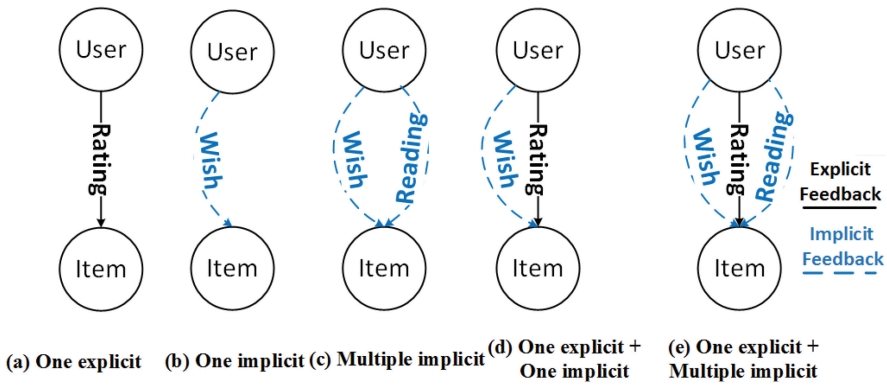


Fig. 2. The schemas of utilizing feedback information.

We first study the personalized ranking recommendation problem integrating multiple feedback and propose a Multiple Feedback based Personalized Ranking (MFPR) recommendation model. We integrate the explicit feedback with one type of implicit feedback using the Bayesian Personalized Ranking framework and then extend this model to integrate more implicit feedback. In addition, a generation algorithm of training samples is proposed, which can effectively uncover the truth ranking information of items contained in feedback information. The major contributions of our article are summarized as follows:

- We first try to solve the personalized ranking recommendation problem by integrating multiple feedback. The problem widely exists in a real recommender system, and it is a general problem setting to encompass existing works.
- We propose a Bayesian Personalized Ranking (BPR) based model MFPR to integrate multiple feedback. Moreover, as there are no readily available training data of pairwise comparisons for this problem, an effective algorithm is designed to generate the training data that are more consistent with multiple feedback for the MFPR model.
- We crawl comprehensive Douban Book and Dianping datasets[1] including ratings and multiple types of implicit feedback. Extensive experiments on these two real datasets validate the effectiveness of the proposed method.

[1]The datasets are available at https://github.com/7thsword/MFPR-Datasets.

The preliminary work was published in Reference [17]. However, this article substantially extends the original work in the following aspects. First, it introduces in detail one important contribution of this article, the training set generation algorithm where *I*tem *P*airs with partial order are obtained from checking adjacent items in a *P*ermutation of an *E*xplicit item set (IPPE), and gives insightful analysis of IPPE in Section 5. Moreover, we further validate its effectiveness in Section 6.5. Second, it adds extensive experiments to sufficiently validate the traits of the proposed MFPR. This includes the following: "Integrate Different Implicit Feedback with Rating" in Section 6.6, "Mean weighted SFPR versus MFPR" in Section 6.7, and "Parameter Study" in Section 6.8. Moreover, it adds another baseline MSVD++ in Section 6.3. These approaches indicate broader use of the proposed method, in addition to its advantages. Third, it provides a clearer description of the proposed method, including the learning algorithm, algorithm framework, and complexity analysis of the MFPR in Section 4.3. In addition, it describes related works in more detail in Section 2 and provides an introduction to the basic model SVD in Section 3.3.

The remainder of this article is organized as follows: We describe the related works in Section 2, and Section 3 presents preliminary knowledge and problem formulation. We introduce the proposed model in Section 4 and then detail the novel training set generation algorithm in Section 5. Experiments and analysis are shown in Section 6. Finally, we conclude the article in Section 7.

2 RELATED WORK

Rating prediction methods are a popular type of recommendation technique. The task is to predict the unknown user-item ratings by minimizing the error of predicted ratings and true ratings in training data. Traditional collaborative filtering (CF) is one of the most popular techniques for rating prediction, including user-based CF and item-based CF. Recently, a series of matrix factorization models [19, 26, 27, 29] showed their power in rating prediction, which mainly factorized a known but incomplete user-item rating matrix into two low-rank user-specific and item-specific matrices. Then, the factorized matrices were used to predict ratings.

For the past number of years, many techniques have been proposed for building recommender systems. According to the input data, these techniques can be roughly classified into three categories: explicit feedback based, implicit feedback based, and hybrid feedback based. Explicit feedback is usually considered more reliable and of high quality. A series of matrix factorization models exploiting explicit feedback show their potential in recommender systems, such as Probabilistic Matrix Factorization (PMF) [26], Singular Value Decomposition (SVD) [14], Non-negative matrix factorization (NMF) [11] and Hete-MF [29].

Since implicit feedback is often easily available, many methods using implicit feedback have been proposed. For example, BPRMF [25] utilized implicit feedback to generate training pairs and then learned the parameters in the BPR model; Fortes and Manzato [5] ensembled several BPRMF and each BPRMF instance utilized one type of the multiple implicit feedback; Gurbanov et al. [9] presented the model MMF that predicted a target user action by leveraging actions of multiple types. Essentially, it utilized multiple types of implicit feedback to predict a target implicit feedback.

In addition, some researchers began to exploit hybrid feedback. For example, Koren [14] designed the SVD++ to combine ratings with single implicit feedback (i.e., whether a user rated an item) for predicting ratings more accurately. Fortes and Manzato [4] developed a hybrid model for personalized ranking that uses SVD to handle explicit feedback and BPRMF to handle implicit feedback (i.e., whether a user tagged an item). Tang et al. [28] conducted a series of experiments to explore how to reasonably integrate user positive and negative implicit feedback for improving the CTR of the news feed and email campaign of the Linkedin system. Gurbanov et al. [8] proposed

a recommender system integrating sequence mining and CF models to predict whether a user will perform an action of a target type on an item. The above methods usually utilized some feedback information, while our method makes full use of implicit and explicit feedback information.

Multi-label classification methods have also been applied in recommendation. For example, Agrawal et al. [1] proposed an algorithm that used multi-label random forests as classifier and recommend bid phrases from a given ad landing page. Oliveira et al. [6] used multi-label k-nearset neighbor as classifier and recommend programming activities. The "label" in these methods usually represents characteristics of items or users, while the "feedback" in our work embodies interactions between users and items. They are two different types of signals in recommender systems and lead to different analysis methods. So multi-label classification methods cannot be directly applied to our problem.

Recently, learning to rank (LTR) [16, 18, 25] has attracted increasing attention in the machine learning community. LTR is the core technology for ranking tasks, such as document ranking in information retrieval. Such techniques began to be applied to the personalized recommendation in recent years. There are many LTR methods, and they can be classified into three categories: pointwise, pairwise, and listwise. In pointwise methods, the model learns to output a score or class label for each input single document. Specifically, the rating prediction models can be considered as a kind of pointwise method. In listwise methods [3, 22], the model learns to output a ranked document list for the input document collections; in pairwise methods [13], the model focuses on learning with a preference for each input document pair, where Bayesian Personalized Ranking [25] is a typical approach.

3 PRELIMINARY

In this section, we introduce some basic concepts, the problem formulation and the base model.

3.1 Explicit and Implicit Feedback

In real recommender systems, feedback information is prevalent between users and items. Feedback data can be divided into two categories: explicit and implicit. Formally, when feedback data are in the form of explicit feedback with single implicit feedback, each user u is associated with two types of item sets: an explicit feedback set $E(u)$ and implicit item set $N(u)$.

Explicit feedback is intentionally provided by users to directly express user preferences (e.g., like or dislike) for items. For example, user ratings are one of the most popular types of explicit feedback. For an item $i \in E(u)$, user u has given the rating R_{ui} to item i. The rating R_{ui} is usually an integer between 1 and 5, indicating the preference of user u for item i. Higher ratings indicate a stronger preference. Explicit feedback is very important for recommender systems. Traditional collaborative filtering methods are usually based on explicit feedback. However, this kind of feedback is usually difficult to collect, since many users are not willing to give ratings to items.

Implicit feedback reflects user opinions indirectly and can imply user probable preferences [20]. For example, in music recommender systems, users may "collect," "download," and "share" songs. For an item $i \in N(u)$, the implicit feedback does not necessarily mean that user u likes the item i. In turn, for an item $i \notin N(u)$, the implicit feedback does not mean that a user dislikes item i. Implicit feedback just provides indication of possible user preferences. For instance, if a user adds a song to a playlist, she may know it through her friends but not have heard it yet, which only indicates that she may like it. Implicit feedback widely exists in recommender systems. These types of data are huge in real systems, since there are many ways to interact with items through these systems, and this interaction with items can be converted into implicit feedback.

When feedback data consist of explicit feedback with multiple types of implicit feedback, each user is associated with a single explicit feedback and τ types of implicit feedback ($\tau \geq 2$). For user

u, the explicit item set still denoted as $E(u)$ contains items user u has rated (i.e., a rating), and the implicit item sets denoted as $N^1(u), N^2(u), \ldots, N^\tau(u)$, where $N^t(u)$, contains items about which user u has expressed the t-type implicit feedback ($t = 1, \ldots, \tau$).

3.2 Problem Formulation

Let $\mathcal{U}$ and $\mathcal{I}$ denote the set of users and items, respectively. We define a ranking recommendation problem on multiple feedback data $R_d = \{\mathcal{U}, \mathcal{I}, E_f, I_f\}$. E_f, defined as $E_f = \{E(u)|u \in \mathcal{U}\}$, to be explicit feedback data consisting of all users' explicit item sets. I_f, defined as $I_f = \{N^t(u)|u \in \mathcal{U}, t = 1, \ldots, \tau\}$, is implicit feedback data consisting of all users' implicit item sets. Hence, as shown in Figure 2(e), our task is to design a model to make full use of the explicit feedback data E_f and the implicit feedback data I_f.

It is obvious that existing works usually utilize incomplete feedback data. For example, traditional collaborative filtering (e.g., SVD [14]) is based on the data $R_d = \{\mathcal{U}, \mathcal{I}, E_f\}$, the widely used SVD++ is based on the data $R_d = \{\mathcal{U}, \mathcal{I}, E_f, I_f\}$ with $\tau = 1$, and the recent work of Fortes and Manzato [5] only considered data $R_d = \{\mathcal{U}, \mathcal{I}, I_f\}$. Thus, our problem setting is a general framework that includes the existing problem setting as a special case. Our problem setting is very popular in the real world.

3.3 Base Learner Integrating Explicit and Implicit Feedback

Some effective learners have been proposed to utilize feedback data. Assume that there are m users and n items (i.e., $|\mathcal{U}| = m$, $|\mathcal{I}| = n$). Given a rating matrix $R = (R_{ui})^{m \times n}$, R_{ui} denotes the score user u has rated on item i. A classical factorization model [14] is induced by an SVD-like low-rank matrix factorization. Each user u and item i are represented by latent vectors $p_u \in \mathbb{R}^d$ and $q_i \in \mathbb{R}^d$, respectively ($d \ll min(m, n)$). Rating prediction for item i by user u can be modeled as follows:

$$\hat{R}_{ui} = p_u q_i^{\mathrm{T}}. \tag{1}$$

In Reference [14], Koren et al. proposed a factorization model called SVD++, considering the integration of explicit and implicit feedback to predict ratings more accurately. The predicted rating $\hat{R}_{ui}$ user u may give to item i can be modeled as

$$\hat{R}_{ui} = \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{k \in N(u)} \gamma_k \right) q_i^{\mathrm{T}}, \tag{2}$$

where $\gamma_k \in \mathbb{R}^d$ is the implicit latent vector of item k and $N(u)$ is the implicit item set as mentioned above. It is worth noting that Equation (2) does not contain the bias and average component. As we use pairwise training data, the user bias and average component are eliminated. The details are described in Section 4.2. Now, a user u is modeled as $p_u + |N(u)|^{-\frac{1}{2}} \sum_{k \in N(u)} \gamma_k$, and the complemented sum term $|N(u)|^{-\frac{1}{2}} \sum_{k \in N(u)} \gamma_k$ represents the perspective of implicit feedback. SVD++ models implicit feedback as a part of the user factor, which is a straightforward but effective method. It makes the best use of explicit feedback and adds implicit feedback as supplements.

Unfortunately, these existing models cannot be directly applied to our problem setting. Although SVD++ also considers explicit and implicit feedback, it just integrates one type of implicit feedback. In addition, SVD++ is originally designed for the rating prediction problem. Since predicting exact ratings is not necessary for many recommendation applications, we propose using a ranking framework.

4 PERSONALIZED RANKING WITH MULTIPLE FEEDBACK

It is not a trivial task to design a unified ranking model integrating multiple feedback. The explicit and implicit feedback have different characteristics, and we must therefore treat them differently. In addition, we also need to integrate multiple implicit feedback. As analyzed above, existing models cannot be directly applied to this problem. A naive method is to treat all feedback as feature vectors. However, as shown in Section 6.3, the FM method [23] does not achieve good performance, because the features of explicit feedback are ignored.

In this article, we propose a unified MFPR. By adapting the SVD++ model with the Bayesian Personalized Ranking, we first design a *P*ersonalized *R*anking model integrating explicit feedback with one-*S*ingle implicit *F*eedback (called SFPR). And we then extend the SFPR model by integrating multi-type implicit feedback. In this section, we first present the SFPR model and its learning method and then put out the MFPR model.

4.1 The SFPR Model

First, we design a ranking model to combine explicit feedback with one type of implicit feedback. Here, we extend the Bayesian Personalized Ranking (BPR) framework [25] originally designed for handling single implicit feedback to integrate explicit with implicit feedback. Assume that a training set $\mathcal{T}_r$ consists of triples of the form (u, i, j) with $i \succ j$ denoting that user u shows more preference on item i than item j. Note that the generation of training set $\mathcal{T}_r$ is an important issue that will be discussed in Section 5. The Bayesian formulation of finding the correct personalized ranking is to maximize the following posterior probability:

$$p(\theta|\mathcal{T}_r) \propto p(\mathcal{T}_r|\theta)p(\theta), \tag{3}$$

where θ is the parameter of a certain base learner and $p(\theta)$ is the prior probability of a base learner parameter.

We use $p(i \succ j; u|\theta)$ to denote the probability that user u prefers item i over item j. With the assumption that each triple $(u, i, j) \in \mathcal{T}_r$ is independent, the likelihood function can be expanded as follows:

$$p(\mathcal{T}_r|\theta) = \prod_{(u,i,j)\in\mathcal{T}_r} p(i \succ j; u|\theta). \tag{4}$$

To integrate single explicit feedback with single implicit feedback, we choose SVD++ in Equation (2) as our base learner; SVD++ effectively differentiates explicit and implicit feedback, and it fully utilizes the explicit feedback. Then the individual probability that a user really prefers item i over item j can be designed as:

$$p(i \succ j; u|\theta) = \sigma(\hat{R}_{ui} - \hat{R}_{uj}), \tag{5}$$

where σ is the logistic sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$.

For convenience, we simplify $\hat{R}_{ui} - \hat{R}_{uj}$ in Equation (5) as $\hat{x}_{uij}$. Note that $\hat{x}_{uij}$ is a real-valued function of θ that captures the ranking relation between item i and item j with the given user u. Assume that $p(\theta)$ is a Gaussian distribution with zero mean and variance-covariance matrix $\sum_\theta = \lambda_\theta I$. Now we can estimate parameter θ of the base learner by maximizing the posterior

probability in Equation (3) as follows:

$$\begin{aligned}\max_{\theta} \mathcal{L} &= ln\, p(\theta|\mathcal{T}_r) \\ &= ln\, p(\mathcal{T}_r|\theta)p(\theta) \\ &= \sum_{(u,i,j)\in\mathcal{T}_r} ln\, p(i \succ j; u|\theta) - \lambda_\theta \|\theta\|^2 \\ &= \sum_{(u,i,j)\in\mathcal{T}_r} ln\, \sigma(\hat{x}_{uij}) - \lambda_\theta \|\theta\|^2, \end{aligned} \tag{6}$$

where $\lambda_\theta \|\theta\|^2$ is an L2 regularization term that can be derived from the Gaussian distribution $p(\theta)$ mentioned above.

4.2 Learning Algorithm of SFPR Model

Note that the objective function in Equation (6) is differentiable, and we can employ gradient ascent-based algorithms as the optimizer. The gradient of Equation (6) with respect to the parameter θ is

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \theta} &= \sum_{(u,i,j)\in\mathcal{T}_r} \frac{\partial}{\partial \theta} ln\sigma(\hat{x}_{uij}) - \lambda_\theta \frac{\partial}{\partial \theta}\|\theta\|^2 \\ &\propto \sum_{(u,i,j)\in\mathcal{T}_r} \frac{1}{1+e^{\hat{x}_{uij}}} \frac{\partial}{\partial \theta}\hat{x}_{uij} - \lambda_\theta \theta. \end{aligned} \tag{7}$$

In the article, we apply stochastic gradient ascent (SGA) to optimize the model SFPR. Then with a training sample (u, i, j), the model parameter θ can be updated as

$$\theta \leftarrow \theta + \eta \left(\frac{1}{1+e^{\hat{x}_{uij}}} \frac{\partial}{\partial \theta}\hat{x}_{uij} - \lambda_\theta \theta \right), \tag{8}$$

where η is the given learning rate. The gradient of $\hat{x}_{uij}$ with respect to each model parameter has to be known before the gradient ascent process. As defined above, we can get the $\hat{x}_{uij} = \hat{R}_{ui} - \hat{R}_{uj}$ as

$$\hat{x}_{uij} = \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{k\in N(u)} \gamma_k \right) (q_i - q_j)^{\mathrm{T}}. \tag{9}$$

The model parameters in Equation (9) are p_u, q_i, q_j, and $|N(u)|$, while $|N(u)|$ is the length of the implicit item set of user u that is fixed, so we can get the derivatives of other parameters as

$$\frac{\partial \hat{x}_{uij}}{\partial \theta} = \begin{cases} q_i - q_j & \text{if } \theta = p_u, \\ p_u + |N(u)|^{-\frac{1}{2}} \sum_{k\in N(u)} \gamma_k & \text{if } \theta = q_i, \\ -(p_u + |N(u)|^{-\frac{1}{2}} \sum_{k\in N(u)} \gamma_k) & \text{if } \theta = q_j, \\ |N(u)|^{-\frac{1}{2}}(q_i - q_j) & \text{if } \theta = \gamma_k. \end{cases} \tag{10}$$

We set regularization parameters λ_p, λ_q, and λ_γ for user explicit latent vectors, item explicit latent vectors and item implicit latent vectors, respectively. Referring to Equation (8), we define $\Delta_{uij} = \frac{1}{1+e^{\hat{x}_{uij}}}$ and for any sample $(u, i, j) \in \mathcal{T}_r$, parameters of SFPR can be updated using SGA:

$$p_u \leftarrow p_u + \eta(\Delta_{uij}(q_i - q_j) - \lambda_p p_u), \tag{11}$$

$$q_i \leftarrow q_i + \eta \left(\Delta_{uij} \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{k\in N(u)} \gamma_k \right) - \lambda_q q_i \right), \tag{12}$$

$$q_j \leftarrow q_j + \eta\left(-\Delta_{uij}\left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{k \in N(u)} \gamma_k\right) - \lambda_q q_j\right), \tag{13}$$

$$\begin{aligned} &for\ k \in N(u): \\ &\quad \gamma_k \leftarrow \gamma_k + \eta\Big(|N(u)|^{-\frac{1}{2}} \Delta_{uij}(q_i - q_j) - \lambda_\gamma \gamma_k\Big). \end{aligned} \tag{14}$$

It is noteworthy that when using the trained SFPR to do prediction, the $\hat{R}_{ui}$ cannot be regarded as the predicted rating (i.e., 1 to 5 scores) as usual. Here, we call $\hat{R}_{ui}$ the predicted ranking score, which implies thr degree that user u prefers to item i. Higher scores indicate a stronger user preference.

4.3 The MFPR Model

The proposed SFPR is designed to integrate single explicit feedback and single implicit feedback. Then, we extend the SFPR model to integrate more implicit feedback. When considering multiple feedback, as mentioned in Section 3.1, each user u is associated with an explicit item set $E(u)$ and τ types of implicit item sets $N^1(u), N^2(u), \ldots, N^\tau(u)$. For integrating multiple implicit feedback, our extended preference predictor can be designed as

$$\hat{R}_{ui} = \left(p_u + \frac{1}{\tau}\sum_{t=1}^{\tau} |N^t(u)|^{-\frac{1}{2}} \sum_{k \in N^t(u)} \gamma_k^t\right) q_i^{\mathrm{T}}, \tag{15}$$

where $\gamma_k^t \in \mathbb{R}^d$ represents the implicit latent vector of item k under the t-th implicit feedback. The model in Equation (15) can be seen as a more general version of the SFPR model.

Now we have the $\hat{x}_{uij} = \hat{R}_{ui} - \hat{R}_{uj}$ as

$$\hat{x}_{uij} = \left(p_u + \frac{1}{\tau}\sum_{t=1}^{\tau} |N^t(u)|^{-\frac{1}{2}} \sum_{k \in N^t(u)} \gamma_k^t\right)(q_i - q_j)^{\mathrm{T}}. \tag{16}$$

With $\Delta_{uij}, \lambda_p, \lambda_q, \lambda_\gamma$ defined as previously, for any sample $(u, i, j) \in \mathcal{T}_r$ yielding:

$$\frac{\partial}{\partial\theta}\hat{x}_{uij} = \begin{cases} q_i - q_j & \text{if } \theta = p_u, \\ p_u + \frac{1}{T}\sum_{t=1}^{\tau} |N^t(u)|^{-\frac{1}{2}} \sum_{k \in N^t(u)} \gamma_k^t & \text{if } \theta = q_i, \\ -(p_u + \frac{1}{T}\sum_{t=1}^{\tau} |N^t(u)|^{-\frac{1}{2}} \sum_{k \in N^t(u)} \gamma_k^t) & \text{if } \theta = q_j, \\ |N^t(u)|^{-\frac{1}{2}}(q_i - q_j) & \text{if } \theta = \gamma_k^t. \end{cases} \tag{17}$$

Similarly, we apply SGA to solve the optimization problem. The whole algorithm framework is shown in Algorithm 1. The time complexity of MFPR can be analyzed as follows. The computation of MFPR mainly contains two parts: (1) calculating parameters and gradients (Lines 6, 7, and 13) and (2) updating parameters (Lines 8–10 and 14). The number of latent dimensions is d, and $|N^t(u)|$ can be estimated by a small constant c and $c \ll m$, $c \ll n$. The complexity in Lines 6 and 7 is $O(c \times \tau \times d)$. The complexity in Line 13 is $O(c \times \tau \times d)$. And the complexity in Lines 8–10 and 14 is $O(d)$. So, the entire complexity of Lines 11–16 is $O(c^2 \times \tau^2 \times d)$. In summary, the complexity of MFPR is $O(c^2 \times \tau^2 \times d \times |\mathcal{T}_r| \times r)$, where r is the number of iterations.

5 TRAINING SET GENERATION ALGORITHM

As mentioned above, our MFPR model is fed with training data in the form of (u, i, j) with $i > j$ denoting that user u prefers item i over item j. There is an important issue of how we can effectively generate (u, i, j) from multiple feedback, since the preference partial pairs significantly affect performances [2]. For those traditional personalized ranking models utilizing only one or more types of implicit feedback, such as BPRMF in Reference [25] and the approach in Reference [5], their

ALGORITHM 1: Algorithm Framework of MFPR

Input: $\mathcal{T}_r$: the training set of training triples
η: learning rate for gradient ascent
$\lambda_p, \lambda_q, \lambda_\gamma$: regularization parameters defined above
Output: $p_u (u = 1 \cdots m)$: the explicit latent vector of user u
$q_i (i = 1 \cdots n)$: the explicit latent vector of item i
$\gamma_k (k = 1 \cdots n)$: the implicit latent vector of item k

Initialize $p_u, q_i, \gamma_k^t (t = 1 \cdots n)$ for all users and all items
Define $\hat{x}_{uij} = \hat{R}_{ui} - \hat{R}_{uj}$ with Equation (16)
Define $\Delta_{uij} = 1/(1 + e^{\hat{x}_{uij}})$ as in Section 4.2
repeat
  for (u, i, j) *in* $\mathcal{T}_r$ **do**
    Calculate $\hat{x}_{uij}, \Delta_{uij}$
    Calculate $\frac{\partial}{\partial p_u}\hat{x}_{uij}, \frac{\partial}{\partial q_i}\hat{x}_{uij}, \frac{\partial}{\partial q_j}\hat{x}_{uij}$
    Update $p_u := p_u + \eta(\Delta_{uij}\frac{\partial}{\partial p_u}\hat{x}_{uij} - \lambda_p p_u)$
    Update $q_i := q_i + \eta(\Delta_{uij}\frac{\partial}{\partial q_i}\hat{x}_{uij} - \lambda_q q_i)$
    Update $q_j := q_j + \eta(\Delta_{uij}\frac{\partial}{\partial q_j}\hat{x}_{uij} - \lambda_q q_j)$
    for $t \leftarrow 1$ *to* τ **do**
      for $k \in N^t(u)$ **do**
        Calculate $\frac{\partial}{\partial \gamma_k^t}\hat{x}_{uij}$
        Update $\gamma_k^t := \gamma_k^t + \eta(\Delta_{uij}\frac{\partial}{\partial \gamma_k^t}\hat{x}_{uij} - \lambda_\gamma \gamma_k^t)$
      end
    end
  end
until *convergence*;

training set generation algorithms just take implicit feedback into count. Specifically, they draw partially ordered item pairs from the Cartesian product of user's interacted items (items that belong to a user's implicit item set) and a user's non-interacted items (items that do not belong to a user's implicit item set). However, in terms of multiple feedback, such a training set generation algorithm is inapplicable for MFPR. In addition to implicit feedback, there is quantified rating information in our problem setting, which can better reflect preference sequences. Hence, we need to design a new training set generation algorithm.

Burgess and Shaked et al. [2] have proved that if the ranking probabilities of every adjacent document pair in a permutation of all documents to be ranked are known, then the ranking probabilities of any document pair can be derived. Inspired by this conclusion, we design a training set generation algorithm that utilizes the most significant preference information in the multiple feedback: rating information. For each user u, we randomly split his or her explicit item set $E(u)$ into two subsets $E_{tr}(u)$ and $E_{te}(u)$ with the given split ratio, where $E_{tr}(u)$ is designed for constructing the training set $\mathcal{T}_r$ and $E_{te}(u)$ is for the testing set $\mathcal{T}_e$. When constructing $\mathcal{T}_r$, we first obtain a random permutation of $E_{tr}(u)$. Then, for every adjacent item pair (i, j) in the permutation: (1) If $R_{ui} > R_{uj}$, then put the triple (u, i, j) into $\mathcal{T}_r$; (2) if $R_{ui} < R_{uj}$, then put the triple (u, j, i) into $\mathcal{T}_r$; and (3) if $R_{ui} = R_{uj}$, then skip and continue to check next adjacent pair. Through the process for every user, we can eventually get the training set $\mathcal{T}_r$. And a similar process is applied to the testing set $\mathcal{T}_e$.

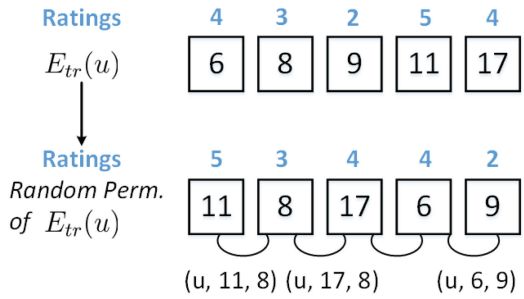


Fig. 3. Example of generating training data for user u.

Figure 3 gives an example for user u. We have $E_{tr}(u) = \{6, 8, 9, 11, 17\}$ and the corresponding ratings are $R_{u,6} = 4, R_{u,8} = 3, R_{u,9} = 2, R_{u,11} = 5$, and $R_{u,17} = 4$. Assume that a random permutation of E_{tr} is $P_{tr} = \{11, 8, 17, 6, 9\}$, and then we in turn check every adjacent item pair $(11, 8)$, $(8, 17)$, $(17, 6)$, $(6, 9)$ of the permutation. Finally, the triples $(u, 11, 8)$, $(u, 17, 8)$, and $(u, 6, 9)$ are selected and put into the training set $\mathcal{T}_r$. Algorithm 2 shows the whole algorithm framework, and we refer to this algorithm as IPPE.

Please note that we only use explicit feedback (i.e., rating information), without implicit feedback, to generate training data. Why not add implicit feedback as a supplement? There are two reasons. (1) Explicit feedback significantly reflects user preferences, while implicit feedback implies user preferences with uncertainty. It will add much noise to the training data when considering implicit feedback. (2) Referring to the update process in Algorithm 1, we can note that the implicit feedback data have been utilized implicitly in the model when we use rating-related training data. So, it is not necessary to adopt implicit feedback in generating training data. In addition, the IPPE method considers every adjacent item pair rather than any item pair. This strategy significantly reduces the size of training samples without much sacrifice in recommendation performance.

6 EXPERIMENT

In this section, we conduct a series of experiments on two real-world datasets and verify the superiority of the proposed models compared to state-of-the-art baselines.

6.1 Datasets

In this article, we focus on exploiting user multiple feedback, including explicit feedback and multiple types of implicit feedback. As far as we know, it is difficult to obtain such public datasets. Hence, we crawled two real-world datasets for the experiments.

The Douban Book dataset is crawled from Douban,[2] which is a well-known social media network in China. When crawling data, we first select some active users in an interest group as seed users, and then crawl other users followed by the seed users in the next iteration. We crawl users iteratively in the above way. At the same time, we crawl the books that the crawled users gave feedback on. Finally, a sub-network of the Douban social network is obtained for our experiments. The dataset contains 190,590 ratings (1–5 scores) from 12,850 users and 22,040 books. The ratings of users to books are considered explicit feedback. There are six types of implicit feedback: "wish," "reading," "read," "tag," "comment," and "rated." This implicit feedback is represented by a binary

[2]http://book.douban.com.

ALGORITHM 2: The Training Set Generation Algorithm IPPE

```
Input: ϵ: split ratio
       E(1), E(2), ..., E(m): user rated item sets
Output: T_r, T_e: training and testing set
Initialize T_r = ∅, T_e = ∅
for u ← 1 to m do
    Randomly split E(u) into E_tr(u) and E_te(u) using ϵ
    Get random permutation P_tr(u), P_te(u) of E_tr(u), E_te(u)
    for adjacent item i, j in P_tr(u) do
        if R_ui > R_uj then
            Put (u, i, j) into T_r
        else
            if R_ui < R_uj then
                Put (u, j, i) into T_r
            end
        end
    end
    for adjacent item i,j in P_te(u) do
        if R_ui > R_uj then
            Put (u, i, j) into T_e
        else
            if R_ui < R_uj then
                Put (u, j, i) into T_e
            end
        end
    end
end
```

matrix ("1" for done and "0" for not). Note that the "rated" implicit feedback is from rating information that has been degraded into a binary matrix ("1" means "rated" and "0" for "not rated").

The Dianping dataset is crawled from the Dianping website,[3] which is a well-known life-service social platform providing reviews of users on businesses in China. This dataset contains 188,813 ratings (1–5 scores) from 10,549 users and 17,707 restaurants. There are four types of ratings in Dianping, including overall rating (1–5 scores) and ratings (1–5 scores) on taste, environment, and service. We use the overall ratings as explicit feedback and degrade overall, taste, environment and service ratings into "1" if rating ≥ 3 and otherwise "0." Then four types of implicit feedback (0/1) are obtained: "good taste," "good environment," "good service," and "good overall." A detailed description of the two datasets can be seen in Table 1.

6.2 Evaluation Metrics

We use two evaluation metrics, which are widely used to evaluate ranking performance. *Zero-One Error* [15] is the average ratio of correctly ordered item pairs of triples (u, i, j) in testing set $\mathcal{T}_e$:

$$\varepsilon_{0/1} = \frac{1}{|\mathcal{T}_e|} \sum_{(u,i,j)\in\mathcal{T}_e} 1[\hat{x}_{uij}(R_{ui} - R_{uj}) > 0], \tag{18}$$

[3] http://www.dianping.com.

Table 1. Statistics of Datasets

Dataset	Type	A-B	#A	#B	#A-B
Douban Book	explicit	rating	12,850	22,040	190,590
	implicit	wish	11,107	16,406	162,565
		reading	9,776	12,787	71,662
		read	12,029	20,014	174,726
		tag	8,487	19,942	162,070
		comment	8,776	18,888	151,758
		rated	12,850	22,040	190,590
Dianping	explicit	rating	10,549	17,707	188,813
	implicit	good taste	10,473	14,043	122,060
		good environment	10,293	12,135	90,350
		good service	10,354	13,271	105,846
		good overall	10,425	14,283	125,173

where $\hat{x}_{uij}$ is the difference between predicted ranking score $\hat{R}_{ui}$ and $\hat{R}_{uj}$ as defined above. And $[c]$ denotes a condition indicator that returns 1 iff c is true and otherwise 0. We can note that the metric *Zero-One Error* is similar to *AUC* (Area Under the ROC Curve).

NDCG@k [15] is designed to take into count the order of items in the recommendation list. To define $NDCG_u@k$ for a user u, $DCG_u@k$ should be given formally first:

$$DCG_u@k = \sum_{i=1}^{k} \frac{2^{R_{ui}} - 1}{log_2(i+1)}, \tag{19}$$

where i ranges over positions in the recommended list of user u, we use the observed rating R_{ui} to weigh the degree user u prefers item i. $NDCG_u@k$ is the ratio of $DCG_u@k$ to ideal *DCG* for that user:

$$NDCG_u@k = \frac{DCG_u@k}{IDCG_u@k}, \tag{20}$$

where $IDCG_u@k$ is the maximum possible *DCG* when the recommended items are just in descending order by user u preference. *NDCG@k* is the mean value of $NDCG_u@k$ over all users, reflecting the model performance of the recommended list at the top k ranking.

6.3 Comparison Methods

We compare the performance of the proposed SFPR and MFPR with five representative methods. According to different problem settings, the methods can be classified into three categories: explicit feedback based (i.e., SVD), implicit feedback based (i.e., BPRMF, EN-BPRMF), hybrid feedback based (i.e., MP, SVD++ and FM). These baselines are summarized as follows.

- Most Popular (MP). This baseline ranks items according to their popularity and is non-personalized.
- SVD [14]. This method is a typical matrix factorization based model. It is a rating prediction model and the input data need only rating information. We rank items using the predicted ratings in our experiments.
- BPRMF [25]. This pairwise ranking method introduced by Rendle et al. is a personalized ranking model using only one type of implicit feedback.

- Ensemble of BPRMF (EN-BPRMF) [5]. This method is an ensemble approach to unify different types of implicit feedback based on BPRMF. In the experiments, we ensemble all types of implicit feedback using this approach.
- SVD++ [14]. This method is also a matrix factorization based rating prediction model and it integrates rating information with one type of implicit feedback. We rank items using the predicted ratings.
- MSVD++. This method is adapted from SVD++. It integrates rating information with multi types of implicit feedback using Equation (15) to predict ratings. Similarly, we rank items by predicted ratings.
- Factorization Machine (FM) [23]. This method is a general predictor working with any real valued feature vector and combines the advantages of support vector machines with factorization models. We integrate rating information and all types of implicit feedback into the feature vector. It is a rating prediction model and, we rank items using the predicted rating.

Since BPRMF, SVD++ and the proposed SFPR need one type of implicit feedback, we choose the "read" feedback in Douban Book and the "good overall" feedback in Dianping for them; the reason is that the best performances are achieved in these conditions, and the details are explained in Section 6.6. In addition, some baselines are obtained from open resources. FM is from libFM [24], and MP and BPRMF are from MyMediaLite [7]. Moreover, these methods are set to optimal parameters on these datasets.

6.4 Effectiveness

This section validates the effectiveness of the proposed SFPR and MFPR compared to those baselines. For Douban Book and Dianping datasets, we generate training set $\mathcal{T}_r$ and testing set $\mathcal{T}_e$ using different split ratios 30%, 50%, 70%. The random split was carried out 5 times independently in all experiments, and we report the mean values of $\varepsilon_{0/1}$ and *NDCG*.

For fair comparison, we set the same number of latent dimension $d = 10$ for all matrix factorization based methods. Parameters of all methods are tuned to the optimal values through cross validation. We select $\varepsilon_{0/1}$, *NDCG*@5, *NDCG*@10, and *NDCG*@15 as evaluation metrics. We also record the improvement ratio on these evaluation metrics of all methods compared to the SVD. Moreover, we also conduct the t-test experiments with 95% confidence, which shows that the $\varepsilon_{0/1}$ and the *NDCG* improvement difference is statistically stable and non-contingent. The experimental results are shown in Tables 2 and 3. The main findings from the experimental comparisons are summarized as follows:

- MFPR achieves the best performance in all conditions, which validates the significant benefits of integrating both explicit feedback and multiple implicit feedback. The experiments also confirm that better performance can be achieved by integrating more feedback information. For example, for those ranking methods, SFPR outperforms BPRMF due to the integration of ratings, and the superiority of MFPR to SFPR is from more implicit feedback. For those rating prediction methods, SVD++ outperforms SVD because of implicit feedback. MSVD++ outperforms SVD++, because it integrates multiple implicit feedback. MSVD++ is not inferior to SFPR, because MSVD++ integrates multiple implicit feedback. Note that MSVD++, FM, and MFPR utilize all feedback information, while MFPR always has better performance; MFPR not only designs an effective mechanism treating explicit and implicit feedback differently but also uses an effective rank model, while FM handles all feedback

Table 2. Performance Comparisons on Douban Book (d = 10, the Baseline of Improvement Ratio Is SVD)

Training	Metric	MP	SVD	BPRMF	EN-BPRMF	SVD++	MSVD++	FM	SFPR	MFPR
30%	$\varepsilon_{0/1}$	0.5210	0.5251	0.5314	0.5372	0.6089	0.6260	0.6145	0.6270	**0.6307**
	Improve	−0.66%		1.20%	2.30%	15.96%	19.22%	17.03%	19.41%	20.11%
	NDCG@5	0.7831	0.7879	0.7845	0.7861	0.8291	0.8371	0.8288	0.8371	**0.8399**
	Improve	−0.78%		−0.43%	−0.23%	5.23%	6.24%	5.19%	6.24%	6.60%
	NDCG@10	0.8301	0.8332	0.8318	0.8323	0.8656	0.8718	0.8691	0.8706	**0.8726**
	Improve	−0.37%		−0.17%	−0.11%	3.89%	4.63%	4.31%	4.49%	4.73%
	NDCG@15	0.8559	0.8576	0.8567	0.8575	0.8852	0.8905	0.8885	0.8897	**0.8917**
	Improve	−0.20%		−0.10%	−0.01%	3.22%	3.84%	3.60%	3.74%	3.98%
50%	$\varepsilon_{0/1}$	0.5225	0.5909	0.5299	0.5374	0.6396	0.6511	0.6399	0.6605	**0.6636**
	Improve	−11.58%		−10.32%	−9.05%	8.24%	10.19%	8.29%	11.78%	12.30%
	NDCG@5	0.7969	0.8347	0.7989	0.7994	0.8516	0.8576	0.8500	0.8564	**0.8611**
	Improve	−4.53%		−4.29%	−4.23%	2.02%	2.74%	1.83%	2.60%	3.16%
	NDCG@10	0.8478	0.8747	0.8493	0.8494	0.8887	0.8927	0.8864	0.8927	**0.8959**
	Improve	−3.08%		−2.90%	−2.89%	1.60%	2.06%	1.34%	2.06%	2.42%
	NDCG@15	0.8705	0.8933	0.8714	0.8719	0.9052	0.9086	0.9035	0.9088	**0.9118**
	Improve	−2.55%		−2.45%	−2.40%	1.33%	1.71%	1.14%	1.74%	2.07%
70%	$\varepsilon_{0/1}$	0.5239	0.6242	0.5312	0.5397	0.6558	0.6639	0.6582	0.6676	**0.6756**
	Improve	−16.07%		−14.90%	−13.54%	5.06%	6.36%	5.45%	6.95%	8.23%
	NDCG@5	0.8338	0.8791	0.8403	0.8409	0.8874	0.8899	0.8875	0.8895	**0.8932**
	Improve	−5.15%		−4.41%	−4.35%	0.94%	1.22%	0.96%	1.18%	1.60%
	NDCG@10	0.8814	0.9110	0.8821	0.8824	0.9172	0.9189	0.9164	0.9196	**0.9220**
	Improve	−3.25%		−3.17%	−3.14%	0.68%	0.88%	0.59%	0.94%	1.21%
	NDCG@15	0.8953	0.9212	0.8957	0.8959	0.9270	0.9282	0.9273	0.9286	**0.9309**
	Improve	−2.81%		−2.27%	−2.75%	0.63%	0.76%	0.66%	0.80%	1.05%

equally. In all, exploiting and integrating multiple feedback is really helpful to improve the performance in the personalized ranking recommendation task.

- When considering different training data ratios, we can find that the improvements of those models integrating explicit feedback with implicit feedback (i.e., SVD++, MSVD++, FM, SFPR, and MFPR) over the SVD are more significant for fewer training data. This indicates that integrating implicit feedback into models can effectively alleviate data sparsity of rating information. Specifically, MSVD++, FM outperforms SVD++ and MFPR outperforms SFPR, because of more implicit feedback is integrated. More combined implicit feedback means more supplementary information for ratings. Thus, it is necessary to achieve much better recommendation performance by integrating comprehensive multiple feedback, particularly when rating information is insufficient.
- From the results, we can also note that pairwise methods are more suitable for personalized ranking recommendations. Specifically, SVD, SVD++, MSVD++, and FM are rating prediction models, also known as pointwise methods, while SFPR and MFPR are pairwise

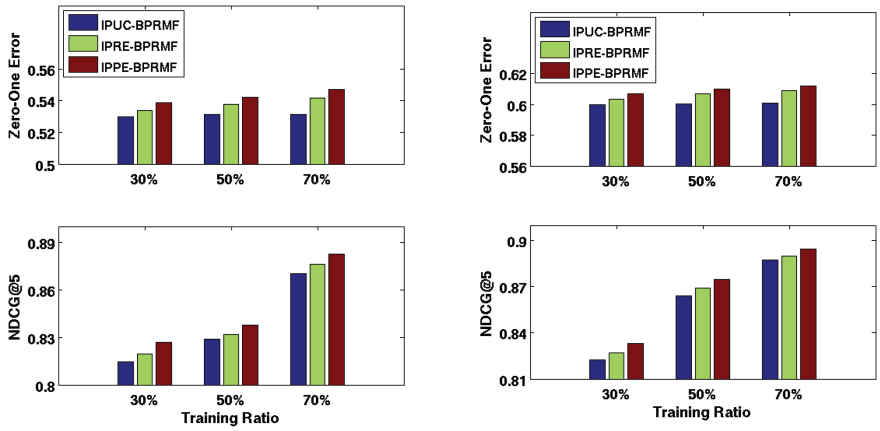
Table 3. Performance Comparisons on Dianping (d = 10, the Baseline of Improvement Ratio Is SVD)

Training	Metric	MP	SVD	BPRMF	EN-BPRMF	SVD++	MSVD++	FM	SFPR	MFPR
30%	$\varepsilon_{0/1}$	0.5957	0.5922	0.5999	0.6072	0.6118	0.6148	0.6220	0.6248	**0.6253**
	Improve	0.59%		1.30%	2.53%	3.31%	3.82%	5.03%	5.50%	5.59%
	NDCG@5	0.8214	0.8178	0.8225	0.8261	0.8293	0.8314	0.8365	0.8377	**0.8387**
	Improve	0.44%		0.57%	1.01%	1.41%	1.67%	2.29%	2.43%	2.56%
	NDCG@10	0.8619	0.8594	0.8630	0.8658	0.8692	0.8704	0.8689	0.8721	**0.8752**
	Improve	0.29%		0.42%	0.74%	1.14%	1.28%	1.11%	1.48%	1.84%
	NDCG@15	0.8776	0.8750	0.8789	0.8814	0.8843	0.8852	0.8843	0.8861	**0.8896**
	Improve	0.30%		0.45%	0.73%	1.06%	1.17%	1.06%	1.27%	1.67%
50%	$\varepsilon_{0/1}$	0.5965	0.6191	0.6009	0.6062	0.6304	0.6330	0.6307	0.6345	**0.6367**
	Improve	−3.65%		−2.94%	−2.08%	1.83%	2.25%	1.87%	2.49%	2.84%
	NDCG@5	0.8628	0.8727	0.8643	0.8674	0.8774	0.8792	0.8778	0.8801	**0.8815**
	Improve	−1.13%		−0.96%	−0.61%	0.54%	0.74%	0.58%	0.85%	1.01%
	NDCG@10	0.8924	0.8999	0.8940	0.8961	0.9044	0.9053	0.9040	0.9056	**0.9076**
	Improve	−0.83%		−0.66%	−0.42%	0.50%	0.59%	0.46%	0.63%	0.86%
	NDCG@15	0.9030	0.9097	0.9045	0.9066	0.9141	0.9149	0.9136	0.9145	**0.9165**
	Improve	−0.74%		−0.57%	−0.34%	0.48%	0.57%	0.43%	0.53%	0.75%
70%	$\varepsilon_{0/1}$	0.5987	0.6348	0.6006	0.6103	0.6411	0.6439	0.6437	0.6468	**0.6498**
	Improve	−5.69%		−5.39%	−3.86%	0.99%	1.43%	1.40%	1.89%	2.36%
	NDCG@5	0.8858	0.8982	0.8875	0.8891	0.9012	0.9015	0.8996	0.9015	**0.9029**
	Improve	−1.38%		−1.19%	−1.01%	0.33%	0.37%	0.16%	0.37%	0.50%
	NDCG@10	0.9099	0.9196	0.9110	0.9126	0.9217	0.9222	0.9209	0.9219	**0.9234**
	Improve	−1.05%		−0.94%	−0.76%	0.23%	0.28%	0.14%	0.25%	0.41%
	NDCG@15	0.9172	0.9259	0.9183	0.9197	0.9276	0.9284	0.9272	0.9280	**0.9297**
	Improve	−0.94%		−0.82%	−0.67%	0.18%	0.27%	0.14%	0.23%	0.41%

ranking models. Particularly, SFPR uses the same base learner as SVD++, while MFPR uses the same base learner as MSVD++. We can see that SFPR and MFPR outperform SVD++ and MSVD++, respectively, which demonstrates the effectiveness of the pairwise method. Note that the other two pairwise ranking models (i.e., BPRMF and EN-BPRMF) fail to defeat those pointwise models. We think the reason lies in the fact that BPRMF and EN-BPRMF only utilize implicit feedback, so they fail to generate accurate partial order item pairs as a training set. In contrast, our SFPR and MFPR generate item pairs with a more accurate ranking order as a training set from explicit feedback.

6.5 Impact of Different Training Set Generation Algorithms

Next, we verify the effectiveness of the designed training set generation method IPPE. As shown in Algorithm 2, the method IPPE is designed to make full use of the high-quality explicit feedback (i.e., rating), and, thus, the proposed training set generation approach mainly focuses on the ratings of users. To validate the superiority of the IPPE, we compare it with the following two baseline



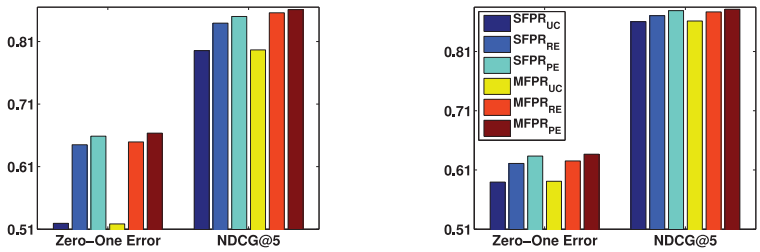
(a) Douban Book (b) Dianping

Fig. 4. The comparison of the algorithms IPUC, IPRE, and IPPE.

methods. Following the idea of BPRMF in Reference [25], for user u, we make a Cartesian product of $E_{tr}(u)$ with a user's unknown items to construct a training set. We name this approach *IPUC*, which means *I*tem *P*airs of partial order are obtained from an *U*nknown item related *C*artesian product. We also consider a variation of the IPPE method. For $E_{tr}(u)$ of each user u, we sample two items each time randomly and generate the item pair with partial order according to their observed ratings. To produce a similar training data size as Algorithm 2, the random process for each user u was conducted $|E_{tr}(u)|$ times. We refer to this approach as *IPRE*, which means *I*tem *P*airs of partial order are obtained from checking *R*andom pairs in an *E*xplicit item set. And we retain the same generation strategy for the testing set as in Algorithm 2 for these two approaches.

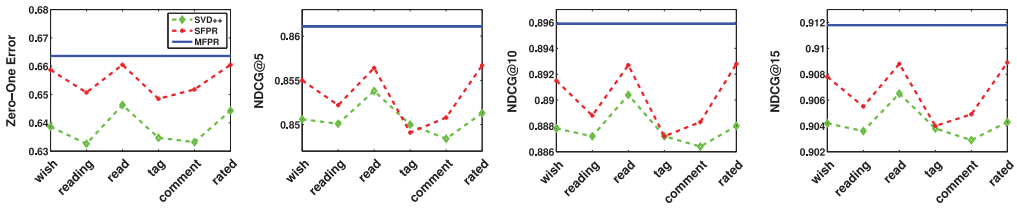
To validate the effectiveness of IPPE, we first apply the IPUC, IPRE, and IPPE algorithms to the BPRMF model [25] and refer to them as IPUC-BPRMF, IPRE-BPRMF, and IPPE-BPRMF, respectively. For fair comparison, we use three algorithms to generate partial pairs from the Douban Book and Dianping datasets with same training set size. Here, we use three different split ratios: 30%, 50%, and 70%, and we report the performance of these three methods on $\varepsilon_{0/1}$ and *NDCG*@5 in Figure 4(a) and Figure 4(b). On both datasets, the IPPE-BPRMF is superior to IPUC-BPRMF and IPRE-BPRMF; the algorithm IPPE makes full use of the rating data, and the generated training pairs have a more accurate partial order.

Furthermore, we apply these three different training set generation algorithms in SFPR and MFPR. As shown in Figure 5, SFPR based on the methods IPUC, IPRE, and IPPE are referred to as SFPR_{UC}, SFPR_{RE}, and SFPR_{PE}, respectively. This is similar for MFPR. We conduct experiments on both the Douban Book and Dianping datasets, and the "read" feedback and the "good overall" feedback are still chosen for the SFPR. The performance on *Zero-One Error* and *NDCG*@5 with the 70% training set are reported in Figure 5(a) and Figure 5(b). We can observe that IPPE-based models show much better performance than IPUC-based models. Specifically, SFPR_{UC} and MFPR_{UC} exhibit very bad performance, such as BPRMF in Tables 2 and 3. Since the method IPPE makes full use of the rating information, the corresponding training set $\mathcal{T}_r$ consists of item pairs with more accurate partial order. On the contrary, the approach IPUC simply discards the item orders implied by rating information and handles the rating as ordinary implicit feedback. Moreover, we observe that SFPR_{PE} and MFPR_{PE} outperform SFPR_{RE} and MFPR_{RE}, respectively, slightly but stably. This shows that sampling adjacent item pairs from random permutations is a better strategy than



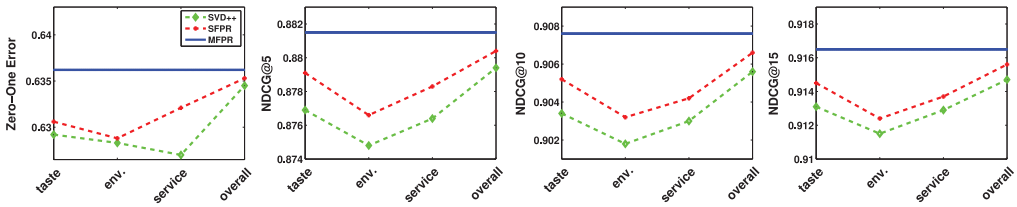
(a) Douban Book (b) Dianping

Fig. 5. Performance of the models with different training set generation algorithms on Douban Book and Dianping.



(a) $\varepsilon_{0/1}$ (b) $NDCG@5$ (c) $NDCG@10$ (d) $NDCG@15$

Fig. 6. Integration with different implicit feedback in Douban Book.



(a) $\varepsilon_{0/1}$ (b) $NDCG@5$ (c) $NDCG@10$ (d) $NDCG@15$

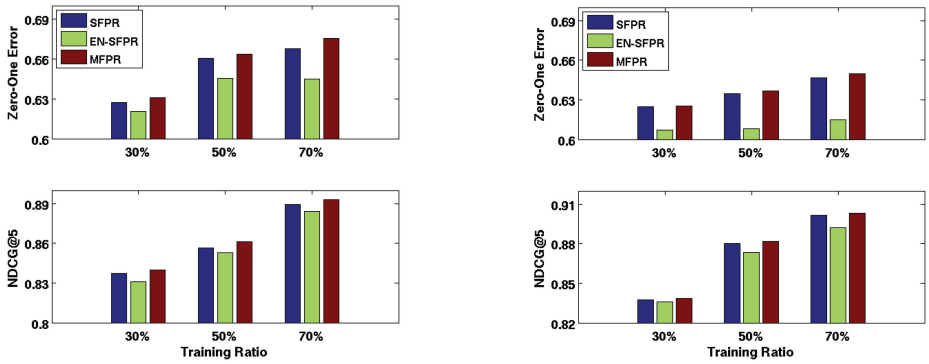
Fig. 7. Integration with different implicit feedback in Dianping.

sampling item pairs randomly. In summary, for multiple feedback data, the proposed IPPE method is more effective at generating training sets for the personalized ranking models.

6.6 Integrate Different Implicit Feedback with Rating

Here we explore the impacts of integrating different types of implicit feedback with rating information. Here, we apply SVD++ and SFPR to integrate different implicit feedback with ratings. That is, we employ six different types of implicit feedback (wish, reading, read, tag, comment, and rated) in Douban Book and four different types of implicit feedback (good taste, good environment, good service and good overall) in Dianping. In addition, we also run MFPR to integrate all feedback. The split ratio is set to 50%, and the average results are shown in Figure 6 and Figure 7.

We observe that various implicit feedback makes substantially different contributions to improvement in personalized ranking performance. In Douban Book, among six different types of implicit feedback, the "read" feedback achieves the best performance while the "tag" performs the worst. We think the reason may lie in that the "read" feedback has a stronger indication of user



(a) Douban Book (b) Dianping

Fig. 8. Performance of mean-weighted blending of SFPR and MFPR on Douban Book and Dianping.

preferences than other implicit feedback. In Dianping, "good overall" has the best performance, while "good environment" and "good service" perform relatively poorly. We think that there is a similar reason for this phenomenon. That is, "good overall" shows a stronger indication of user preferences, since it is derived from the overall ratings. In general, the better performance is achieved by integrating implicit feedback with stronger indications of user preferences. Additionally, the "read" feedback in Douban Book and the "good overall" feedback in Dianping perform the best, so we choose the two implicit feedback for those models in the above effectiveness experiments.

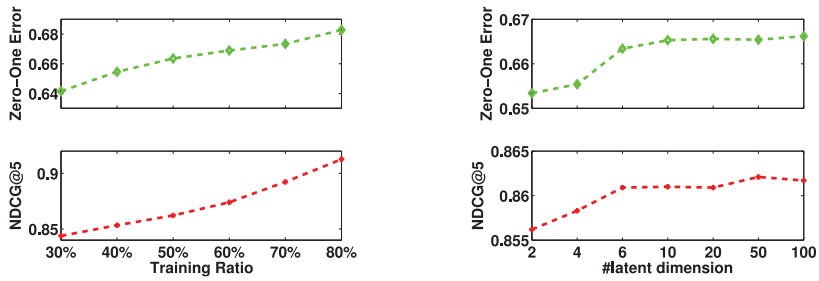
Moreover, MFPR, which integrates all types of implicit feedback, always has better performance than SVD++ and SFPR, which can integrate only one type of implicit feedback with ratings. On both datasets, SFPR outperforms SVD++ in most cases. This verifies again that the pairwise method is more powerful than the pointwise method for this ranking task.

6.7 Mean-weighted SFPR versus MFPR

As mentioned previously, explicit and implicit feedback have different characteristics; thus, the relations between the explicit feedback and the implicit feedback and the relations among different forms of implicit feedback are key points to be considered when designing the personalized ranking model. We verify this claim in this section.

A simple and intuitive way to integrate user multiple feedback is to conduct linear blending of SFPR models with the average weight. Since there are six different types of implicit feedback in the Douban Book dataset and four different types of implicit feedback in the Dianping dataset, the EN-SFPR in Figure 8(a) represents the linear blending of six various SFPR based on six different types of implicit feedback using the average weight $\frac{1}{6}$, and the EN-SFPR in Figure 8(b) represents the linear blending of four various SFPR using the average weight $\frac{1}{4}$. Meanwhile, other models (i.e., SFPR and MFPR) in Figure 8 are the same model as those in Table 2 and Table 3. Here, we use three different split ratios: 30%, 50%, 70%, and the performance of these models on $\varepsilon_{0/1}$ and *NDCG*@5 are shown in Figure 8(a) and Figure 8(b).

Consistent with previous experimental results, MFPR shows better performance than the SFPR. We also observe that the MFPR is superior to the EN-SFPR, and this indicates that simple mean-weighted linear blending of SFPR does address the relations among different forms of implicit feedback well. However, it is not the case for MFPR; the learning process of MFPR handles the relations among various forms of implicit feedback well by delicately modeling those relations



(a) Performance with various ϵ (b) Performance with various d

Fig. 9. Performance of MFPR with various training ratio ϵ and the latent dimension d on Douban Book.

in the MFPR. Moreover, we note that the EN-SFRP is even inferior to the SFPR. As concluded in Section 6.6, each type of implicit feedback contributes significantly differently in promoting model recommendation performance. The EN-SFPR, simple blending of the SFPR, cannot tap the potential of integrating user multiple feedback and even lowers the importance of the most important implicit feedback (e.g., the "read" in Douban Book and the "good overall" in Dianping).

6.8 Parameter Study

Finally, we explore how the training ratio ϵ and the number of latent dimensions d affect the performance of MFPR. Due to the similarity of the experimental results to those of Dianping, here we only show the experimental results of Douban Book.

The training ratio ϵ controls the ratio of explicit feedback data to be trained. In the experiments, we set ϵ with 30%, 40%, 50%, 60%, 70%, and 80%, and Figure 9(a) shows the corresponding results. The performance of MFPR improves as training ratio ϵ increases. It is reasonable that more training data are helpful to enhance the recommendation performance.

The number of latent dimensions d is an important parameter for matrix factorization-based models. Generally, performance of matrix factorization-based models improves as the latent dimension d increases. However, considering the time complexity of MFPR (see Section 4.3), a larger d indicates a longer training time and lower prediction efficiency. Thus, the proper d is set to balance accuracy and efficiency. In the experiments, we set d with 2, 4, 6, 10, 20, 50, and 100, and the corresponding results are shown in Figure 9(b). We observe that when d grows from 2 to 10, the performance of MFPR improves significantly. However, when d grows from 10 to 100, the performance of MFPR for the most part remains steady. Hence, to balance the model's accuracy and efficiency, $d = 10$ is set for all matrix factorization-based methods in our experiments, as mentioned.

7 CONCLUSION AND FUTURE WORK

In this article, we conjecture that integrating explicit feedback (i.e., ratings) and multiple implicit feedback can effectively improve personalized recommendation performance. Hence, we study the personalized ranking recommendation problem integrating multiple feedback, and a unified multiple feedback based personalized ranking framework MFPR. Extensive experiments on two real-world datasets show that MFPR outperforms state-of-the-art models that use rating or implicit feedback or hybrid feedback. Moreover, we have also designed a delicate algorithm IPPE to generate training data with a more accurate partial order for the proposed ranking model. The empirical evaluation results also show that IPPE is a good training data generation strategy.

The implicit feedback exploited in this article all indicate positive user preferences. In the future, we will further exploit implicit feedback with negative user preferences (e.g., "dislike" and "skip") and other types of explicit feedback. In addition, other LTR models (e.g., the listwise rank model) can be applied to better integrate multiple feedback information. Since the proposed method is an offline algorithm, an online version would be useful to extend its applicability.

REFERENCES

[1] Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 13–24.

[2] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML'05)*. ACM, 89–96.

[3] Christopher J. C. Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23–581 (2010), 81.

[4] Arthur F. da Costa and Marcelo G. Manzato. 2014. Multimodal interactions in recommender systems: An ensembling approach. In *Poceedings of the Brazilian Conference on Intelligent Systems (BRACIS'14*). IEEE, 67–72.

[5] Arthur da Costa Fortes and Marcelo Garcia Manzato. 2014. Ensemble learning in recommender systems: Combining multiple user interactions for ranking personalization. In *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*. ACM, 47–54.

[6] MáRcia GonçAlves De Oliveira, Patrick Marques Ciarelli, and Elias Oliveira. 2013. Recommendation of programming activities by multi-label classification for a formative assessment of students. *Expert Syst. Appl.* 40, 16 (2013), 6641–6651.

[7] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. MyMediaLite: A free recommender system library. In *Proceedings of the ACM Conference Series on Recommender Systems (RecSys'11)*. ACM, 305–308.

[8] Tural Gurbanov and Francesco Ricci. 2017. Action prediction models for recommender systems based on collaborative filtering and sequence mining hybridization. In *Proceedings of the Symposium on Applied Computing*. ACM, 1655–1661.

[9] Tural Gurbanov, Francesco Ricci, and Meinhard Ploner. 2016. Modeling and predicting user actions in recommender systems. In *Proceedings of the User Modelling, Adaptation and Personalization Conference (UMAP'16)*. ACM, 151–155.

[10] Ruining He and Julian McAuley. 2016. VBPR: Visual bayesian personalized ranking from implicit feedback. In *AAAI*. 144–150.

[11] Patrik O. Hoyer. 2004. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research* 5 (2004), 1457–1469.

[12] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'08)*. IEEE, 263–272.

[13] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceeedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'02)*. ACM, 133–142.

[14] Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceeedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'08)*. ACM, 426–434.

[15] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2014. Local collaborative ranking. In *Proceedings of the International World Wide Web Conference (WWW'14)*. ACM, 85–96.

[16] Hang Li. 2009. Learning to rank. In tutorial given at the *Association for Computational Linguistics International Joint Conference on Natural Language Processing (ACL-IJCNLP'09)*.

[17] Jian Liu, Chuan Shi, Binbin Hu, Shenghua Liu, and S. Yu Philip. 2017. Personalized ranking recommendation via integrating multiple feedbacks. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 131–143.

[18] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Found. Trends Inf. Retriev.* 3, 3 (2009), 225–331.

[19] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM'11)*. ACM, 287–296.

[20] Douglas W. Oard, Jinmook Kim, and others. 1998. Implicit feedback for recommender systems. In *Proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI'98)*. 81–83.

[21] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'08)*. IEEE, 502–511.

[22] Christopher J. Burges, Robert Ragno, and Quoc V. Le. 2007. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems*. 193–200.
[23] Steffen Rendle. 2010. Factorization machines. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'10)*. IEEE, 995–1000.
[24] Steffen Rendle. 2012. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.* 3, 3 (2012), 57:1–57:22.
[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI'09)*. AUAI Press, 452–461.
[26] Andriy Mnih and Ruslan R. Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*. 1257–1264.
[27] Nathan Srebro, Tommi Jaakkola, and others. 2003. Weighted low-rank approximations. In *Proceedings of the International Conference on Machine Learning (ICML'03)*, Vol. 3. 720–727.
[28] Liang Tang, Bo Long, Bee-Chung Chen, and Deepak Agarwal. 2016. An empirical study on recommendation with multiple types of feedback. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 283–292.
[29] Xiao Yu, Xiang Ren, Quanquan Gu, Yizhou Sun, and Jiawei Han. 2013. Collaborative filtering with entity similarity regularization in heterogeneous information networks. In *Proceedings of the International Joint Conference on Artificial Intelligence Workshop on Heterogeneous Information Network Analysis (IJCAI HINA'13)*.

Received May 2017; revised April 2018; accepted May 2018