Boosting Response Aware Model-Based Collaborative Filtering

Haiqin Yang, *Member, IEEE*, Guang Ling, Yuxin Su, Michael R. Lyu, *Fellow, IEEE*, and Irwin King, *Senior Member, IEEE*

Abstract—Recommender systems are promising for providing personalized favorite services. Collaborative filtering (CF) technologies, making prediction of users' preference based on users' previous behaviors, have become one of the most successful techniques to build modern recommender systems. Several challenging issues occur in previously proposed CF methods: 1) most CF methods ignore users' response patterns and may yield *biased parameter estimation* and suboptimal performance; 2) some CF methods adopt heuristic weight settings, which lacks a systematical implementation; and 3) the multinomial mixture models may weaken the computational ability of matrix factorization for generating the data matrix, thus increasing the computational cost of training. To resolve these issues, we incorporate users' response models into the probabilistic matrix factorization (PMF), a popular matrix factorization CF model, to establish the response aware probabilistic matrix factorization (RAPMF) framework. More specifically, we make the assumption on the user response as a Bernoulli distribution which is parameterized by the rating scores for the observed ratings while as a step function for the unobserved ratings. Moreover, we speed up the algorithm by a mini-batch implementation and a crafting scheduling policy. Finally, we design different experimental protocols and conduct systematical empirical evaluation on both synthetic and real-world datasets to demonstrate the merits of the proposed RAPMF and its mini-batch implementation.

Index Terms—Recommender systems, collaborative filtering, matrix factorization, missing data theory

1 Introduction

Recently, online shopping and entertainment services are growing explosively. Popular service providers, e.g., Amazon, Netflix, iTunes Match, Yahoo! Music, etc., have contributed to building up platforms for consumers to buy new products or rate them. As a coin has two sides, these platforms can provide users attractive services to improve their lifestyle, they also introduce inundated choice which increases users' information overload. Matching consumers' taste and presenting the most appropriate products to them is a key to enhance users' satisfaction and loyalty in using these online services. Hence, *recommender systems*, providing personalized favorite recommendations, have been prevalently adopted in these services to boost the sales of retailers and trigger the growth of business.

Due to the prominence of the commercial value and technical challenges, recommender techniques have attracted the interests of researchers from academia and practitioners from industry [2], [5], [6], [14], [17], [20], [37], [40]. *Collaborative filtering* (CF) technologies, aiming to automatically predict consumers' preferences by analyzing their previous behaviors, e.g., the transaction history or product ratings, become mainstream techniques for recommender systems. These techniques can usually be classified into *memory-based CF methods* and *model-based CF methods*, see [2], [40] and the references therein.

Overall, previously proposed CF methods mainly focus on manipulating the explicitly observed rating scores to understand users' preferences for future prediction. An explicit rating score clearly indicates a user's preference on a particular item as well as an item's inherent features. The scores that a user assigns to different items convey information on what the user likes and what the user dislikes. The rating values that an item received from different users also carry information on intrinsic properties of the item. The rating information indeed can present users' preferences on different items. However, valuable implicit information of users' response patterns, i.e., some items are rated while others not, is usually less explored in existing CF methods.

Several pieces of research publications have been conducted to exploit users' response patterns. For example, the original problem is formulated as the one-class collaborative filtering task, where a heuristic weight in the range of 0 to 1 is introduced to calibrate the loss on those unseen ratings [27], [28], [39], or the user information is embedded to optimize the weight on the unseen ratings via users' similarity [19]. The multinomial mixture model is combined with conditional probability tables with Bernoulli distribution to model the non-random response [26]. This work is also extended to specify the probability that a rating is missing in a logistic form which depends on both the values of the underlying ratings and the identity of the items [25]. The previous work, however, may suffer from some practical limitations: 1) the heuristic weight setting methods may lack a systematic way to model users' response patterns; 2) the multinomial mixture models may weaken the computational ability of generating data matrix and increase the computational cost of training the model.

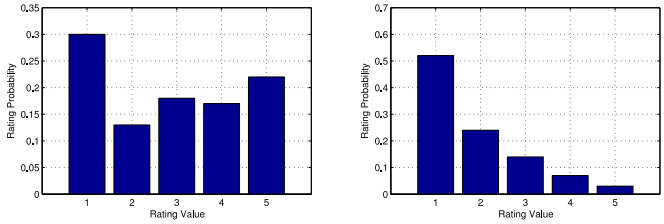
• *The authors are with Shenzhen Key Laboratory of Rich Media Big Data Analytics and Applications, Shenzhen Research Institute, and the Department of Computer Sciences and Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong.*
E-mail: hqyang@ieee.org, {gling, yxsu, lyu, king}@cse.cuhk.edu.hk.

Manuscript received 12 May 2014; revised 20 Jan. 2015; accepted 27 Jan. 2015. Date of publication 18 Feb. 2015; date of current version 2 July 2015.
Recommended for acceptance by M. Spiliopoulou.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2015.2405556

To overcome the above limitations, in this paper, we propose a response aware probability matrix factorization (RAPMF) framework by expanding the Bernoulli response patterns to probability matrix factorization (PMF) for users' ratings in [22]. Different from previously proposed methods, we present a succinct assumption on response patterns and further investigate the properties and effectiveness of the proposed RAPMF. We highlight the key contributions of this article as follows:

- First, our proposed RAPMF framework consists of a data model and a response model. The data model generates users' ratings on items via the inner product of two low-rank feature matrices captured by probabilistic matrix factorization (PMF), a popular model-based matrix factorization method. Meanwhile, the response model is assumed following a Bernoulli distribution. That is, the response patterns are assumed based on whether the ratings are observed or not, where a deterministic Bernoulli parameter is given to the observed ratings while a step function is assumed on the unobserved ratings. The treatment of the response for unobserved ratings allows us to marginalize the underlying response and the data model on the unobserved ratings. This assumption is more precise and easy to marginalize the missing responses. This is slightly different from the setup in [22], where the responses may depend on the latent features.
- Second, we seek the optimal solution of RAPMF by gradient descent, which consumes the time complexity of $O(N \times M)$, where N and M are the number of users and items, respectively. It is too expensive for real-world recommender systems, which contain over millions or even billions of users and items. To resolve the computational issue, we realize a mini-batch implementation for RAPMF and reduce its training cost to $O(B^2)$ for each mini-block with B users and B items. The mini-batch is executed in parallel via multiple threads with a crafting scheduling policy. In an extreme case, when $B = 1$, the algorithm is equivalent to a parallel implementation of stochastic gradient ascent. Hence, the mini-batch implementation will reduce the training consumption of RAPMF largely while maintaining the same test cost.
- Third, we design different experimental protocols to reveal different distributions on the training data and the test data. We conduct model evaluation on both synthetic and real-world datasets under different protocols to compare the model performance. Our experimental results demonstrate that our proposed RAPMF contains several merits.

The rest of the paper is organized as follows. In Section 2, we present the preliminaries on the basic model setup and a motivating example, review several existing work, and the motivation of the work. In Section 3, we develop the proposed RAPMF model on how to incorporate response models into PMF and elaborate its properties. In Section 4, we present the mini-batch learning implementation for RAPMF. In Section 5, we conduct comparison on the models and present detailed explanation. Finally, we conclude the paper in Section 6.



(a) User selected ratings (b) Randomly selected ratings

Fig. 1. Distribution of rating scores from Yahoo!Music [26]. (a) shows the probability of rating scores obtained from the system while (b) presents the probability of ratings, where the songs are randomly selected from the system and requested users to rate.

2 Preliminaries and Related Work

In the following, we will first present the basic setup and the objective of this paper with a motivating example. After that, we will review three main topics related to our work. They include missing data theory, collaborative filtering, and online learning algorithms. We will emphasize how these topics motivate our work.

2.1 Setup and a Motivating Example

Let $\mathbb{D} = \{1, 2, \ldots, D\}$ be the set of rating scores (grades) in the range 1 to D. For example, in the Yahoo!Music's LaunchCast dataset, D is 5 and therefore the rating values range from 1 (indicating no interest) to 5 (implying a strong interest). Collecting all data of N users and M items from a recommender system can form an $N \times M$ matrix X, where a row of the matrix indicates a user's ratings on the items and a column of the matrix represents the ratings on a specific item. Usually, the observed matrix X is highly sparse. For example, in the Yahoo!Music's LaunchCast dataset, only about 2 percent of the ratings are observed. Formally, we denote Ω as the set of the indexes of the observations in X and likewise $\bar{\Omega}$ for the unobserved data. Hence, we separate X into two sets, X_Ω and $X_{\bar{\Omega}}$, for the observed ratings and unobserved ratings, respectively, where

$$X_{ij} = \begin{cases} a \in \mathbb{D}, & \text{if } (i,j) \in \Omega, \\ 0, & \text{if } (i,j) \in \bar{\Omega}. \end{cases} \tag{1}$$

Correspondingly, we can then construct the fully observed response matrix R as

$$R_{ij} = \begin{cases} 1, & \text{if } (i,j) \in \Omega, \\ 0 & \text{if } (i,j) \in \bar{\Omega}. \end{cases} \tag{2}$$

Hence, $R = R_\Omega \cup R_{\bar{\Omega}}$ and $R_\Omega \cap R_{\bar{\Omega}} = \emptyset$.

In most of previously proposed CF methods, users response patterns are ignored, which is equivalent to assuming the missing of users' ratings on items occurs randomly. That is, all users would rate *all* the inspected items, or more generally they will *randomly* select the inspected items to rate. It should be noted that in real-world recommender systems, this assumption may be violated. To verify this phenomenon, we show in Fig. 1 for the distributions of

TABLE 1
Skewed Ratings of Five Users on Five Items and the Corresponding Response Patterns

Ratings

	i_1	i_2	i_3	i_4	i_5
u_1	5	4			
u_2		5		4	
u_3	4			4	
u_4	5		5		
u_5		4			5

Response patterns

	i_1	i_2	i_3	i_4	i_5
u_1	**1**	**1**	0	0	0
u_2	0	**1**	0	**1**	0
u_3	**1**	0	0	**1**	0
u_4	**1**	0	**1**	0	0
u_5	0	**1**	0	0	**1**

rating scores collected from a real-world system, the Yahoo! Music's LaunchCast Radio service [26]. Fig. 1a shows the distribution of rating scores on those items that users *choose to rate*, while Fig. 1b shows the distribution of rating scores for the songs which are *randomly* selected from the whole music pool and asked for rating by the same group of users. Obviously, these two distributions are dramatically different. For those songs that the users have rated, more items are rated on high scores than those randomly selected from the music pool. This is a compelling evidence showing that the assumption that all the users would rate all the inspected items or randomly select items to rate is unlikely to be true. The investigation of the Yahoo!Music LaunchCast data indicates that users are more likely to rate items they do love or hate than those neutral to them [26], [38].

Table 1 again gives us a vivid example of skewed ratings of five users on five items and their corresponding response patterns. This extreme case (ratings skewed to either 4 or 5) clearly shows that without considering users' response patterns, user-based approaches [3], [11] and item-based approaches [9], [20], [35] are more likely to predict rating values in the range of 4 to 5. The extreme example implies that the response patterns have to be taken into account to enhance model performance. Hence, in this paper, we aim to boost the model performance by exploiting both partially-observed rating matrix X_Ω and fully-observed response matrix R.

2.2 Missing Data Theory

In the literature, missing data theory [23] has established a systematic framework to explore missing response patterns. In the following, we review this theory and elaborate how it can be utilized in collaborative filtering because ignoring the missing responses will yield *biased* parameter estimation.

Following missing data theory, we can model the available data in Eq. (1) and Eq. (2) as a two-step procedure. First, a data model $P(X|\theta)$ parameterized by θ generates the full data matrix X. Then, a response model $P(R|X,\mu)$ determines which elements in X are observed. Hence, we can take a parametric joint probability on the partially observed data matrix X_Ω and the fully observed response matrix R, conditioned on the model parameters, θ and μ as follows:

$$\begin{aligned} P(R, X_\Omega|\mu,\theta) &= P(R|X_\Omega,\mu,\theta)P(X_\Omega|\mu,\theta) \\ &= P(R|X_\Omega,\mu)P(X_\Omega|\theta). \end{aligned} \tag{3}$$

In Eq. (3), the response parameter, μ, is not related to users' rating and therefore we discard it in calculating the probability $P(X_\Omega|\theta)$. The probability, $P(R|X_\Omega,\mu)$ is also referred to as the *missing data model*. In the following, we use *response model* and *missing data model* interchangeably.

According to the missing data theory, there are three kinds of missing data assumptions:

- *Missing completely at random (MCAR).* This is the strongest independence assumption. Whether there is a response is fully determined by a parameter, which is irrelevant to users' ratings and the model's latent variables. One typical example where MCAR holds is that given an inspected item, whether it will be observed or not is a Bernoulli trail with probability μ. That is,

$$P(R|X,\mu) = P(R|X_\Omega,\mu) = P(R|\mu). \tag{4}$$

- *Missing at random (MAR).* The probability of observing a particular response can only depend on the observed elements of the data vector. In other words, the probability of response is not related to missing data values or the model's latent variables. The assumption can be formulated as follows:

$$P(R|X,\mu) = P(R|X_\Omega,\mu). \tag{5}$$

- Not Missing At Random (NMAR). The response patterns depend on either the unobserved data vectors, the unobserved values of latent variables, or the observed data. This assumption requires an explicit response model to learn unbiased model parameters. The response model has to be incorporated into the data model for estimating missing ratings.

In the following, we derive the likelihood of μ and θ given the available data X_Ω and R, $\mathcal{L}(\mu,\theta|X_\Omega,R)$, under the MCAR or the MAR assumption for model parameter estimation

$$\begin{aligned} \mathcal{L}(\mu,\theta|X_\Omega,R) &= P(R, X_\Omega|\mu,\theta) \\ &= \int_{X_{\bar\Omega}} P(R, X|\mu,\theta)dX_{\bar\Omega} \\ &= \int_{X_{\bar\Omega}} P(R|X,\mu)P(X|\theta)dX_{\bar\Omega} \\ &= \int_{X_{\bar\Omega}} P(R|X_\Omega,\mu)P(X|\theta)dX_{\bar\Omega} \\ &= P(R|X_\Omega,\mu)\int_{X_{\bar\Omega}} P(X|\theta)dX_{\bar\Omega} \\ &= P(R|X_\Omega,\mu)P(X_\Omega|\theta) \\ &\propto P(X_\Omega|\theta). \end{aligned} \tag{6}$$

In the above derivation, the key to marginalizing the missing data is that the missing data model depends only on the observed data. That is, the expression in Eq. (6) can be replaced by the MCAR assumption in Eq. (4), or the MAR assumption in Eq. (5). It is noted that if both MCAR and MAR fail to hold, the marginalization in Eq. (6) cannot be taken out easily. Hence, directly maximizing $P(X_\Omega|\theta)$ will yield a *biased* θ. In this case, the NMAR assumption has to be explicitly made on the response model to learn an unbiased model.

2.3 Collaborative Filtering Techniques

Collaborative filtering approaches are effective recommendation techniques to filter out irrelevant information only based on users' previous behaviors and to provide items/products that users may be interested [2], [3], [14], [40]. Due to effective performance, they have been successfully deployed in various real-world recommender systems [20], [40]. Based on different assumptions, CF approaches are usually classified into two main categories: *memory-based methods* and *model-based methods* [2], [3], [40].

Memory-based methods are very popular and applied widely in commercial websites [20], [31]. These methods make predictions based on users' previous ratings to compute similarity between users or items. They can further be classified into user-based methods and item-based methods with the facts that neighbor users share similar tasks and users tend to assign similar ratings to similar items, respectively [3], [31], [35]. The success of memory-based methods relies on accurately computing the paired similarity between users and items from previously observed ratings. However, for those unobserved ratings, the information is discarded. The response patterns are usually ignored in these methods. Some other methods, e.g., nearest neighbor regression [40], may be able to correctly identify relevant neighbors for a user or an item in the presence of non-random missing data using common similarity measures like Pearson correlation. If data are not missing at random, these models will yield the predicted results bias [25]. Clearly, as referred to the data in Table 1, user-based approaches [3], [11] and item-based approaches [9], [20], [35] are more likely to predict rating values in the range of 4 to 5.

Model-based approaches, instead of manipulating the ratings directly, train a predefined compact model based on partially-observed user-item rating data to recover the whole matrix. Various models lie in this category, including the aspect models [12], [13], [37], the latent factor model [4], [15], the Bayesian hierarchical model [47], restricted Boltzmann machines [34], SVD++ [16], [17] multi-domain collaborative filtering [46], pair-wise tensor factorization [29], and matrix factorization with social regularization [24], etc. Among model-based approaches, low-rank matrix approximation methods have demonstrated their efficiency and good performance for real-world recommender systems in dealing with large-scale data [14], [16], [17], [18], [30], [32], [33].

Currently, there are two main streams of work trying to include the response patterns in the CF methods. One line of work is to explore the response patterns into the one-class collaborative filtering task [19], [27], [28], [39]. SVD++ with implicit feedback [16], [17] follows similar framework, but embedded users' rating and un-rating behaviors by a latent unknown matrix. In these methods, when the ratings are unobserved, a heuristic weight in the range of 0 to 1 is introduced to calibrate the loss [27], [28], [39] while the ratings are set to 0. The weight on the unseen ratings is also optimized by calculating users' similarity from the embedded users' profile information [19]. However, these methods do not directly explore users' missing response patterns and integrate them with the ratings. The other line of work models the response patterns through missing data theory [23]. In [26], the multinomial mixture model is combined with conditional probability tables with Bernoulli distribution to model the non-random response. This work is also extended to specify the probability that a rating is missing in a logistic form which depends on both the value of the underlying rating and the identity of the item [25]. These methods model users' ratings matrix via the multinomial mixture model and discard the effectiveness and interpretability of the matrix factorization approaches [17], [32]. The PMF for users' data generation model has also incorporated the Bernoulli response patterns [22]. However, the assumption on the missing response patterns can further be simplified. The insufficiency of previous work motivates our exploration of the missing response patterns and matrix factorization model in this paper.

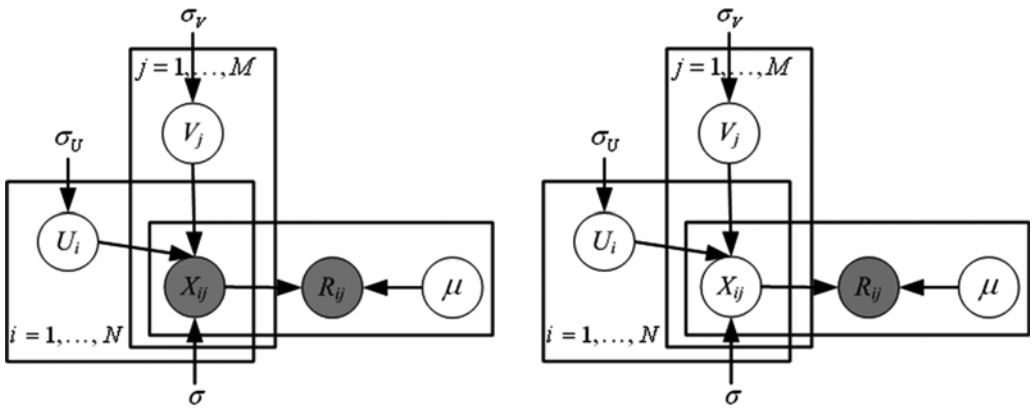
2.4 Online Learning

Online learning is a family of efficient and scalable machine learning algorithms [7], [44], [45]. Different from traditional batch-trained learning algorithms which require that all training data are available prior to the learning task, online learning promptly update the predictive model when a new instance appears [36], [48]. It can avoid the cost of retraining effort largely when a new instance appears. Hence, it is more appropriate for recommender systems to capture users' preference as in real-world systems, ratings are obtained sequentially.

Nowadays, online learning has been extended and explored in collaborative filtering. The implementation of these algorithms can be categorized as Perceptron-like algorithms [10] and stochastic gradient descent approaches [1], [8], [21], [43]. In [10], the work casts online collaborative filtering as an online ranking algorithm. However, it requires to know users' all preferences and it is undesirable for real-world scenario. The implementation of collaborative filtering in stochastic gradient descent is efficient and allows to scale for building on-the-fly recommender systems. In [8], online collaborative filtering is conducted on the probabilistic latent semantic analysis (PLSA) model for the personalized news recommendation. In [1], online collaborative filtering is performed on the low-rank approximation matrix factorization with and without feature models. In [21], online collaborative filtering is also conducted on low-rank approximation matrix factorization models to attain good rating fitting and ranking orders. The online collaborative filtering framework is also extended in [43] to deem each user as each task and reformulate the problem as a multi-task learning problem. The performance is boosted by including users' similarity information as the relationship of tasks. The previously proposed work is also possible and promising for parallelization. The success of previous work motivates us to explore the implementation of stochastic gradient descent in RAPMF. However, since our RAPMF requires the information of both observed and unobserved ratings, to allow balance, we consider the mini-batch method [41], which requires a small bunch of data to update the models, but enjoys the additional advantages of parallelization speedups.

3 Model and Analysis

In this section, we first review the basic model of probabilistic matrix factorization. After that, we present the response



(a) Observed ratings, $(i,j) \in \Omega$ (b) Unobserved ratings, $(i,j) \in \bar{\Omega}$

Fig. 2. Graphical model representation of rating dominant response aware PMF. The shaded and unshaded variables indicate observed and latent variables, respectively. An arrow indicates a conditional dependency between variables and stacked panes indicate a repeated sampling. Note that in (a), both ratings and responses are observed while in (b), only responses are observed.

aware PMF and show how it can incorporate PMF with the response models. The updating rules and complexity analysis are provided accordingly.

3.1 Probabilistic Matrix Factorization

PMF [32] is one of the most popular matrix factorization models in collaborative filtering, which represents the data matrix as the inner product of two low-rank latent feature matrices, U, V, and learns them from the partially observed data matrix X_Ω, where $U \in \mathbb{R}^{K\times N}$, $V \in \mathbb{R}^{K\times M}$, and $K \ll \min(N, M)$. That is,

$$X \approx U^T V.$$

More specifically, PMF assumes Gaussian noise on the observed ratings as follows:

$$P(X_\Omega|U, V, \sigma^2) = \prod_{(i,j)\in\Omega} \mathcal{N}(X_{ij}|U_i^T V_j, \sigma^2), \tag{7}$$

and two zero-mean spherical Gaussian priors on the latent feature matrices

$$P(U|\sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}(U_i|\mathbf{0}, \sigma_U^2\mathbf{I}), P(V|\sigma_V^2) = \prod_{j=1}^{M} \mathcal{N}(V_j|\mathbf{0}, \sigma_V^2\mathbf{I}), \tag{8}$$

where $\mathcal{N}(x|u, \sigma^2)$ is the probability density function of the Gaussian distribution with mean u and variance σ^2.

The latent feature matrices can then be attained by maximizing the following log-likelihood of the posterior distribution with respect to the user and item features:

$$\begin{aligned}\mathcal{L}_{PMF} &= \log P(U, V|X_\Omega, \sigma^2, \sigma_U^2, \sigma_V^2) \\ &\propto \log\big(P(X_\Omega|U, V, \sigma^2)P(U|\sigma_U^2)P(V|\sigma_V^2)\big) \\ &= -\frac{1}{2\sigma^2}\sum_{(i,j)\in\Omega}(X_{ij} - U_i^T V_j)^2 - \frac{\|U\|_F^2}{2\sigma_U^2} - \frac{\|V\|_F^2}{2\sigma_V^2},\end{aligned} \tag{9}$$

where $\|\cdot\|_F^2$ denotes the Frobenious norm.

After training the PMF model via gradient descent or stochastic gradient algorithms [32], the predicted rating that user i would assign to item j can be computed as the expected mean of the Gaussian distribution $\hat{X}_{ij} = U_i^T V_j$.

3.2 Response Aware PMF

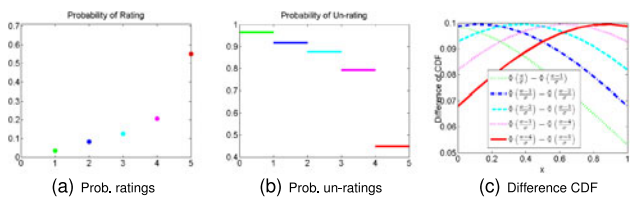
We now start to exploit the response patterns explicitly and present how to include them in the data generation model. Due to the effectiveness and interpretability of PMF, we consider to unify it with explicit response models, which we refer to as response aware PMF.

In RAPMF, the data generation model follows the same as PMF, which can be decomposed into two low-rank feature matrices, see Fig. 2. For the response patterns, we require a correct and tractable distribution. Hence, we employ Bernoulli distribution as it is an intuitive distribution to explain data missing phenomena [26]. More specifically, we propose the *rating dominant* response model in the following.

As we have D discretized ratings, for simplicity, we only take D discretized responses, i.e., $\mu = (\mu_1, \ldots, \mu_D)$. For observed ratings, μ_i indicates the probability of whether a user will rate the item when the preference score is i. For unobserved ratings, since the recovered values for unobserved ratings are continuous, we adopt a *step function* to cover the whole range of all ratings. This gives the following *rating dominant* response patterns assumption $P(R_{ij}|X_{ij}, \mu)$:

$$P(\cdot|\cdot,\cdot) = \begin{cases} \mu_{X_{ij}}, & \text{if } (i,j) \in \Omega, \\ 1 - \mu_k, & \text{if } (i,j) \in \bar{\Omega} \text{ and } k-1 < X_{ij} \le k. \end{cases} \tag{10}$$

Here, we assume that the probability of a user rating on an item follows a Bernoulli distribution, which is illustrated in



(a) Prob. ratings (b) Prob. un-ratings (c) Difference CDF

Fig. 3. Probability of observed ratings with Bernoulli distributions, unobserved ratings in a step function, and the difference of cumulative distribution function on the standard normal distribution.

Fig. 3a. Different from [25], [26] we adopt a step function as shown in Fig. 3b to approximate the probability of un-rating for the sake of calculation simplicity. This assumption makes sense as it encodes the habit of users who usually do not like to rate lowly interested items. For example, suppose k is the underlying score of the user i rating the item j. If the user dislikes the item, the value of k is small, and the corresponding probability μ_k is also small due to the rating habit of users. This yields a large value of $1-\mu_k$. That is, the probability of being an unobserved rating is high.

Moreover, for the expectations of Bernoulli distributions, μ_k's should be in the range of 0 to 1. With the analytical consideration, the logistic function is usually adopted to constrain the range of μ_k's as [25],

$$g(\mu_k) = \frac{1}{1+\exp(-\mu_k)}, \quad k = 1, \ldots, D. \tag{11}$$

Now, we can derive the posterior probability of $P(U, V, \mu | R, X_\Omega, \sigma^2, \sigma_U^2, \sigma_V^2)$ by

$$P(\cdot|\cdot) \propto P(U, V, \mu, R, X_\Omega | \sigma^2, \sigma_U^2, \sigma_V^2). \tag{12}$$

The above proportionality relationship holds since $P(R, X_\Omega | \sigma^2, \sigma_U^2, \sigma_V^2)$ is constant when R and X_Ω are given.

To calculate $P(U, V, \mu, R, X_\Omega | \sigma^2, \sigma_U^2, \sigma_V^2)$, we derive it by separating the data into observed and unobserved ones through the Bayesian inference as follows:

$$\begin{aligned}
&P\big(U, V, \mu, R, X_\Omega | \sigma^2, \sigma_U^2, \sigma_V^2\big) \\
&= \int_{X_{\bar\Omega}} P\big(U, V, \mu, R_\Omega, R_{\bar\Omega}, X_\Omega, X_{\bar\Omega} | \sigma^2, \sigma_U^2, \sigma_V^2\big) dX_{\bar\Omega} \\
&= \int_{X_{\bar\Omega}} P(R_\Omega | X_\Omega, \mu) P(R_{\bar\Omega} | X_{\bar\Omega}, \mu) P(X_\Omega | U, V, \sigma^2) \\
&\quad P(X_{\bar\Omega} | U, V, \sigma^2) P\big(U | \sigma_U^2\big) P\big(V | \sigma_V^2\big) P(\mu) dX_{\bar\Omega} \\
&\propto P(R_\Omega | X_\Omega, \mu) P(X_\Omega | U, V, \sigma^2) P\big(U | \sigma_U^2\big) P\big(V | \sigma_V^2\big) \\
&\quad \int_{X_{\bar\Omega}} P(R_{\bar\Omega} | X_{\bar\Omega}, \mu) P(X_{\bar\Omega} | U, V, \sigma^2) dX_{\bar\Omega}.
\end{aligned} \tag{13}$$

The first equation is to separate the response patterns based on the observed ratings and unobserved ratings, while marginalizing the unobserved ratings. The second equation is to factorize the joint probability based on the graphical model in Fig. 2. The last expression in Eq. (13) takes the parts related to the observed ratings out of the marginalization on the unobserved ratings and set the probability $P(\mu)$ to a constant, which has proved its effectiveness in [22].

Based on the step function assumption on the response in Eq. (10), we can marginalize the unobserved ratings in Eq. (13) as follows:

$$\begin{aligned}
S_{\bar\Omega} &= \int_{X_{\bar\Omega}} P(R_{\bar\Omega} | X_{\bar\Omega}, \mu) P(X_{\bar\Omega} | U, V, \sigma^2) dX_{\bar\Omega} \\
&= \prod_{(i,j)\in\bar\Omega} \sum_{k\in\mathbb{D}} (1 - g(\mu_k)) \psi(i, j, k),
\end{aligned} \tag{14}$$

where $\psi(i,j,k) = \Phi(\frac{U_i^T V_j - k + 1}{\sigma}) - \Phi(\frac{U_i^T V_j - k}{\sigma})$ defines the difference of cumulative distribution function on the standard normal distribution with the center in one unit difference and $\Phi(z)$ is the cumulative distribution function for the standard normal Gaussian distribution:

$$\Phi(z) = \Pr(\{\mathcal{N}(0,1) \le z\}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-s^2/2} ds\,.$$

Substituting the data generation model on observed ratings in Eq. (7) and Eq. (8), and the response pattern assumption on the observed ratings in Eq. (10), and the marginalization of unobserved ratings in Eq. (14) into the posterior probability in Eq. (13) and taking the log-likelihood, we obtain the log-likelihood function on U, V, and μ as follows:

$$\begin{aligned}
\mathcal{L}(U, V, \mu) = &\sum_{(i,j)\in\Omega} \log g(\mu_{X_{ij}}) - \frac{1}{2\sigma^2} (U_i^T V_j - X_{ij})^2 \\
&- \frac{\|U\|_F^2}{2\sigma_U^2} - \frac{\|V\|_F^2}{2\sigma_V^2} + \sum_{(i,j)\in\bar\Omega} \log\Big(S_{\bar\Omega_{ij}}\Big),
\end{aligned} \tag{15}$$

where $S_{\bar\Omega_{ij}}$ is calculated by $\sum_{k\in\mathbb{D}}(1 - g(\mu_k))(\Phi(\frac{U_i^T V_j - k + 1}{\sigma}) - \Phi(\frac{U_i^T V_j - k}{\sigma}))$.

We can then obtain the gradients of the log-likelihood, $\mathcal{L}$, with respect to U_i and V_j in Eq. (16) and Eq. (17), respectively. The gradient of $\mathcal{L}$ with respect to μ_k is calculated by

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial U_i} = &- \sum_{j,(i,j)\in\Omega} \frac{1}{\sigma^2} (U_i^T V_j - X_{ij}) V_j - \frac{1}{\sigma_U^2} U_i + \sum_{j,(i,j)\in\bar\Omega} \\
&\frac{\sum_{k\in\mathbb{D}} (1 - g(\mu_k)) \left(\exp\left\{ -\frac{(U_i^T V_j - k + 1)^2}{2\sigma^2} \right\} - \exp\left\{ -\frac{(U_i^T V_j - k)^2}{2\sigma^2} \right\} \right) V_j}{\sigma\sqrt{2\pi} S_{\bar\Omega_{ij}}},
\end{aligned} \tag{16}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial V_j} = &- \sum_{i,(i,j)\in\Omega} \frac{1}{\sigma^2} (U_i^T V_j - X_{ij}) U_i - \frac{1}{\sigma_V^2} V_j + \sum_{i,(i,j)\in\bar\Omega} \\
&\frac{\sum_{k\in\mathbb{D}} (1 - g(\mu_k)) \left(\exp\left\{ -\frac{(U_i^T V_j - k + 1)^2}{2\sigma^2} \right\} - \exp\left\{ -\frac{(U_i^T V_j - k)^2}{2\sigma^2} \right\} \right) U_i}{\sigma\sqrt{2\pi} S_{\bar\Omega_{ij}}}.
\end{aligned} \tag{17}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mu_k} = &\frac{g'(\mu_k)}{g(\mu_k)} \sum_{(i,j)\in\Omega} \mathbf{1}(X_{ij} = k) \\
&+ \sum_{(i,j)\in\bar\Omega} \frac{g'(\mu_k) \left(\Phi\left(\frac{U_i^T V_j - k}{\sigma} \right) - \Phi\left(\frac{U_i^T V_j - k + 1}{\sigma} \right) \right)}{S_{\bar\Omega_{ij}}},
\end{aligned} \tag{18}$$

where $\mathbf{1}(X_{ij} = k)$ is equal to 1 if $X_{ij} = k$ and 0 for other cases.

To learn the above parameters, U, V, and μ, we can update U, V, and μ alternatively using the gradient ascent algorithm with a learning rate η by maximizing the log-likelihood as follows:

$$U_i \leftarrow U_i + \eta \frac{\partial \mathcal{L}}{\partial U_i}, \qquad V_j \leftarrow V_j + \eta \frac{\partial \mathcal{L}}{\partial V_j}. \tag{19}$$

Then using the updated U and V, we update μ_k by

$$\mu_k \leftarrow \mu_k + \eta \frac{\partial \mathcal{L}}{\partial \mu_k}. \tag{20}$$

Similar to PMF [32], we linearly map the rating values in $[1, D]$ to $[0, 1]$ and pass $U_i^T V_j$ through the sigmoid function as defined in Eq. (11). To avoid cluttered notations, we drop all the logistic function in our derivation process. After obtaining the trained model, we convert the expected value, $g(U_i^T V_j)$, back to the scale of 1 to D and set it as the predicted score of user i's rating on item j. Hence, we summarize the algorithm for rating dominant response aware PMF in Algorithm 1.

Algorithm 1. Rating Dominant Response Aware PMF (RAPMF-r)

1: **Parameters**: $N, M, D, K, \sigma, \sigma_U, \sigma_V$
2: **Input**: Partially observed ratings, X_Ω and response matrix, R
3: **Initialize** $U \in \mathbb{R}^{K \times N}$, $V \in \mathbb{R}^{K \times M}$, $\mu \in \mathbb{R}^D$ randomly
4: **while** stop criteria not met **do**
5: Update U_i and V_j: $U_i \leftarrow U_i + \eta \frac{\partial \mathcal{L}}{\partial U_i}$, $V_j \leftarrow V_j + \eta \frac{\partial \mathcal{L}}{\partial V_j}$.
6: Update μ: $\mu_k \leftarrow \mu_k + \eta \frac{\partial \mathcal{L}}{\partial \mu_k}$.
7: **end while**

3.3 Complexity Analysis

In Algorithm 1, to update the latent feature of a user, we need to sum all items together. Hence, its computation cost is $O(M)$. To update all the users, we have to run through all users at least once, which yields the cost of $O(NM)$. This is quite time consuming compared with that of PMF, which is linear in the number of observations, $O(|\Omega|)$. However, we argue that the time spent on training is worthy since it can boost the model performance. More importantly, the prediction complexity of RAPMF is the same as that of PMF, $O(K)$, which can be taken as a constant time given a moderate sized K. Since the training procedure can be performed offline, RAPMF can accommodate the hard response time constraint in real-world recommender systems due to the succinct prediction cost.

4 Mini-Batch Learning

To speed up the computation of RAPMF, we adopt a mini-batch learning implementation. The main steps include

- First, we divide the response matrix into blocks each with B users and their corresponding B items. We denote the corresponding mini-batch index set, $A = \text{Block}_N(B) \times \text{Block}_M(B)$, where $\text{Block}_N(B)$ indicates the set of the selected B indexes from all users and likewise for $\text{Block}_M(B)$.
- Second, we update the corresponding U_i and V_j in the mini-batch set A. The corresponding updating rule for a user is just to replace the index of Ω and $\bar{\Omega}$ in Eq. (16) by A_Ω and $A_{\bar{\Omega}}$, respectively, where A_Ω and $A_{\bar{\Omega}}$ are the observed ratings and unobserved ratings in the set A, respectively. The updating rule of an item is changed similarly.
- When the user latent matrix and the item latent matrix are updated, we update the rating probability μ by (18), where Ω and $\bar{\Omega}$ are replaced by A_Ω and $A_{\bar{\Omega}}$, respectively.

Similar to the analysis in [32], the updating of our proposed mini-batch learning takes time linear in $O(B^2)$. When B is small, the updating is very efficient. We argue that we still can attain good performance due to the sparse nature of the data. This is verified in Section 5.

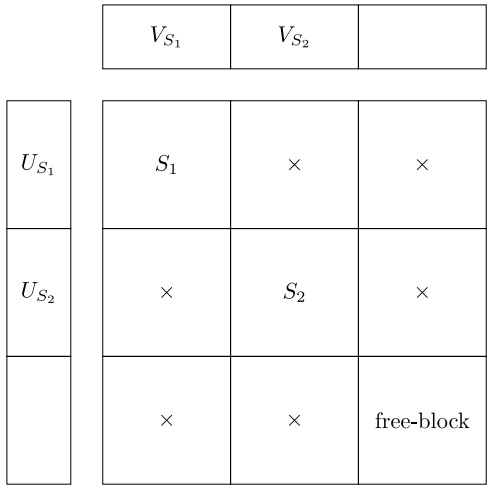


Fig. 4. Illustration about scheduling policy of our parallel mini-batch implementation. If S_1 and S_2 are assigned to different threads by the scheduler, only one free-block is left in the task queue managed by the scheduler.

However, in parallel execution, one serious issue is that if two threads update the blocks with the same set of users $\text{Block}_N(B)$ or items $\text{Block}_M(B)$, they may yield inconsistency of U and V as they update the same indexes of latent variables U or V simultaneously. To avoid updating inconsistency and maintaining the efficiency, we borrow the idea of free-block from [49] and propose a precise scheduling policy.

First, we define a *free-block* if the block does not share any users or items with all other blocks being executed. Fig. 4 gives an example of the scheduling with two parallel threads. Suppose S_1 and S_2 are two blocks executing in two parallel threads simultaneously. The schedule is performed as follows: when a thread finishes the updating in a block, e.g., U_{S_1} and V_{S_1}, it will assign a new free-block with the smallest updating count among all free blocks. It is observed that all the executing blocks and free-blocks do not share the same users or items. Hence, when a thread finishes the execution, it can fetch a mini-batch from the free-blocks to update the corresponding user latent matrices and item latent matrices. The updating can be continuously executed without waiting. Therefore, we can conduct further speedup on the execution. To meet this scheduling policy, the max number of threads is bounded as $\min\{\lfloor N/B \rfloor, \lfloor M/B \rfloor\} - 1$. Algorithm 2 sketches the procedure of our proposed mini-batch implementation in parallel version.

5 Experiments and Results

We conduct empirical evaluation on both a synthetic dataset and a real-world dataset, the Yahoo!Music's LaunchCast dataset, in a server with an eight-core 2.5 GHz Intel Xeon processor and 64 GB main memory running Ubuntu 13.10 operating system. We compare the performance of the following models:

- Probability matrix factorization [32] with the objective function shown in Eq. (9);

- the multinomial mixture model combined with conditional probability tables with Bernoulli distribution parameterized by the ratings, namely CPT-v [26];
- the multinomial mixture model combined with conditional probability tables with Bernoulli distribution parameterized in a logistic form, namely Logit-vd [25];
- SVD++ with implicit feedback [16]; and
- our RAPMF[1] shown in Algorithm 1, namely RAPMF–r, and its mini-batch implementation in Algorithm 2, namely RAPMF–rmb, respectively.

Algorithm 2. Parallel Mini-Batch Learning for RAPMF-r (RAPMF-rmb)

Parameters: $N, M, D, K, B, \sigma, \sigma_U, \sigma_V$
Input: Partially observed ratings, X_Ω, and response matrix, R
Initialize $U \in \mathbb{R}^{K\times N}$, $V \in \mathbb{R}^{K\times M}$, $\mu \in \mathbb{R}^D$ randomly
Grid the response matrix R into $B \times B$ blocks
  Compute the number of threads $P = \min\{\lfloor N/B \rfloor, \lfloor M/B \rfloor\} - 1$
while stop criteria not met **do**
  for $p = \{1, \ldots, P\}$ parallelly **do**
    Obtain a $B \times B$ block from the scheduler
    Update U_i and V_j within the block: $U_i \leftarrow U_i + \eta \frac{\partial \mathcal{L}}{\partial U_i}$, $V_j \leftarrow V_j + \eta \frac{\partial \mathcal{L}}{\partial V_j}$.
    Update μ: $\mu_k \leftarrow \mu_k + \eta \frac{\partial \mathcal{L}}{\partial \mu_k}$.
  end for
end while

We try to answer the following questions:

1) How to design experiment protocols to evaluate the performance of the models with and without response models fairly?
2) What is the performance of these models on the datasets?
3) How do the parameters affect the performance of RAPMF-r?
4) What is the convergence property of RAPMF-r?
5) How efficient is the mini-batch learning implementation for RAPMF-r?

5.1 Evaluation Metrics

In a recommender system deployed in the real world, there are exactly three types of relations regarding an item to a user: *un-inspected*, *inspected-unrated*, and *inspected-rated*. Traditional collaborative filtering approaches only focus on inspected-rated data since they do not consider the response patterns. These methods usually separate the inspected-rated data into a training set and a test set and evaluate the model on the test set. Since both the training set and the test set belong to the inspected-rated type, their rating distributions are the same. However, in real-world recommender systems, many items may be inspected but unrated. Users' response patterns also reveal users' preferences implicitly. Hence, the traditional evaluation scheme may undermine the significance of the missing response patterns. To explore the difference, we will investigate different experimental protocols as follows:

- *Traditional protocol.* Both the training set and the test set are randomly selected from inspected-rated items together with the users who have rated them and the corresponding rating scores. This is exactly the traditional experimental protocol [32], which ignores the response patterns.
- *Realistic protocol.* The training set is randomly selected from inspected-rated items, but the test set is randomly selected from un-inspected items. This is an experimental protocol adopted in [25], [26]. This protocol captures the ultimate goal of a recommender system, i.e., recommending un-inspected items to potential users who are interested.

Moreover, we will design a new experimental protocol to test the model performance when the distributions of training set and test set are divergent, or even complementary:

- *Adversarial protocol.* The training set is randomly selected from inspected-rated items, but the test set is randomly selected from inspected-unrated items. This protocol clearly shows the divergence of the training set and the test set since in real-world systems, e.g., the Yahoo!Music's LaunchCast service, most of the inspected-rated items receive very high scores, while those inspected-unrated items have low scores or average scores. This protocol can evaluate the performance of models in this scenario whether they can capture users' response behaviors.

In the experiment, we use root mean square error (RMSE), a popular metric in collaborative filtering [26], [32], [42], to evaluate the performance of different models. That is, $\text{RMSE} = \sqrt{\frac{1}{|\mathcal{T}|}\sum_{(i,j,x)\in\mathcal{T}}(\hat{x}_{ij} - x)^2}$, where $\mathcal{T}$ is the set of (i, j, x) triplets reserved for testing and $\hat{x}_{ij}$ is the model prediction for user i's rating on item j.

5.2 Datasets

We conduct empirical evaluation on both a synthetic dataset and two real-world datasets. The synthetic dataset provides both benchmark information on user-item ratings and user-item response patterns. Hence, we can use it to evaluate all the compared models under all three evaluation protocols. The real-world Yahoo! dataset is collected from Yahoo! Music's LaunchCast Radio service and is particularly prepared for evaluating the *realistic protocol* in real-world recommender systems [26]. For other benchmark real-world datasets, we select MovieLens to evaluate the performance on traditional protocol.

Synthetic dataset. The data generation process consists of generating a full rating matrix and the corresponding response matrix. To generate the full rating matrix, we first generate the latent user features and item features from zero-mean spherical Gaussian by $U_i \sim \mathcal{N}(\mathbf{0}_K, \sigma_U^2\mathbf{I}_K), V_j \sim \mathcal{N}(\mathbf{0}_K, \sigma_V^2\mathbf{I}_K)$, where $i = 1, \ldots, N$, $j = 1, \ldots, M$, $\mathbf{0}_K$ is a K-dimensional vector with each element being 0 and $\mathbf{I}_K$ is the $K \times K$ identity matrix, and $K = 5$. The full rating matrix X

1. Our codes can be downloaded in https://www.dropbox.com/s/4h2tql4d1hk1f8s/rapmf.tar.gz

TABLE 2
Parameters for Generating the Synthetic Dataset

N	M	D	K	$P_{inspect}$
1,000	1,000	5	5	0.2
P_1	P_2	P_3	P_4	P_5
0.073	0.068	0.163	0.308	0.931

is then obtained by re-scaling the sigmoid value of U^TV to 1 to D by $X_{ij} = \lceil g(U_i^T V_j) \times D \rceil$.

To generate the response matrix R, we first set the inspection probability of a user inspecting an item, $P_{inspect}$. Then, the partitioning of inspected ratings and un-inspected ratings is done by the Bernoulli trials with success probability $P_{inspect}$. For all the inspected ratings, we model their response probability by a Bernoulli distribution with the success probability P_k, where $k \in \{1, 2, \ldots, D\}$. Table 2 summarizes the parameters used for generating the synthetic dataset. The parameters are selected according to Fig. 1, where high scores represent high response probability. They are selected to faithfully simulate real users' ratings and response behaviors.

To minimize the effect of randomness, we generate the dataset independently 10 times and test them to average the results. On average, we provide about 3.3 percent of the full matrix as the training set, around 3.4 percent as the test set for traditional protocol, around 17.3 percent as the test set for adversarial protocol, and all the remaining 80 percent as the test set for realistic protocol.

Yahoo! dataset. The Yahoo! dataset[2] consists of *Yahoo! Music ratings for User Selected and Randomly Selected songs, version 1.0*. More specifically, it contains 311,704 ratings collected from 15,400 users on 1,000 songs during the normal interaction between the users and the Yahoo!Music system, with at least 10 ratings for each user. During a survey conducted by Yahoo!Research, exactly 10 songs randomly selected from these 1,000 songs are presented to the users to listen and rate. In total there are 5,400 users participating this survey and the resulting 54,000 ratings are the *survey* ratings. It is noted that in [26], by carefully designing, the 1,000 surveyed songs are assigned to at least one user.

To get a complete evaluation, we select the ratings initially rated by the 5,400 surveyed users and the songs are in the set of the randomly selected 1,000 surveyed songs. We then construct a *user intention (UI)* set, which consists of 129,179 ratings and yields the density about 2.39 percent. The distributions of the rating scores in the UI set and the survey set are dramatically different, as seen in Fig. 1. In evaluating different protocols, we adopt different settings. To evaluate the models under the traditional protocol, we have to select ratings following the same distribution. Hence, we select ratings only from the UI set and separate them as training set and test set in the ratio of 4 to 1. That is, 80 percent of data from the UI set are used in training the models while the rest 20 percent are for the test. For the realistic protocol, we use all the data from the UI set for training and use the ratings from the survey set as test set, since these two datasets are dramatically different and contain the benchmark of users' response patterns.

MovieLens dataset. The selected MovieLens 100 K dataset[3] consists of 100,000 ratings from 1,000 users on 1,700 movies. Here, we focus on evaluating the performance on the traditional protocol. We use the default splitting provided in the original dataset for training and test.

5.3 Settings

We tune the parameters to attain good performance via cross-validation on a validation set. The parameters include

- K: For simplicity, we set $K = 5$ for the synthetic dataset and tune it for the other two real-world datasets.
- $\lambda_U = \frac{\sigma^2}{\sigma_U^2}$ and $\lambda_V = \frac{\sigma^2}{\sigma_V^2}$: they are first tuned by the grid search in the range of $\{10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10, 10^2\}$ and fine-tune to achieve the best performance of PMF.
- σ: We first fix the optimal λ_U and λ_V obtained from PMF and then tune it in $\{10^{-3}, 10^{-2}, 10^{-1}, 1.0\}$ and fine-tune it further for RAPMF-r; see sensitivity analysis results in Fig. 5.

For CPT-v and Logit-vd, the hyper-parameters follow the tuning in [25].

5.4 Model Performance

We report the average performance for all datasets in Table 3. For the synthetic dataset, we run 10 independent trials and test them on three protocols. For the Yahoo! dataset, we run 10 independent trials on the traditional protocol and 10 different initialization on the realistic protocol. We do not test it under the adversarial protocol because we do not have the inspected-unrated information. For the MovieLens dataset, we only test the traditional protocol on the default five-fold separating data. From Table 3, we have the following observations:

- For the synthetic dataset, our proposed RAPMF-r attains significantly better performance than other benchmark methods under most protocols. More significantly, when CPT-v and Logit-vd cannot beat PMF under the adversarial protocol, our RAPMF-r can obtain at least 37.0 percent improvement over PMF. SVD++ can achieve slightly better performance under the adversarial protocol. By observing the detailed results, we find that the results of SVD++ fluctuate and obtain larger standard deviation, 0.042, than that of RAPMF-r, 0.01. Our mini-batch implementation, RAPMF-rmb, can achieve similar performance as the RAPMF-r.
- For the Yahoo! dataset, our proposed RAMPF-r also beats other methods, including SVD++. The mini-batch implementation can attain nearly the same RMSE as the original implementation.
- For the MovieLens dataset, our RAPMF-r attains at least 14.1 percent improvement over PMF, CPT-v,

2. http://webscope.sandbox.yahoo.com

3. http://grouplens.org/datasets/movielens/

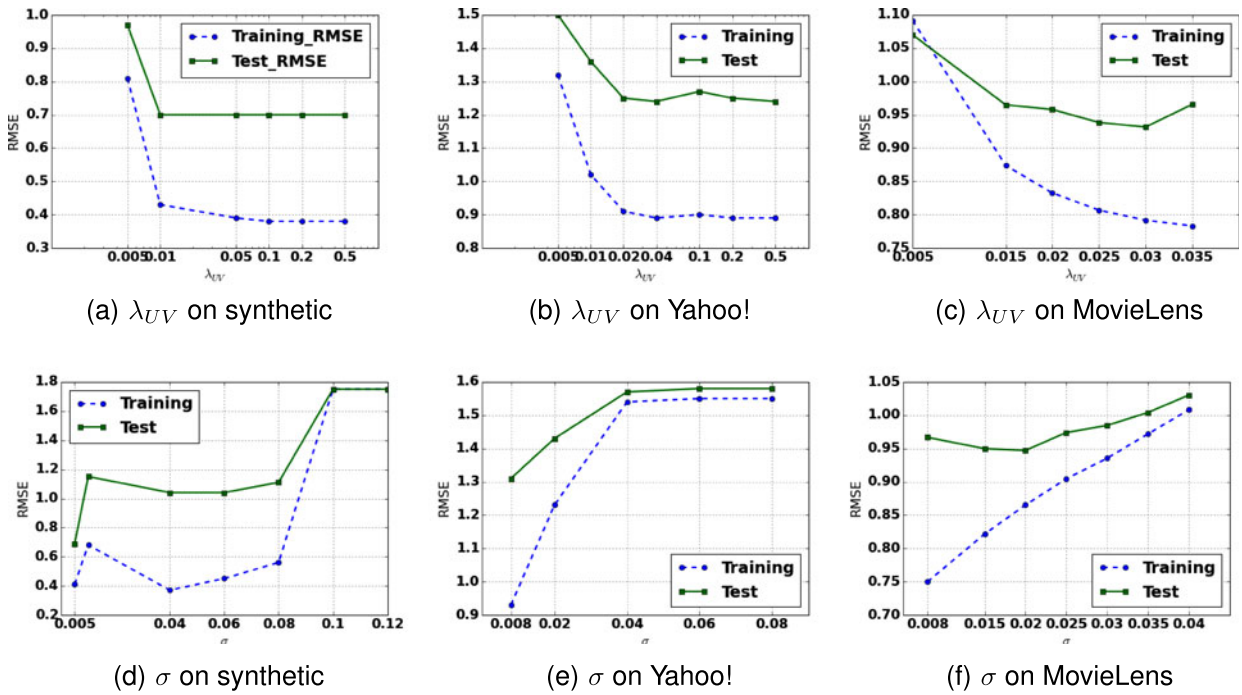


Fig. 5. Sensitivity analysis of hyper-parameters of RAPMF-r on the synthetic, Yahoo!, and MovieLens datasets for one-trial test.

and Logit-vd. RAPMF-r only achieves slightly worse performance than SVD++, a state-of-the-art collaborative filtering model for recommender system under the traditional protocol. However, we also observe that SVD++ is very sensitive to the implicit regularization parameter. The mini-batch implementation of RAPMF-r, i.e., RAPMF-rmb, achieves 0.969 RMSE, nearly the same as that of RAPMF-r at 0.962.

- The above results imply that when we want to apply our proposed RAPMF-r in real-world systems, we only need to tune the model which achieves the best cross-validation performance on the training set.

Moreover, we also observe that the learned response probabilities, i.e., the learned Bernoulli parameters for the five grades, are $[0.0232, 0.0052, 0.0089, 0.0149, 0.9928]$, for a typical run of RAPMF-r on the synthetic dataset. Obviously, the values precisely follow the trend of the parameters used in Table 2. This again explains the significant performance boost for the synthetic dataset under the realistic and the adversarial protocols.

5.5 Sensitivity Analysis

In the following, we investigate how the model parameters affect the performance of RAPMF. All the sensitivity analysis is done under the realistic setting in one-trial.

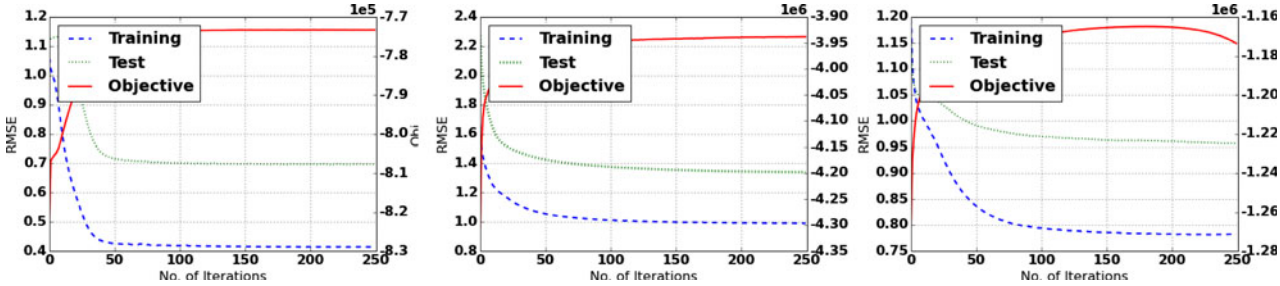
Impact of λ_U, λ_V. As observed in [22], regularization on μ plays not much effect. Hence, in this work, we only place the regularization parameters λ on U, V. Since in the dataset, users and items are symmetric, we use the same regularization parameter λ_{UV} for U and V.

Figs. 5a, 5b, and 5c show the impact of λ_{UV} on the performance of RAPMF-r for the synthetic, Yahoo!, and MovieLens dataset, respectively. It is observed that both the curves of training RMSE and test RMSE change similarly. When λ_{UV} is small, RAPMF-r cannot generalize well. When it becomes larger, the performance becomes smooth. For the synthetic dataset, when $\lambda_{UV} = 0.1$, RAPMF-r attains the best performance on both training set and test set. For the Yahoo! dataset, when $\lambda_{UV} = 0.04$, RAPMF-r attains the best performance on both training set and test set. For the MovieLens dataset, RAPMF-r attains the best performance on the test set when $\lambda_{UV} = 0.3$.

TABLE 3
Results of the Compared Models on the Synthetic, Y, and MovieLens Datasets

Dataset	Synthetic						Yahoo!				MovieLens	
Method	Traditional	Imp.	Realistic	Imp.	Adversarial	Imp.	Traditional	Imp.	Realistic	Imp.	Traditional	Imp.
PMF	1.109 ± 0.005	58.7	1.507 ± 0.01	51.5	1.534 ± 0.002	37.0	1.580 ± 0.004	18.9	1.592 ± 0.003	28.3	1.182 ± 0.041	22.9
CPT-v	1.108 ± 0.015	58.5	1.482 ± 0.020	48.9	1.642 ± 0.018	46.6	1.801 ± 0.005	35.5	1.793 ± 0.006	44.5	1.168 ± 0.073	21.4
Logit-vd	1.020 ± 0.009	45.9	1.440 ± 0.013	44.7	1.629 ± 0.013	45.4	1.813 ± 0.004	36.4	1.790 ± 0.003	44.2	1.098 ± 0.056	14.1
SVD++	0.747 ± 0.012	6.9	0.999 ± 0.017	0.4	**1.058** ± 0.042	-5.5	1.330 ± 0.008	0.1	1.295 ± 0.011	4.4	**0.938** ± 0.090	-2.5
RAPMF-r	**0.699** ± 0.004	–	**0.995** ± 0.007	–	1.120 ± 0.010	–	**1.329** ± 0.003	–	**1.241** ± 0.003	–	0.962 ± 0.095	–
RAPMF-rmb	0.700 ± 0.016	0.1	1.043 ± 0.012	4.8	1.182 ± 0.007	5.5	1.331 ± 0.004	0.2	1.243 ± 0.003	0.2	0.969 ± 0.088	0.7

The improvement (Imp.) indicates the improvement of RAPMF-r over other models in percentage (%).



(a) Synthetic (b) Yahoo! (c) MovieLens

Fig. 6. Convergence analysis of RAPMF-r on the the three evaluation datasets for one-trial test.

Impact of σ. As shown in Fig. 2, the parameter σ is the standard deviation of the noise. It defines the spread of the data. From Eq. (15), when we multiply σ^2 for the log-likelihood, it can also control the weight of marginalization on the unobserved data.

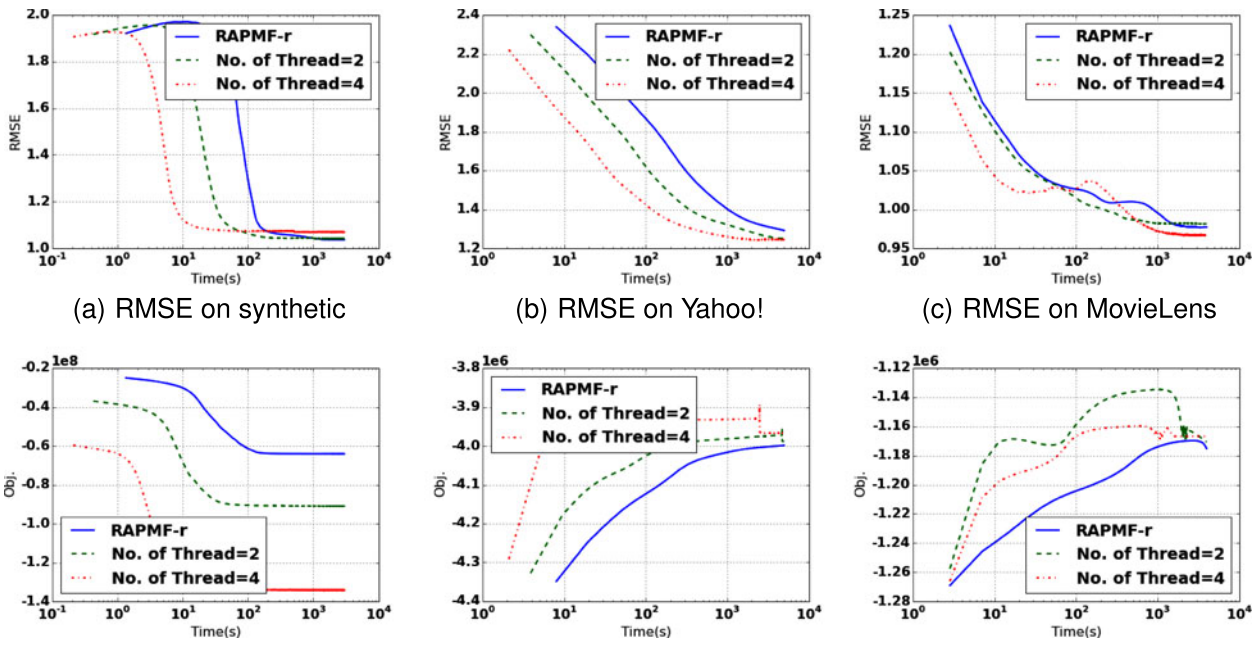
Figs. 5d, 5e, and 5f plot the performance of RAPMF-r versus the logarithmic scale of σ on the synthetic, Yahoo!, and MovieLens dataset, respectively. Overall, as σ increases, the training RMSE and the test RMSE increase accordingly. For the synthetic dataset, Fig. 5d shows that the best performance attains when $\sigma = 0.04$, while it is 0.008 for the Yahoo! dataset, and 0.02 for the MovieLens dataset. From Fig. 5e, one may conjecture that the RMSE may be smaller when σ is smaller than 0.008. However, our experiment finds that when σ is smaller than 0.008, the algorithm diverges and attain poor performance.

5.6 Convergence Analysis of Batched Training

In the following, we analyze how RAPMF-r converge in one trial on all the evaluation datasets. In the experiments, we have set the maximum number of iteration as 250, a suitable value for the convergence. From the figures shown in Fig. 6, we can see that RAPMF-r on the synthetic and Yahoo! datasets increase the objective (log-likelihood) values rapidly and the values become stable soon, while for the MovieLens dataset, the objective values follow similar trend and becomes decrease when the number of iterations increases. By examining the performance on the three datasets in Figs. 6a, 6b, and 6c, we can see that RMSEs on both the training set and the test set decrease gradually when the number of iterations increases and become stable when the number of iterations is greater than a certain value.

5.7 Efficiency of Mini-Batched Learning

In the following, we demonstrate the correctness and speedup of our mini-batch learning approach. In the experiment, we employ the maximum number of threads computed from $\min\{\lfloor N/B \rfloor, \lfloor M/B \rfloor\} - 1$ to run the mini-batch program. B is the number of users and items in a mini-block of training data. To avoid the influence from the uncertain scheduling time in operating system, we average 10 runs of our results with the same configuration. Fig. 7 shows



(a) RMSE on synthetic (b) RMSE on Yahoo! (c) RMSE on MovieLens

(d) Obj. on synthetic (e) Obj. on Yahoo! (f) Obj. on MovieLens

Fig. 7. Convergence curves of RMSE and objective function values of RAPMF-r and the mini-batch implementation on three evaluation datasets.

the convergence of RMSE and objective function values on three evaluation datasets. We have the following observations:

- As the number of threads increases, RAPMF converges faster. That is, RMSEs attain its local minimal and the objective function values achieve its local maximum much faster than the batch-trained RAPMF-r. This reveals that our mini-batch implementation can take the advantage of multi-thread technique to speed up the training process which is usually a time-consuming task in many similar approaches [26], [32].
- RMSEs of the mini-batch implementation of RAPMF-r on the test set is slightly worse than that of batched RAPMF. The error brought by mini-batch learning may be caused by the in-consistent of latent variables U and V in the mini-batch learning. This situation is inevitable for all mini-batch learning [41].

6 Conclusion and Future Work

In this paper, we establish a response aware probabilistic matrix factorization framework to unify users' response behaviors and a popular model-based collaborative filtering technique, probabilistic matrix factorization. More specifically, we model the probability of whether a user will rate an item by engaging the Bernoulli distribution, where the parameters are determined by the rating scores, and assume the corresponding probability of unobserved ratings as a step function. More significantly, we speed up the algorithm by mini-batch implementation and conduct a crafting scheduling policy on automatically selecting free-blocks to further accelerate the original batch-trained RAPMF algorithm. Empirically, we verify the performance of RAPMF under carefully designed experimental protocols and show that RAPMF performs best when it tries to fulfill the ultimate goal of real-world recommender systems, i.e., recommending items to those users who do not see the items before, but may be interested in them. The empirical evaluation demonstrates the potential of our RAPMF model in real-world recommender system deployment.

There are several interesting directions worthy of future consideration. The first direction is to incorporate other side information to boost the model performance. Another promising avenue is to investigate how to model the response when the response patterns are hidden. For example, in location-based social network applications such as Foursquare and Google Latitude, the check-in frequency can be obtained explicitly, but the response regarding whether a user will check-in a place or not is unknown. The third direction is to design a smart way in efficiently tuning the hyper-parameters or to design a learning scheme in obtaining the model parameters automatically.

Acknowledgments

The work described in this paper was fully supported by the National Grand Fundamental Research 973 Program of China (No. 2014CB340401 and No. 2014CB340405), the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Nos. CUHK 413212 and CUHK 415113), and Microsoft Research Asia Regional Seed Fund in Big Data Research (Grant No. FY13-RES-SPONSOR-036). Haiqin Yang is the corresponding author.

References

[1] J. Abernethy, K. Canini, J. Langford, and A. Simma, "Online collaborative filtering," Department of Computer Science, Univ. California at Berkeley, Berkeley, CA, USA, 2007.
[2] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.
[3] J. S. Breese, D. Heckerman, and C. M. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. 14th Conf. Uncertainty Artif. Intell.*, 1998, pp. 43–52.
[4] J. Canny, "Collaborative filtering with privacy via factor analysis," in *Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Tampere, Finland, 2002, pp. 238–245.
[5] C. Cheng, H. Yang, , M. R. Lyu, and I. King, "Where you like to go next: Successive point-of-interest recommendation," in *Proc. 23rd Int. Joint Conf. Artif. Intell.*, Beijing, China, 2013, pp. 2605–2611.
[6] C. Cheng, H. Yang, I. King, and M. R. Lyu, "Fused matrix factorization with geographical and social influence in location-based social networks," in *Proc. 6th AAAI Conf. Artif. Intell.*, Toronto, ON, Canada, 2012, pp. 17–23.
[7] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learning Res.*, vol. 7, pp. 551–585, 2006.
[8] A. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 271–280.
[9] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, pp. 143–177, Jan. 2004.
[10] E. F. Harrington, "Online ranking/collaborative filtering using the perceptron algorithm," in *Proc. Int. Conf. Mach. Learning*, 2003, pp. 250–257.
[11] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 1999, pp. 230–237.
[12] T. Hofmann, "Collaborative filtering via gaussian probabilistic latent semantic analysis," in *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2003, pp. 259–266.
[13] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 89–115, 2004.
[14] Z. Huang, H. Chen, and D. Zeng, "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 116–142, 2004.
[15] R. Jin, L. Si, and C. Zhai, "Preference-based graphic models for collaborative filtering," in *Proc. 19th Conf. Uncertainty Artif. Intell.*, 2003, pp. 329–336.
[16] Y. Koren, "Collaborative filtering with temporal dynamics," *Commun. ACM*, vol. 53, no. 4, pp. 89–97, 2010.
[17] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Trans. Knowl. Discovery Data*, vol. 4, no. 1, pp. 1:1–1:24, Jan. 2010.
[18] Y. Koren, R. M. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Comput.*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[19] Y. Li, J. Hu, C. Zhai, and Y. Chen, "Improving one-class collaborative filtering by incorporating rich user information," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage.*, 2010, pp. 959–968.
[20] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan. 2003.
[21] G. Ling, H. Yang, I. King, and M. R. Lyu, "Online learning for collaborative filtering," in *Proc. IEEE World Congr. Comput. Intell.*, 2012, pp. 1–8.
[22] G. Ling, H. Yang, I. King, and M. R. Lyu, "Response aware model-based collaborative filtering," in *Proc. 28th Conf. Uncertainty Artif. Intell.*, 2012, pp. 501–510.
[23] R. J. A. Little and D. B. Rubin, *Statistical Analysis With Missing Data*, 2 ed. New York, NY, USA: Wiley-Interscience, Sept. 2002.

[24] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proc. 4th ACM Int. Conf. Web Search Data Mining*, 2011, pp. 287–296.
[25] B. M. Marlin and R. S. Zemel, "Collaborative prediction and ranking with non-random missing data," in *Proc. 3rd ACM Conf. Recommender Syst.*, 2009, pp. 5–12.
[26] B. M. Marlin, R. S. Zemel, S. T. Roweis, and M. Slaney, "Collaborative filtering and the missing at random assumption," in *Proc. 23rd Conf. Uncertainty Artif. Intell.*, 2007, pp. 267–275.
[27] R. Pan and M. Scholz, "Mind the gaps: Weighting the unknown in large-scale one-class collaborative filtering," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 667–676.
[28] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. M. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *Proc. IEEE 8th Int. Conf. Data Mining*, 2008, pp. 502–511.
[29] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," in *Proc. 3rd ACM Int. Conf. Web Search Data Mining*, 2010, pp. 81–90.
[30] J. D. M. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proc. 22nd Int. Conf. Mach. Learning*, 2005, pp. 713–719.
[31] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An open architecture for collaborative filtering of netnews," in *Proc. Conf. Comput. Supported Cooperative Work*, 1994, pp. 175–186.
[32] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1257–1264.
[33] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *Proc. 25th Int. Conf. Mach. Learning*, 2008, pp. 880–887.
[34] R. Salakhutdinov, A. Mnih, and G. E. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proc. 24th Int. Conf. Mach. Learning*, 2007, pp. 791–798.
[35] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 285–295.
[36] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends. Mach. Learning*, vol. 4, no. 2, pp. 107–194, 2012.
[37] L. Si and R. Jin, "Flexible mixture model for collaborative filtering," in *Proc. Int. Conf. Mach. Learning*, 2003, pp. 704–711.
[38] H. Steck, "Training and testing of recommender systems on data missing not at random," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 713–722.
[39] H. Steck, "Evaluation of recommendations: Rating-prediction and ranking," in *Proc. 7th ACM Conf. Recommender Syst.*, 2013, pp. 213–220.
[40] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. Artif. Intell.*, vol. 2009, 2009, pp. 1–19.
[41] M. Takác, A. S. Bijral, P. Richtárik, and N. Srebro, "Mini-batch primal and dual methods for SVMs," in *Proc. Int. Conf. Mach. Learning*, 2013, pp. 1022–1030.
[42] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 448–456.
[43] J. Wang, S. C. H. Hoi, P. Zhao, and Z.-Y. Liu, "Online multi-task collaborative filtering for on-the-fly recommender systems," in *Proc. 7th ACM Conf. Recommender Syst.*, 2013, pp. 237–244.
[44] H. Yang, M. R. Lyu, and I. King, "Efficient online learning for multi-task feature selection," *ACM Trans. Knowl. Discovery Data*, vol. 7, 2013, pp. 6:1–27.
[45] H. Yang, Z. Xu, I. King, and M. R. Lyu, "Online learning for group lasso," in *Proc. 27th Int. Conf. Mach. Learning*, 2010, pp. 1191–1198.
[46] Y. Zhang, B. Cao, and D.-Y. Yeung, "Multi-domain collaborative filtering," in *Proc. 26th Conf. Uncertainty Artif. Intell.*, 2010, pp. 725–732.
[47] Y. Zhang and J. Koren, "Efficient Bayesian hierarchical user modeling for recommendation system," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Amsterdam, The Netherlands, 2007, pp. 47–54.
[48] P. Zhao, S. C. H. Hoi, and R. Jin, "DUOL: A double updating approach for online learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 2259–2267.
[49] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin, "A fast parallel SGD for matrix factorization in shared memory systems," in *Proc. 7th ACM Conf. Recommender Syst.*, 2013, pp. 249–256.

Haiqin Yang (M'11) received the BSc degree in computer science from Nanjing University, and the MPhil and PhD degrees in computer science and engineering from the Chinese University of Hong Kong, Hong Kong. His current research interests include machine learning, data mining, and big data analytics. He has published two books and more than 40 technical publications in journals/conferences in his areas of expertise. He has initiated and co-organized five international workshops on the topic of scalable machine learning and scalable data analytics. He also served as a program committee member and a reviewer of over 10 top-tier conferences/journals. He is a member of the IEEE.

Guang Ling received the BSc degree in computer science from the Chinese University of Hong Kong, Hong Kong, where he is currently working toward the PhD degree in computer science and engineering. His research interests include recommender systems, online learning, spammer detection, topic modeling, and machine learning in general. He had published technical publications in conferences/journals in his area of expertise. He had served as a reviewer or subreviewer in many top-tier conferences. He had also served as a student helper and helped co-organised conferences.

Yuxin Su received the BSc degree from Sun Yat-sen University, Guangzhou, China, and the MEng degree from Zhejiang University, Hangzhou, China. He is currently working toward the PhD degree in computer science and engineering in The Chinese University of Hong Kong, Hong Kong. His current research interests include large-scale machine learning methods in distributed environment and general system framework for big data analytics. His previous research interest is synchronization problems in complex networks. He has published 10 technical publications in journals/conferences in his areas of expertise.

Michael R. Lyu (F'04) received the BS degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, the MS degree in computer engineering from the University of California, Santa Barbara, and the PhD degree in computer engineering from the University of California, Los Angeles. He is a professor in the Computer Science and Engineering Department, Chinese University of Hong Kong, Hong Kong. He was at the Jet Propulsion Laboratory, Pasadena, CA, Bellcore, Piscataway, NJ, and the Bell Laboratory, Murray Hill, NJ, and taught at the University of Iowa, Iowa City. He has participated in more than 30 industrial projects. He has published close to 400 papers in the following areas. His current research interests include software engineering, distributed systems, multimedia technologies, machine learning, social computing, and mobile networks. He initiated the International Symposium on Software Reliability Engineering (ISSRE), and was a Program chair for ISSRE in 1996, the Program Co-chair for the 10th International World Web Conference, the Symposium on Reliable Distributed Systems in 2005, the International Conference on e-Business Engineering in 2007, and the International Conference on Services Computing in 2010. He was the General chair for ISSRE in 2001, the Pacific Rim International Symposium on Dependable Computing in 2005, and the International Conference on Dependable Systems and Networks in 2011. He also received the Best Paper Awards in ISSRE in 1998 and 2003, and the SigSoft Distinguished Paper Award in International Conference on Software Engineering in 2010. He has been named by the IEEE Reliability Society as the Reliability engineer of the Year in 2011, for his contributions to software reliability engineering and software fault tolerance. He is a fellow of the American Association for the Advancement of Science and the IEEE.

Irwin King (SM'08) received the BSc degree in engineering and applied science from the California Institute of Technology (Caltech), Pasadena, and the MSc and PhD degrees in computer science from the University of Southern California (USC), Los Angeles. He is an associate dean (education), Faculty of Engineering, and a professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. He was at AT&T Labs Research and also taught a number of courses at UC Berkeley as a Visiting professor. His research interests include machine learning, social computing, Big Data, web intelligence, data mining, and multimedia information processing. In these research areas, he has well over 200 technical publications in top international journals and conferences. In addition, he has contributed more than 30 book chapters and edited volumes. Moreover, he has over 30 research and applied grants and industry projects. Some notable projects include the VeriGuide system and the Knowledge and Education Exchange Platform (KEEP). He is an associate editor of the *ACM Transactions on Knowledge Discovery from Data (ACM TKDD)* and *Journal of Neural Networks*. Currently, he is serving as the vice-president and the Governing Board member of both the International Neural Network Society (INNS) and the Asian Pacific Neural Network Assembly (APNNA). He serves as the General Co-chair of WSDM2011, RecSys2013, and ACML2015. He is a senior member of the IEEE.