Scalable Variational Bayesian Matrix Factorization with Side Information

Yong-Deok Kim and Seungjin Choi
Department of Computer Science and Engineering
Pohang University of Science and Technology
77 Cheongam-ro, Nam-gu, Pohang 790-784, Korea
{karma13,seungjin}@postech.ac.kr

Abstract

Bayesian matrix factorization (BMF) is a popular method for collaborative prediction, because of its robustness to overfitting as well as of being free from cross-validation for fine tuning of regularization parameters. In practice, however, due to its cubic time complexity with respect to the rank of factor matrices, existing variational inference algorithms for BMF are not well suited to web-scale datasets where billions of ratings provided by millions of users are available. The time complexity even increases when the side information, such as user binary implicit feedback or item content information, is incorporated into variational Bayesian matrix factorization (VBMF). For instance, a state of the arts in VBMF with side information, is to place Gaussian priors on user and item factor matrices, where mean of each prior is regressed on the corresponding side information. Since this approach introduces additional cubic time complexity with respect to the size of feature vectors, the use of rich side information in a form of high-dimensional feature vector is prohibited. In this paper, we present a scalable inference for VBMF with side information, the complexity of which is linear in the rank K of factor matrices. Moreover, the algorithm can be easily parallelized on multi-core systems. Experiments on large-scale datasets demonstrate the useful behavior of our algorithm such as scalability, fast learning, and prediction accuracy.

Appearing in Proceedings of the 17^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2014, Reykjavik, Iceland. JMLR: W&CP volume 33.

1 INTRODUCTION

Matrix factorization refers to a method for uncovering a low-rank latent structure of data, approximating the data matrix as a product of two factor matrices. Matrix factorization is popular for collaborative prediction, where unknown ratings are predicted by user and item factor matrices which are determined to approximate a user-item matrix as their product [1, 2, 3, 4, 5, 6]. Suppose that $\boldsymbol{X} \in \mathbb{R}^{I \times J}$ is a user-item rating matrix, the (i,j)-entry of which, X_{ij}, represents the rating of user i on item j. Matrix factorization determines factor matrices $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_I] \in \mathbb{R}^{K \times I}$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_J] \in \mathbb{R}^{K \times J}$ (where K is the rank of factor matrices), to approximate the rating matrix $\boldsymbol{X}$ by $\boldsymbol{U}^\top \boldsymbol{V}$:

$$\boldsymbol{X} \approx \boldsymbol{U}^\top \boldsymbol{V}. \tag{1}$$

A popular approach is to minimize the regularized squared error loss defined as

$$\sum_{(i,j)\in\Omega} \left[(X_{ij} - \boldsymbol{u}_i^\top \boldsymbol{v}_j)^2 + \lambda(\|\boldsymbol{u}_i\|^2 + \|\boldsymbol{v}_j\|^2) \right], \tag{2}$$

where Ω is a set of indices of observed entries in $\boldsymbol{X}$, and λ is the regularization parameter. The problem (2) can be efficiently solved by alternating least sqaures or stochastic gradient descent methods [6, 7, 8], and parallelization of these methods are recently developed [9, 10, 11, 12]. However these approaches are prone to overfitting on the training data, hence require careful tuning of meta parameter such as regularization parameter, learning rate, and the number of iteration.

Bayesian treatment of matrix factorization successfully alleviates the overfitting problem by integrating out all model parameter, thus allowing for complex models to be learned without requiring much parameter tuning [2, 13, 3, 14, 15, 16]. In addition, Bayesian matrix factorization (BMF) easily incorporates side information such as user binary implicit feedback and item content information by placing Gaussian priors on user and item factor matrices, where mean of each prior is regressed on corresponding side information [17, 18].

The time and space complexity of VBMF depends on the mean-field assumption for variational distributions. When variational distributions are assumed to be matrix-wise independent, VBMF requires cubic time and quadratic space complexity with respect to K, existing variational inference algorithms for BMF are not well suited to web-scale datasets where billions of ratings provided by millions of users are available. When side information is incorporated into VBMF, it requires additional cubic time and quadratic space complexity with respect to the size of feature vectors formed by side information [18]. Thus, the use of rich side information in the form of high dimensional feature vector is prohibited. Assuming that variational distributions fully factorize to satisfy element-wise independence, a naive implementation of coordinate ascent algorithm requires quadratic time complexity with respect to K to update variational posterior means in element-wise fashion.

In this paper, we also assume that variational distributions fully factorize, but present a scalable algorithm for VBMF with side information, the time complexity of which is *linear* in K to update variational posterior means in element-wise fashion. Regression on side information also requires only linear time complexity with respect to the number of nonzero elements in feature vectors provided by side information. In addition, the proposed algorithm is easily parallelized on multi-core systems. Experiments on large-scale datasets (MovieLens10M, Douban, Netflix, and Yahoo! Music) with various side information (binary implicit feedback, neighborhood, social network, movie contents, music taxonomy) demonstrate the useful behavior of our algorithm such as scalability, fast learning, and prediction accuracy.

2 METHOD

In this section, we present our main contribution, a scalable algorithm for VBMF with side information, whose graphical representation is shown in Fig. 1.

2.1 Model

VBMF with side information assumes that the observed data X_{ij} is generated by

$$X_{ij} = \boldsymbol{u}_i^\top \boldsymbol{v}_j + e_{ij}, \tag{3}$$

for $(i,j) \in \Omega$, where Ω is a set of indices of observe entries in $\boldsymbol{X}$. The uncertainty in the model is absorbed by the noise e_{ij} which is assumed to be Gaussian, i.e., $e_{ij} \sim \mathcal{N}(e_{ij} \,|\, 0, \tau^{-1})$ where τ is the precision (the inverse of variance). Thus, the likelihood is given by

$$p(\boldsymbol{X} \,|\, \boldsymbol{U}, \boldsymbol{V}, \tau) = \prod_{(i,j)\in\Omega} \mathcal{N}(X_{ij} \,|\, \boldsymbol{u}_i^\top \boldsymbol{v}_j, \tau^{-1}). \tag{4}$$

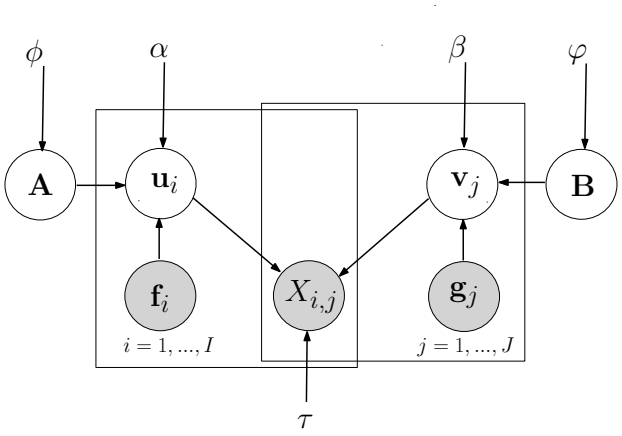


Figure 1: A graphical model for VBMF with side information.

Suppose that side information is given in the form of feature matrices $\boldsymbol{F} = [\boldsymbol{f}_1, ..., \boldsymbol{f}_I] \in \mathbb{R}^{M\times I}$ and $\boldsymbol{G} = [\boldsymbol{g}_1, ..., \boldsymbol{g}_J] \in \mathbb{R}^{N\times J}$, where $\boldsymbol{f}_i$ and $\boldsymbol{g}_j$ are feature vectors related with user i and item j respectively. VBMF with side information assumes that factor matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are generated from the side information via the linear regression,

$$\boldsymbol{U} = \boldsymbol{A}^\top \boldsymbol{F} + \boldsymbol{E}^u, \tag{5}$$
$$\boldsymbol{V} = \boldsymbol{B}^\top \boldsymbol{G} + \boldsymbol{E}^v, \tag{6}$$

where $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_K] \in \mathbb{R}^{M\times K}$ and $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_K] \in \mathbb{R}^{N\times K}$ are regression coefficient matrices. More precisely, Gaussian priors are placed on user and item factor matrices, where mean of each prior is regressed on corresponding side information:

$$p(\boldsymbol{U}|\boldsymbol{A}, \boldsymbol{F}, \boldsymbol{\alpha}) = \prod_{k=1}^{K}\prod_{i=1}^{I} \mathcal{N}(u_{ki} \,|\, \boldsymbol{a}_k^\top \boldsymbol{f}_i, \alpha_k^{-1}), \tag{7}$$
$$p(\boldsymbol{V}|\boldsymbol{B}, \boldsymbol{G}, \boldsymbol{\beta}) = \prod_{k=1}^{K}\prod_{j=1}^{J} \mathcal{N}(v_{kj} \,|\, \boldsymbol{b}_k^\top \boldsymbol{g}_j, \beta_k^{-1}). \tag{8}$$

Finally Gaussian priors are placed on regression coefficient matrices $\boldsymbol{A}$ and $\boldsymbol{B}$:

$$p(\boldsymbol{A}|\boldsymbol{\phi}) = \prod_{m=1}^{M}\prod_{k=1}^{K} \mathcal{N}(a_{mk} \,|\, 0, \phi_k^{-1}), \tag{9}$$
$$p(\boldsymbol{B}|\boldsymbol{\varphi}) = \prod_{n=1}^{N}\prod_{k=1}^{K} \mathcal{N}(b_{nk} \,|\, 0, \varphi_k^{-1}), \tag{10}$$

with hyperparameters $\boldsymbol{\phi}, \boldsymbol{\varphi}$.

2.2 Variational Inference

We approximately computes posterior distributions over factor matrices $\boldsymbol{U}, \boldsymbol{V}$ and regression coefficient matrices

Table 1: Variational distributions and corresponding updating rules are summarized when matrix-wise independence (13) is assumed. In addition, the space and time complexity for updating rules are also summarized. Ω_i is a set of indices j for which X_{ij} is observed, $\mathrm{diag}(\boldsymbol{\alpha})$ is a diagonal matrix with diagonal entries $\{\alpha_k\}$, $\boldsymbol{I}_M$ is the identity matrix with dimension M, and $U_k \in \mathbb{R}^{1\times I}$ is the k^{th} row vector of $\boldsymbol{U}$.

Variational distributions	Updating rules	Space complexity	Time complexity
$q(\boldsymbol{U})$ $= \prod_{i=1}^{I} \mathcal{N}(\boldsymbol{u}_i \vert \overline{\boldsymbol{u}}_i, \boldsymbol{\Lambda}_i^u)$	$\boldsymbol{\Lambda}_i^u = \left(\mathrm{diag}(\boldsymbol{\alpha}) + \tau \sum_{j\in\Omega_i} (\overline{\boldsymbol{v}}_j \overline{\boldsymbol{v}_j}^\top + \boldsymbol{\Lambda}_j^v)\right)^{-1}$, $\overline{\boldsymbol{u}}_i = \boldsymbol{\Lambda}_i^u \left(\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{A}^\top \boldsymbol{f}_i + \tau \sum_{j\in\Omega_i} X_{ij}\overline{\boldsymbol{v}}_j\right)$	$I(K+K^2)$	$\vert\Omega\vert K^2 + IK^3$
$q(\boldsymbol{V})$ $= \prod_{j=J}^{I} \mathcal{N}(\boldsymbol{v}_j \vert \overline{\boldsymbol{v}}_j, \boldsymbol{\Lambda}_j^v)$	$\boldsymbol{\Lambda}_j^v = \left(\mathrm{diag}(\boldsymbol{\beta}) + \tau \sum_{i\in\Omega_j} (\overline{\boldsymbol{u}}_i \overline{\boldsymbol{u}_i}^\top + \boldsymbol{\Lambda}_i^u)\right)^{-1}$, $\overline{\boldsymbol{v}}_j = \boldsymbol{\Lambda}_j^v \left(\mathrm{diag}(\boldsymbol{\beta})\boldsymbol{B}^\top \boldsymbol{g}_i + \tau \sum_{i\in\Omega_j} X_{ij}\overline{\boldsymbol{u}}_i\right)$	$J(K+K^2)$	$\vert\Omega\vert K^2 + JK^3$
$q(\boldsymbol{A})$ $= \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{a}_k \vert \overline{\boldsymbol{a}}_k, \boldsymbol{\Lambda}_k^a)$	$\boldsymbol{\Lambda}_k^a = \left(\phi_k \boldsymbol{I}_M + \alpha_k \boldsymbol{F}\boldsymbol{F}^\top\right)^{-1}$, $\overline{\boldsymbol{a}}_k = \boldsymbol{\Lambda}_k^a \left(\alpha_k \boldsymbol{F}\overline{U}_k^\top\right)$	$K(M+M^2)$	$K(IM+M^3)$
$q(\boldsymbol{B})$ $= \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{b}_k \vert \overline{\boldsymbol{b}}_k, \boldsymbol{\Lambda}_k^a)$	$\boldsymbol{\Lambda}_k^b = \left(\varphi_k \boldsymbol{I}_N + \beta_k \boldsymbol{G}\boldsymbol{G}^\top\right)^{-1}$, $\overline{\boldsymbol{b}}_k = \boldsymbol{\Lambda}_k^b \left(\beta_k \boldsymbol{G}\overline{V}_k^\top\right)$	$K(N+N^2)$	$K(JN+N^3)$

$\boldsymbol{A}, \boldsymbol{B}$ by maximizing a lower-bound on the marginal log-likelihood. Let $\mathcal{Z}$ be a set of all latent variables:

$$\mathcal{Z} = \{\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{A}, \boldsymbol{B}\}. \tag{11}$$

The marginal log-likelihood $\log p(\boldsymbol{X})$ is given by

$$\begin{aligned} \log p(\boldsymbol{X}) &= \log \int p(\boldsymbol{X}, \mathcal{Z}) d\mathcal{Z} \\ &\geq \int q(\mathcal{Z}) \log \frac{p(\boldsymbol{X}, \mathcal{Z})}{q(\mathcal{Z})} d\mathcal{Z} \\ &\equiv \mathcal{F}(q), \end{aligned} \tag{12}$$

where Jensen's inequality was used to obtain the *variational lower-bound* $\mathcal{F}(q)$ and $q(\mathcal{Z})$ is *variational distribution*.

In our previous work on VBMF with side information [18], variational distributions are assumed to be matrix-wise independent as in VBMF [2]

$$q(\mathcal{Z}) = q(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{A}, \boldsymbol{B}) = q(\boldsymbol{U})q(\boldsymbol{V})q(\boldsymbol{A})q(\boldsymbol{B}). \tag{13}$$

The free-form optimization results in products of multivariate Gaussian distributions as summarized in Table 1. Numerical experiments with MovieLens datasets showed that prediction accuracy can be significantly improved by incorporating user demographic information and movie genre information.

In practice, however, due to the cubic time and quadratic space complexity with respect to K, the variational inference algorithms under matrix-wise independence assumption are not well suited to web-scale datasets billions of ratings provided by millions of users are available. To make matter worse, the use of rich side information in the form of high dimensional feature vector (e.g. $M, N \gg 10000$) is prohibited because of cubic time complexity with respect to M and N.

2.3 Scalable Algorithm

Instead of the matrix-wisely factorized variational distributions (13), we consider fully factorized variational distributions which satisfies an element-wise independence,

$$\begin{aligned} q(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{A}, \boldsymbol{B}) = \\ \prod_{k,i} q(u_{ki}) \prod_{k,j} q(v_{kj}) \prod_{m,k} q(a_{mk}) \prod_{n,k} q(b_{nk}). \end{aligned} \tag{14}$$

The free-form optimization results in products of univariate Gaussian distributions:

$$\begin{aligned} q(\boldsymbol{U}) &= \prod_{k=1}^{K} \prod_{i=1}^{I} \mathcal{N}(u_{ki} \vert \overline{u}_{ki}, s_{ki}^u), \\ q(\boldsymbol{V}) &= \prod_{k=1}^{K} \prod_{j=1}^{J} \mathcal{N}(v_{kj} \vert \overline{v}_{kj}, s_{ki}^v), \\ q(\boldsymbol{A}) &= \prod_{m=1}^{M} \prod_{k=1}^{K} \mathcal{N}(a_{mk} \vert \overline{a}_{mk}, s_{mk}^a), \\ q(\boldsymbol{B}) &= \prod_{n=1}^{N} \prod_{k=1}^{K} \mathcal{N}(b_{nk} \vert \overline{b}_{nk}, s_{nk}^b). \end{aligned} \tag{15}$$

We compute the variational lower-bound $\mathcal{F}(q)$:

$$\begin{aligned} \mathcal{F}(q) = \mathbb{E}_q \{ &\log p(\boldsymbol{X} \vert \boldsymbol{U}, \boldsymbol{V}) \\ &+ \log p(\boldsymbol{U} \vert \boldsymbol{A}, \boldsymbol{F}) + \log p(\boldsymbol{V} \vert \boldsymbol{B}, \boldsymbol{G}) \\ &+ \log p(\boldsymbol{A}) + \log p(\boldsymbol{B}) - \log q(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{A}, \boldsymbol{V}) \}, \end{aligned}$$

where $\mathbb{E}_q\{\cdot\}$ denotes the statistical expectation with respect to $q(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{A}, \boldsymbol{B})$ that is of the form (15). Invoking the model (4) with prior distributions (7), (8), (9) and (10) as well as the fully factorized variational distribution (15), we

calculate $\mathcal{F}(q)$ as

$$\mathcal{F}(q) = \sum_{(i,j)\in\Omega} \mathcal{F}_{ij} + \sum_{k=1}^{K}\sum_{i=1}^{I} \mathcal{F}^u_{ki} + \sum_{k=1}^{K}\sum_{j=1}^{J} \mathcal{F}^v_{kj} + \sum_{m=1}^{M}\sum_{k=1}^{K} \mathcal{F}^a_{mk} + \sum_{n=1}^{N}\sum_{k=1}^{K} \mathcal{F}^b_{nk}, \quad (16)$$

where

$$\begin{aligned}
\mathcal{F}_{ij} &= \mathbb{E}_q\{\log p(X_{ij}|\boldsymbol{U},\boldsymbol{V})\} \\
&= -\frac{\tau}{2}(R_{ij}^2 + W_{ij}) - \frac{1}{2}\log(2\pi\tau^{-1}), \\
R_{ij} &= X_{ij} - \sum_{k=1}^{K} \overline{u}_{ki}\overline{v}_{kj}, \\
W_{ij} &= \sum_{k=1}^{K} (\overline{u}_{ki}^2 s^v_{kj} + \overline{v}_{kj}^2 s^u_{ki} + s^u_{ki}s^v_{kj}), \\
\mathcal{F}^u_{ki} &= \mathbb{E}_q\{\log p(u_{ki}|\boldsymbol{a}_k,\boldsymbol{f}_i) - \log q(u_{ki})\} \\
&= -\frac{\alpha_k}{2}\left[(\overline{u}_{ki} - \overline{\boldsymbol{a}}_k^\top \boldsymbol{f}_i)^2 + s^u_{ki} + \sum_{m=1}^{M} s^a_{mk} f^2_{mi}\right] \\
&\quad + \frac{1}{2}\log(s^u_{ki}\alpha_k) + \frac{1}{2}, \\
\mathcal{F}^v_{kj} &= \mathbb{E}_q\left\{\log p(v_{kj}|\overline{\boldsymbol{b}}_k,\boldsymbol{g}_j) - \log q(v_{kj})\right\} \\
&= -\frac{\beta_k}{2}\left[(\overline{v}_{kj} - \overline{\boldsymbol{b}}_k^\top \boldsymbol{g}_j)^2 + s^v_{kj} + \sum_{n=1}^{N} s^b_{nk} g^2_{nj}\right] \\
&\quad + \frac{1}{2}\log(s^v_{kj}\beta_k) + \frac{1}{2}, \\
\mathcal{F}^a_{mk} &= \mathbb{E}_q\{\log p(a_{mk}) - \log q(a_{mk})\} \\
&= -\frac{\phi_k}{2}(\overline{a}_{mk}^2 + s^a_{mk}) + \frac{1}{2}\log(s^a_{mk}\phi_k) + \frac{1}{2}, \\
\mathcal{F}^b_{nk} &= \mathbb{E}_q\{\log p(b_{nk}) - \log q(b_{nk})\} \\
&= -\frac{\varphi_k}{2}(\overline{b}_{nk}^2 + s^b_{nk}) + \frac{1}{2}\log(s^b_{nk}\varphi_k) + \frac{1}{2}.
\end{aligned}$$

The updating rules for variational parameters can be obtained by setting derivative the variational lower-bound (16) to zero. For example, in the case of $q(u_{ki})$, the updating rules for variance and mean are given by

$$\begin{aligned}
s^u_{ki} &= (\alpha_k + \tau\sum_{j\in\Omega_i}(\overline{v}_{kj}^2 + s^v_{kj}))^{-1}, \quad (17) \\
\overline{u}_{ki} &= s^u_{ki}(\alpha_k \overline{\boldsymbol{a}}_k^\top \boldsymbol{f}_i + \tau\sum_{j\in\Omega_i}(X_{ij} - \sum_{k'\neq k}\overline{u}_{k'i}\overline{v}_{k'j})\overline{v}_{kj}),
\end{aligned}$$

where Ω_i is a set of indices j for which X_{ij} is observed. Since updating of $\overline{u}_{ki}$ requires $O(|\Omega_i|K)$, and similarly updating $\overline{v}_{kj}$ requires $O(|\Omega_j|K)$, the time complexity for updating all variational posterior means of factor matrices is quadratic with respect to K:

$$O\left(\sum_{k=1}^{K}\sum_{i=1}^{I}|\Omega_i|K + \sum_{k=1}^{K}\sum_{j=1}^{J}|\Omega_j|K\right) = O\left(2|\Omega|K^2\right).$$

Although this naive implementation requires quadratic time complexity with respect to K, we can efficiently reduce the time complexity to linear with a simple trick. Let $R_{ij} = X_{ij} - \sum_{k=1}^{K}\overline{u}_{ki}\overline{v}_{kj}$ denote the residual on (i,j) observation. With the residual R_{ij}, the updating rule (17) can be rewritten as

$$\overline{u}_{ki} = s^u_{ki}\left(\alpha_k \overline{\boldsymbol{a}}_k^\top \boldsymbol{f}_i + \tau\sum_{j\in\Omega_i}(R_{ij} + \overline{u}_{ki}\overline{v}_{kj})\overline{v}_{kj}\right)$$

and it requires constant time complexity $O(|\Omega_i|)$ with respect to K. When $\overline{u}_{ki}$ is changed to $\overline{u}'_{ki}$, R_{ij} can be easily updated to R'_{ij} for all $j \in \Omega_i$ by:

$$R'_{ij} = R_{ij} - (\overline{u}'_{ki} - \overline{u}_{ki})\overline{v}_{kj}. \quad (18)$$

Now time complexity per iteration (including updating of variational posterior variances and residuals) is reduced to linear with respect to K:

$$O\left(\sum_{k=1}^{K}\sum_{i=1}^{I} 3|\Omega_i| + \sum_{k=1}^{K}\sum_{j=1}^{J} 3|\Omega_j|\right) = O\left(6|\Omega|K\right). \quad (19)$$

Using the similar idea, variational posterior means of regression coefficient matrices can be efficiently updated. Let $\boldsymbol{R}^u = \overline{\boldsymbol{U}} - \overline{\boldsymbol{A}}^\top \boldsymbol{F}$ denote the residual on the variational posterior mean $\overline{\boldsymbol{U}}$. With the residual $\boldsymbol{R}^u$, the updating rules for s^a_{mk} and $\overline{a}_{mk}$ can be written as

$$\begin{aligned}
s^a_{mk} &= \left(\phi_k + \alpha_k\|F_m\|^2\right)^{-1}, \quad (20) \\
\overline{a}_{mk} &= s^a_{mk}\alpha_k(R^u_k + \overline{a}_{mk}F_m)F_m^\top, \quad (21)
\end{aligned}$$

where $R^u_k \in \mathbb{R}^{1\times I}$ is the k^{th} row vector of $\boldsymbol{R}^u$ and $F_m \in \mathbb{R}^{1\times I}$ is the m^{th} row vector of $\boldsymbol{F}$. When $\overline{a}_{mk}$ is updated to $\overline{a}'_{mk}$, R^u_k is updated to $R^{u'}_k$:

$$R^{u'}_k = R^u_k - (\overline{a}'_{mk} - \overline{a}_{mk})F_m. \quad (22)$$

Note that updating s^a_{mk} and $\overline{a}_{mk}$ takes $O(3\|F_m\|_0)$, where $\|\cdot\|$ denotes the L_0 norm which equals to number of non-zero elements. Similarly updating s^b_{nk} and $\overline{b}_{nk}$ takes $O(3\|G_n\|_0)$, hence overall time complexity for updating variational posterior variances and means $\{\boldsymbol{S}^a, \overline{\boldsymbol{A}}, \boldsymbol{S}^b, \overline{\boldsymbol{B}}\}$ of regression coefficient matrices is linear in the number of nonzero elements in feature matrices:

$$O\left(\sum_{m=1}^{M}\sum_{k=1}^{K} 3\|F_m\|_0 + \sum_{n=1}^{N}\sum_{k=1}^{K} 3\|G_n\|_0\right) = O\left(3(\|\boldsymbol{F}\|_0 + \|\boldsymbol{G}\|_0)K\right).$$

Algorithm 1 VBMFSI-CA: linear complexity learning algorithm for variational Bayesian matrix factorization with side information. VBMFSI-CA can be easily parallelized in column-by-column manner.

Initialize $\overline{\boldsymbol{U}},\ \boldsymbol{S}^u,\ \overline{\boldsymbol{V}},\ \boldsymbol{S}^v,\ \overline{\boldsymbol{A}},\ \boldsymbol{S}^a,\ \overline{\boldsymbol{B}},\ \boldsymbol{S}^b$.
$\boldsymbol{R} = \boldsymbol{X} - \overline{\boldsymbol{U}}^\top \overline{\boldsymbol{V}}$,
$\boldsymbol{R}^u = \overline{\boldsymbol{U}} - \overline{\boldsymbol{A}}^\top \boldsymbol{F}, \quad \boldsymbol{R}^v = \overline{\boldsymbol{V}} - \overline{\boldsymbol{B}}^\top \boldsymbol{G}$
for $t = 1 \dots T$ **do**
  /********** Update $q(\boldsymbol{U})$ **********/
  parallel for $i = 1, \dots, I$ **do**
    for $k = 1, \dots, K$ **do**
      $\xi \leftarrow \overline{u}_{ki}$
      $s^u_{ki} \leftarrow \left(\alpha_k + \tau \sum_{j \in \Omega_i} (\overline{v}^2_{kj} + s^v_{kj})\right)^{-1}$
      $\theta \leftarrow \left(\alpha_k \overline{\boldsymbol{a}}_k^\top \boldsymbol{f}_i + \tau \sum_{j \in \Omega_i} (R_{ij} + \overline{u}_{ki}\overline{v}_{kj})\overline{v}_{kj}\right)$
      $\overline{u}_{ki} \leftarrow s^u_{ki}\, \theta$
      for $j \in \Omega_i$ **do**
        $R_{ij} \leftarrow R_{ij} - (\overline{u}_{ki} - \xi)\overline{v}_{kj}$
      end for
    end for
  end parallel for
  /********** Update $q(\boldsymbol{V})$ **********/
  parallel for $j = 1, \dots, J$ **do**
    for $k = 1, \dots, K$ **do**
      $\xi \leftarrow \overline{v}_{kj}$
      $s^v_{kj} \leftarrow \left(\beta_k + \tau \sum_{i \in \Omega_j} (\overline{u}^2_{ki} + s^u_{ki})\right)^{-1}$
      $\theta \leftarrow \left(\beta_k \overline{\boldsymbol{b}}_k^\top \boldsymbol{g}_j + \tau \sum_{i \in \Omega_j} (R_{ij} + \overline{u}_{ki}\overline{v}_{kj})\overline{u}_{ki}\right)$
      $\overline{v}_{kj} \leftarrow s^v_{kj}\, \theta$
      for $i \in \Omega_j$ **do**
        $R_{ij} \leftarrow R_{ij} - \overline{u}_{ki}(\overline{v}_{kj} - \xi)$
      end for
    end for
  end parallel for
  /********** Update $q(\boldsymbol{A})$ **********/
  parallel for $k = 1, \dots, K$ **do**
    for $m = 1, \dots, M$ **do**
      $\xi \leftarrow \overline{a}_{mk}$
      $s^a_{mk} \leftarrow \left(\phi_k + \alpha_k \|F_m\|^2\right)^{-1}$
      $\overline{a}_{mk} \leftarrow s^a_{mk}\alpha_k(R^u_k + \overline{a}_{mk}F_m)F_m^\top$
      $R^u_k \leftarrow R^u_k - (\overline{a}_{mk} - \xi)F_m$
    end for
  end parallel for
  /********** Update $q(\boldsymbol{B})$ **********/
  parallel for $k = 1, \dots, K$ **do**
    for $n = 1, \dots, N$ **do**
      $\xi \leftarrow \overline{b}_{nk}$
      $s^b_{nk} \leftarrow \left(\varphi_k + \beta_k \|G_n\|^2\right)^{-1}$
      $\overline{b}_{nk} \leftarrow s^b_{nk}\beta_k(R^v_k + \overline{b}_{nk}G_n)G_n^\top$
      $R^v_k \leftarrow R^v_k - (\overline{b}_{nk} - \xi)G_n$
    end for
  end parallel for
  /********** Update hyperparameters **********/
  $\tau \leftarrow |\Omega| / \sum_{(i,j)\in\Omega}(R^2_{ij} + W_{ij})$
  for $k = 1, \dots, K$ **do**
    $$\alpha_k \leftarrow \frac{I}{\sum_{i=1}^{I}\left[(\overline{u}_{ki} - \overline{\boldsymbol{a}}_k^\top \boldsymbol{f}_i)^2 + s^u_{ki} + \sum_{m=1}^{M} s^a_{mk} f^2_{mi}\right]}$$
    $$\beta_k \leftarrow \frac{J}{\sum_{j=1}^{J}\left[(\overline{v}_{kj} - \overline{\boldsymbol{b}}_k^\top \boldsymbol{g}_j)^2 + s^v_{kj} + \sum_{n=1}^{N} s^b_{nk} g^2_{nj}\right]}$$
    $\phi_k \leftarrow M / \left(\sum_{m=1}^{M} \overline{a}^2_{mk} + s^a_{mk}\right)$
    $\varphi_k \leftarrow N / \left(\sum_{n=1}^{N} \overline{b}^2_{nk} + s^b_{nk}\right)$
  end for
end for

Our linear time complexity learning algorithm for VBMF with side information, which is referred in to as VBMFSI-CA in this paper, is summarized in Algorithm 1. The space complexity of VBMFSI-CA is also linear with respect to K because we only have to save the variational posterior means and variances instead of covariances. VBMFSI-CA is a coordinate ascent method, where a single variational posterior parameter or hyperparameter is updated at a time while keeping others fixed. The convergence of coordinate ascent method is guaranteed when the the solution for each coordinate is uniquely attained [19] and VBMFSI-CA fulfills this condition. Note that parallelization can be easily done in a column-by-column manner because each column of variational parameters $\{\boldsymbol{S}^u, \overline{\boldsymbol{U}}, \boldsymbol{S}^v, \overline{\boldsymbol{V}}, \boldsymbol{S}^b, \overline{\boldsymbol{A}}, \boldsymbol{S}^b, \overline{\boldsymbol{B}}\}$ can be updated independently from the updates of other columns.

Our approach bases on the element-wise independence assumption for variational distributions which is much stricter than matrix-wise independence assumption, hence there will be an loss in prediction accuracy. However our experimental results confirmed that the element-wise independence assumption does not sacrifice the prediction accuracy. We empirically observed that variational posterior covariances in (13) are almost diagonal, so it looks like that information losses due to the element-wise independent assumption is not critical, although we cannot theatrically guarantee it. In fully observed case, there is an interesting theoretical results [20] but it cannot be applied in the presence of missing values.

3 RELATED WORK

3.1 Matrix Factorization with Side Information

We briefly review several matrix factorization models which incorporates side information. *Matrix co-factorization* is one of an efficient ways to exploit the side information. Matrix co-factorization jointly decomposes

multiple data matrices, where each decomposition is coupled by sharing some factor matrices [21, 22]. For example, rating matrix $\boldsymbol{X}$, user side information matrix $\boldsymbol{F}$ and item side information matrix $\boldsymbol{G}$ are jointly decomposed as

$$\boldsymbol{F} = \boldsymbol{A}^\top \boldsymbol{U} + \boldsymbol{E}^f,\ \boldsymbol{X} = \boldsymbol{U}^\top \boldsymbol{V} + \boldsymbol{E}^x,\ \boldsymbol{G} = \boldsymbol{B}^\top \boldsymbol{V} + \boldsymbol{E}^g,$$

where factor matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are shared in the first two and last two decompositions respectively, and $\boldsymbol{E}^f, \boldsymbol{E}^x, \boldsymbol{E}^g$ represents Gaussian noise which reflect uncertainties. Recently Bayesian treatment of matrix co-factorization was developed, where the inference was performed using a sampling method [23] and the variational method [24, 25].

There are two different approaches to incorporate the side information with regression in Bayesian matrix factorization framework [26, 17]. In the first approach [26], the regression coefficient and side information are augmented in factor matrices such that the rating matrix $\boldsymbol{X}$ is estimated by the sum of the collaborative prediction part and regression part against side information:

$$\begin{aligned} \boldsymbol{X} &= \left[\boldsymbol{U}^\top \boldsymbol{A}^\top \boldsymbol{F}^\top\right] \left[\boldsymbol{V}^\top \boldsymbol{G}^\top \boldsymbol{B}^\top\right]^\top + \boldsymbol{E}^x \\ &= \boldsymbol{U}^\top \boldsymbol{V} + \boldsymbol{A}^\top \boldsymbol{G} + \boldsymbol{F}^\top \boldsymbol{B} + \boldsymbol{E}^x. \end{aligned}$$

On the other hand, *Regression-based Latent Factor Model* (RLFM) [17] assumes that the user and item factor matrices are generated from the side information via the linear regression,

$$\boldsymbol{U} = \boldsymbol{A}^\top \boldsymbol{F} + \boldsymbol{E}^u, \quad \boldsymbol{V} = \boldsymbol{B}^\top \boldsymbol{G} + \boldsymbol{E}^v, \tag{23}$$

and then rating matrix $\boldsymbol{X}$ is generated by (1). Recently RLMF is generalized to hierarchical Bayesian matrix factorization with side information [18].

It was hard to directly compare the effectiveness of previous three different modelings for side information, because each model was trained by different inference algorithm such as Markov Chain Monte Carlo [26], Monte Carlo Expectation Maximization [17], and variational method [24, 25, 18]. The experimental results in [18], where all discussed models are trained by variational method for fair comparisons, showed that the third approach is the most promising way to incorporate side information in both cold and warm start situation. Note that our work also belongs to this third approach. All discussed methods assumed that the side information is given in form of the predefined feature vector. Recently a general functional matrix factorization which automatically constructs the feature function has been proposed [27].

3.2 Optimization

In the perspective of optimization techniques, our work is closely related with [8], where similar idea is used to reduce the time complexity of ALS algorithm. The trick that reuses the residuals on the observation and efficient updating of them by (18), is a key point of reducing the time complexity from quadratic to linear and it was presented in [8]. Note that computational complexity for optimizing the objective function of weighed trace norm regularized matrix factorization (2) and the variational lower-bound from the fully factorized variational distribution (16) via coordinate ascent method is similar:

$$O(4|\Omega|K) \ \text{ vs. } \ O(6|\Omega|K).$$

However the benefit from the extra small effort is crucial: more accurate model is obtainable without tuning of regularization parameter.

Similar to our VBMFSI-CA, the learning algorithm for VBMF in [13], which is referred to as VBMF-GA in this paper, also considers the fully factorized variational distribution. VBMF-GA updates variational posterior variances $\{\boldsymbol{S}^u, \boldsymbol{S}^v\}$ and hyperparameters $\{\tau, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$ by fixed-point rules as in VBMFSI-CA, however updates variational posterior means $\{\overline{\boldsymbol{U}}, \overline{\boldsymbol{V}}\}$ by scaled gradient ascent method,

$$\begin{aligned} \overline{u}_{ki} &\leftarrow \overline{u}_{ki} + \eta \left(\frac{\partial^2 \mathcal{F}}{\partial^2 \overline{u}_{ki}}\right)^{-\gamma} \frac{\partial \mathcal{C}}{\partial \overline{u}_{ki}}, \\ \overline{v}_{kj} &\leftarrow \overline{v}_{ki} + \eta \left(\frac{\partial^2 \mathcal{F}}{\partial^2 \overline{v}_{kj}}\right)^{-\gamma} \frac{\partial \mathcal{C}}{\partial \overline{v}_{kj}}, \end{aligned} \tag{24}$$

where $\mathcal{F}$ is the variational lower-bound, η is a learning rate, and diagonal part of the Hessian matrix is used for scaling with control parameter γ. The parameter γ allows the learning algorithm to vary from standard gradient ascent ($\gamma = 0$) to diagonal Newton's method ($\gamma = 1$). VBMF-GA also takes linear space and time complexity but it requires tuning of hyperparameters η and γ. In addition, empirical results showed that learning speed is slower than our VBMFSI-CA.

4 NUMERICAL EXPERIMENTS

4.1 Datasets and Side Information

We performed experiments on four large-scale datasets (MovieLens-10M[1], Douban[2], Netflix[3], and Yahoo! Music[4]) with various side information (binary implicit feedback, neighborhood, social network, movie contents, and music taxonomy). Table 2 presents the statistics of these four datasets.

Binary implicit feedback: One of the lesson learnt from the Netflix Prize competition is an importance of integrating implicit feedback into models [28, 29]. For a dataset

[1] http://www.grouplens.org/datasets/movielens
[2] https://www.cse.cuhk.edu.hk/irwin.king/pub/data/douban
[3] http://www.netflixprize.com
[4] http://kddcup.yahoo.com

Table 2: Statistics of datasets.

Dataset	$\|\Omega\|$	I	J	$\frac{\|\Omega\|}{IJ}(\%)$
MovieLens	10M	69,878	10,677	1.34
Douban	17M	129,490	58,541	0.22
Netflix	100M	480,189	17,770	1.18
Yahoo!	263M	1,000,990	624,961	0.04

such as the Netflix, the most natural choice for implicit feedback would be movie rental history, which tells us about user preferences without requiring them to explicitly provide their ratings. For other datasets, browsing or purchase history could be used as implicit feedback. Although such data is not available to us for experiments, we alternatively used the binary indicator matrix which indicates whether the user rates the movie as implicit feedback. In our experiments, we used the normalized binary implicit feedback as the user side information $\boldsymbol{F} \in \mathbb{R}^{M \times I}$ defined as

$$f_{mi} = \begin{cases} 1/\sqrt{|\Omega_i|} & \text{if } (i,m) \in \Omega, \\ 0 & \text{otherwise,} \end{cases} \tag{25}$$

where the dimension of feature M is equal to the number of item N, and each feature vector $\boldsymbol{f}_i$ is normalized to unit length. If we incorporate the normalized binary implicit feedback feature (25) with our user generative model (5), then user i is modeled as

$$\boldsymbol{u}_i \quad = \quad \boldsymbol{A}^\top \boldsymbol{f}_i + \boldsymbol{e}_i^u = \frac{\sum_{j \in \Omega_i} A_j^\top}{\sqrt{|\Omega_i|}} + \boldsymbol{e}_i^u,$$

where $A_j \in \mathbb{R}^{1 \times K}$ is the j^{th} row vector of $\boldsymbol{A}$. Note that this user modeling is equivalent to the one used in SVD++.

Neighborhood: The neighborhood-based approach is one of the primary method for recommendation system. In [29], the item-based neighborhood model was integrated with matrix factorization and showed good prediction accuracy. In our experiments, the item-based neighborhood information is incorporated by defining the item feature matrix $\boldsymbol{G} \in \mathbb{R}^{N \times J}$ as

$$g_{nj} = \begin{cases} c_{nj}/Z_j & \text{if } c_{nj} \geq \theta, \\ 0 & \text{otherwise,} \end{cases} \tag{26}$$

where dimension of feature N is equal to the number of item J, c_{nj} is the similarity between two items n and j, Z_j is the appropriate normalizer which makes unit length $\boldsymbol{g}_j$, θ is the threshold for determining the neighborhood, and Pearson correlation coefficient is used as the similarity criterion [30]. With neighborhood feature (26), item j is modeled as

$$\boldsymbol{v}_j \quad = \quad \boldsymbol{B}^\top \boldsymbol{g}_j + \boldsymbol{e}_j^v = \sum_{n \in \Upsilon_j} g_{nj} B_n^\top + \boldsymbol{e}_j^v,$$

where Υ_j is the set of indices n for which $g_{nj} > 0$. Note g_{nj} has the positive value only if item n is the neighborhood of item j.

Social network: Based on the intuition that users' social relations can be employed to enhance the recommender system, several social recommendation methods have been proposed [31, 32, 33, 34, 35]. In our experiments, the social network information is incorporated by defining the user feature matrix $\boldsymbol{F} \in \mathbb{R}^{M \times I}$ as

$$f_{mi} = \begin{cases} c_{mi}/Z_i & \text{if } (m,i) \in \Gamma \text{ and } c_{mi} \geq \theta, \\ 0 & \text{otherwise,} \end{cases} \tag{27}$$

where dimension of feature M is equal to the number of user I, $(m,i) \in \Gamma$ denotes that user m is the friend of user i, c_{mi} is the similarity between two users m and i, Z_i is the appropriate normalizer which makes $\boldsymbol{f}_i$ to unit length.

Movie content: MovieLens-10M dataset provides the genre information of the movies. We extend the content information by collecting the director, actor, and tag information from the IMDB. The extended content information consists of 19 genres, 1710 directors, 35779 actors, and 5294 tags. This movie content information is encoded in binary valued vector with dimension $N = 42802$ and then normalized to unit length.

Music taxonomy: A distinctive feature of the Yahoo! Music dataset is that user ratings are given to entities of four different types: *tracks*, *albums*, *artists*, and *genres*. The majority of items are tracks (81.15%), followed by albums (14.23%), artists (4.46%), and genres (0.16%) [36]. In addition, items are tied together within a taxonomy. That is, for a track we know the identity of its album, performing artist and associated genres. Similarity we have artist and genre annotation for the albums. There is no genre information for artists, because artists may switch between many genres in their career. This taxonomy information is encoded in normalized binary valued vector with dimension $N = 117789$, which equals to the sum of the number of albums, artists, and genres.

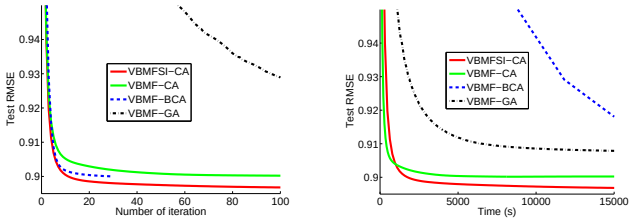
4.2 Learning Speed

We compare our learning algorithm for VBMF with side information, which is referred in to as VBMFSI-CA in this paper, with three learning algorithms for VBMF.

VBMF-CA: This is a reduced version of VBMFSI-CA, where side information is not incorporated.

VBMF-BCA: The variational lower bound from the matrix-wisely factorized variational distributions is optimized by block coordinate ascent method [2]. The time complexity of VBMF-BCA is cubic with respect to K.

VBMF-GA: The variational lower bound (16) is optimized by gradient ascent method [13]. The time complexity of VBMF-GA is linear with respect to K. We followed the same adaptation rule for learning rate η and setting of the scaling parameter $\gamma = 2/3$ presented in [13] for fair comparison.



(a) (b)

Figure 2: Comparison between VBMFSI-CA, VBMF-CA, VBMF-BCA, VBMF-GA on the Netflix dataset with $K = 50$: (a) Number of iteration vs. RMSE, (b) Time vs. RMSE.

We do not compared the existing learning algorithm for VBMF with side information proposed in [18], which uses similar updating rules described in Table 1, because it can't handle the side information in the form of high dimensional feature vector (e.g. $M, N \gg 10000$). We used a quad-core Intel(R) core(TM) i7-3820 @ 3.6GHz processor and 64GB memory for the comparison. All algorithms were implemented in Matlab2011a, where main computational modules are implemented in C++ as mex files and parallelized with the OpenMP library. [5]

For the Netflix dataset, we trained the VBMF model with four algorithms, where rank K was set to 50. In the case of VBMFSI-CA, the binary implicit feedback was used for side information. Fig. 2-(a) shows the comparison of the number of iteration versus test RMSE for the four learning algorithms. In the perspective of the number of iteration, VBMFSI-CA is the best, VBMF-GA is the worst and VBMF-BCA (RMSE=0.9000) is slightly better than VBMF-CA (RMSE=0.9002). However the time per iteration of each algorithm is quite different: 158 seconds, 70 seconds, 66 minutes, and 19 seconds for VBMFSI-CA, VMBF-CD, VBMF-BCA and VBMF-GA respectively. Hence we also compare the four learning algorithm in the perspective of training time as shown in Fig. 2-(b). After factoring the time per iteration, VBMF-GA converges faster than VBMF-BCA but it usually finds bad local solution. Our VBMFSI-CA and VBMF-CA clearly outperforms VBMF-GA and VBMF-BCA both in the learning speed and the prediction accuracy. Especially VBMFSI-CA finds the most accurate model by incorporating the binary implicit feedback information and its additional computational cost is acceptable because it is linear in the number of nonzero elements in feature matrix, that is almost equal to the number of observed ratings ($\|\boldsymbol{F}\|_0 \approx |\Omega|$).

4.3 Effect of Side Information

We trained VBMFSI-CA for each dataset, where $K = 50$ and the maximum number of training iteration was set to 100. We performed 5-fold cross-validation for Douban dataset and used pre-defined train/test split for others. As shown in Table 3, significant improvement is achieved when side information is included, where B, N, S, C, and T denote the binary implicit feedback, neighborhood, social network, movie contents, and music taxonomy respectively. Especially, the binary implicit feedback gave the most effective result.

[5] Source code: http://mlg.postech.ac.kr.com/∼karma13

Table 3: Test RMSE and time per iteration on different datasets with various kinds of side information.

MovieLens 10M, r_a			
	RMSE	% Improvement	Time (s)
none	0.8582	0.00%	5.3
C	0.8553	0.34%	5.5
B	0.8455	1.48%	8.2
B, C	0.8444	1.61%	8.4
Douban, 5-fold cross-validation			
	RMSE	% Improvement	Time (s)
none	0.6913	0.00%	17
S	0.6894	0.30%	18
B	0.6856	0.84%	24
Netflix, probe10			
	RMSE	% Improvement	Time (s)
none	0.9002	0.00%	70
N	0.9000	0.02%	80
B	0.8968	0.38%	158
B, N	0.8944	0.64%	167
Yahoo! Music, validation (not test)			
	RMSE	% Improvement	Time (s)
none	22.0313	0.00%	237
T	21.8808	0.68%	242
B	21.7045	1.48%	524
B, T	21.5673	1.87%	531

5 CONCLUSIONS

We have presented a scalable algorithm for VBMF with side information, the time complexity of which is linear in K to update variational parameters in element-wise fashion. Incorporating side information also requires only linear time complexity with respect to the number of nonzero elements in feature vectors provided by side information. In addition, the proposed algorithm is easily parallelized on multi-core systems. Experimental results on large-scale datasets with various side information confirmed the useful behavior of our algorithm such as scalability, fast learning, and prediction accuracy.

Acknowledgements

This work was supported by National Research Foundation (NRF) of Korea (NRF-2013R1A2A2A01067464) and POSTECH Rising Star Program.

References

[1] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, Omaha, NE, 2007.

[2] Y. J. Lim and Y. W. Teh. Variational Bayesian approach to movie rating prediction. In *Proceedings of KDD Cup and Workshop*, San Jose, CA, 2007.

[3] R. Salakhutdinov and A. Mnih. Bayesian probablistic matrix factorization using MCMC. In *Proceedings of the International Conference on Machine Learning (ICML)*, Helsinki, Finland, 2008.

[4] Y.-D. Kim and S. Choi. Weighted nonnegative matrix factorization. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009.

[5] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005.

[6] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research*, 10:623–656, 2009.

[7] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[8] I .Pilászy, D. Zibriczky, and D. Tikk. Fast ALS-based matrix factorization for explicit and implicit feedback datasets. In *Proceedings of the ACM International Conference on Recommender Systems (RecSys)*, Barcelona, Spain, 2010.

[9] H.-F. Yu, C.-J. Hsieh, S. Si, and I. Dhillon. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, Brussels, Belgium, 2012.

[10] R. Gemulla, P. J. Hass, E. Nijkamp, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, San Diego, CA, USA, 2011.

[11] Y. Zhuang, W. Chin, Y. Juan, and C. Lin. A fast parallel SGD for matrix factorization in shared memory systems. In *Proceedings of the ACM International Conference on Recommender Systems (RecSys)*, Hong Kong, China, 2013.

[12] S. Schelter, C. Boden, M. Schenck, A. Alexandrov, and V. Markl. Distributed matrix factorization with mapreduce using a series of broadcast-joins. In *Proceedings of the ACM International Conference on Recommender Systems (RecSys)*, Hong Kong, China, 2013.

[13] T. Raiko, A. Ilin, and J. Karhunen. Principal component analysis for large scale problems with lots of missing values. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 691–698, Warsaw, Poland, 2007.

[14] D. Stern, R. Herbrich, and R. Herbrich. Matchbox: Large scale online Bayesian recommendations. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 111–120, 2009.

[15] Y.-D. Kim and S. Choi. Variational Bayesian view of weighted trace norm regularization for matrix factorization. *IEEE Signal Processing Letters*, 20(3):261–264, 2013.

[16] N. Koenigstein and U. Paquet. Xbox movies recommendations: Variational Bayes matrix factorization with embedded feature selection. In *Proceedings of the ACM International Conference on Recommender Systems (RecSys)*, Hong Kong, China, 2013.

[17] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, Paris, France, 2009.

[18] S. Park, Y.-D. Kim, and S. Choi. Hierarchical Bayesian matrix factorization with side information. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Beijing, China, 2013.

[19] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[20] S. Nakajima, M. Sugiyama, and D. Babacan. Global solution of fully-observed variational bayesian matrix factorization is column-wise independent. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24, 2011.

[21] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, Las Vegas, Nevada, 2008.

[22] G. Bouchard, S. Guo, and D. Yin. Convex collective matrix factorization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Scottsdale, AZ, USA, 2013.

[23] A. P. Singh and G. J. Gordon. A Bayesian matrix factorization model for relational data. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, CA, 2010.

[24] J. Yoo and S. Choi. Bayesian matrix co-factorization: Variational algorithm and Cramér-Rao bound. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, Athens, Greece, 2011.

[25] J. Yoo and S. Choi. Hierarchical variational Bayesian matrix co-factorization. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Kyoto, Japan, 2012.

[26] I. Proteous, A. Asuncion, and M. Welling. Bayesian matrix factorization with side information and Dirichlet process mixtures. In *Proceedings of the AAAI National Conference on Artificial Intelligence (AAAI)*, Atlanta, Georgia, USA, 2010.

[27] T. Chen, H. Li, Q. Yang, and Y. Yu. General functional matrix factorization using gradient boosting. In *Proceedings of the International Conference on Machine Learning (ICML)*, Atlanta, Georgia, USA, 2013.

[28] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the International Conference on Machine Learning (ICML)*, Corvallis, OR, USA, 2007.

[29] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, Las Vegas, Nevada, USA, 2008.

[30] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, Madison, WI, 1998.

[31] H. Ma, H. Yang, M. R. Lyu, and I. King. SoRec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, Napa Valley, CA, 2008.

[32] H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, Boston, MA, 2009.

[33] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, Hong Kong, 2010.

[34] M. Jamali and M. Ester. TrustWalker: A random walk model for combining trust-based and item-based recommendation. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, Paris, France, 2009.

[35] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the ACM International Conference on Recommender Systems (RecSys)*, Barcelona, Spain, 2010.

[36] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. The yahoo! music dataset and KDD-Cup'11. In *Proceedings of KDD Cup and Workshop*, 2011.