Bayesian Deep Collaborative Matrix Factorization

Teng Xiao,[1,2] **Shangsong Liang,**[1,2,*] **Weizhou Shen,**[1,2] **Zaiqiao Meng**[1,2]
[1]School of Data and Computer Science, Sun Yat-sen University, China
[2]Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou 510006, China
{cstengxiao, liangshangsong}@gmail.com, shenwzh3@mail2.sysu.edu.cn, zqmeng@aliyun.com

Abstract

In this paper, we propose a **B**ayesian **D**eep **C**ollaborative **M**atrix **F**actorization (BDCMF) algorithm for collaborative filtering (CF). BDCMF is a novel Bayesian deep generative model that learns user and item latent vectors from users' social interactions, contents of items as the auxiliary information and user-item rating (feedback) matrix. It alleviates the problem of matrix sparsity by incorporating items' auxiliary and users' social information into the model. It can learn more robust and dense latent representations by integrating deep learning into Bayesian probabilistic framework. As being one of deep generative models, it has both non-linearity and Bayesian nature. Additionally, in BDCMF, we derive an efficient EM-style point estimation algorithm for parameter learning. To further improve recommendation performance, we also derive a full Bayesian posterior estimation algorithm for inference. Experiments conducted on two sparse datasets show that BDCMF can significantly outperform the state-of-the-art CF methods.

Introduction

Social media, e.g., Facebook and Twitter, has greatly influenced people's daily lives, and thus building an effective Recommender System (RS) for them has attracted many interests in recent years. The most used technique in RS is collaborative filtering (CF), which makes use of users' ratings over items. Matrix factorization (MF) (Mnih and Salakhutdinov 2008) is one of the most commonly used CF methods due to its effectiveness and scalability. The goal of MF is to learn latent factors of both users and items from user-item feedback matrix. However, traditional MF methods suffer from the matrix sparsity problem – recommendation performance drops dramatically when the user-item feedback matrix is very sparse. In addition, traditional MF methods assume users are independent and identically distributed, and ignore the social connections among users.

To alleviate the matrix sparsity problem, many social MF methods such as those in (Park, Kim, and Choi 2013; Adams, Dahl, and Murray 2014) incorporate auxiliary information, e.g., content information of items, into MF. However, these methods incorporate auxiliary information as a linear regularization term, resulting in the fact that learning latent representations of users and items is not effective when the auxiliary information is also sparse. Aiming at learning more effective representations, many CF-based models using different types of auxiliary information, such as latent Dirichlet allocation (LDA) (Wang and Blei 2011), stacked denoising autoencoder (SDAE) (Wang, Wang, and Yeung 2015; Zhang et al. 2016), variational autoencoder(Li and She 2017) and marginalized denoising autoencoder (Li, Kawale, and Fu 2015), have been proposed to obtain items' latent content representations. Nevertheless, all these models ignore social connections among users.

In order to jointly utilize item contents and social network information, some hybrid MF methods have been proposed, which can be roughly classified into two categories: i.e., Collaborative Topic Regression (CTR)-based methods (Wang and Blei 2011; Chen et al. 2014; Purushotham, Liu, and Kuo 2012; Wang, Chen, and Li 2013; Xu et al. 2017) and Collaborative Deep Learning (CDL)-based methods (Wang, Wang, and Yeung 2015; Zhang et al. 2016; Nguyen and Lauw 2017). Although CTR-based and CDL-based methods have achieved great performance, the full Bayesian posterior estimation is intractable in these models. All these models need to maximize a point estimation for parameter learning. However, in this paper, we point out that maximizing such a point estimation actually assumes that the variation distributions are discrete from variational Bayesian perspective (will be discussed later), which leads to a higher variance in prediction and ignores uncertainty in the model parameters (Lim and Teh 2007). This Bayesian estimation also leads to overly optimistic estimates of test-set log-likelihood (Welling, Chemudugunta, and Sutter 2008) and easily overfit the observed data, as shown in the following experiments. These drawbacks limit CTR-based and CDL-based methods to learn better latent representations from sparse datasets.

To tackle the above problems, in this paper, we aim to build a full Bayesian deep framework to jointly learn latent factors of users and items from three source data, i.e., item content, user-item feedback and user social matrices. We propose a **B**ayesian **D**eep **C**ollaborative **M**atrix **F**actorization model, abbreviated as BDCMF, which integrates item contents and user social information into probabilistic matrix factorization (PMF). Different from CTR-

[*]Shangsong Liang is the corresponding author.

based and CDL-based methods which utilize LDA and SDAE to capture items' latent content vectors, our BDCMF is a full deep generative model and considers the various item content information to be generated by items' latent content factors through a generative network. In addition, with both item content and social information, BDCMF can effectively tackle the matrix sparsity problem. In our BDCMF, to infer latent factors of users and items, we first propose a novel variational EM algorithm from a Bayesian point estimation perspective. Due to the Bayesian nature of BDCMF and drawbacks of Bayesian point estimation, we also propose Bayesian posterior estimation to infer the posterior distributions of users' and items' latent factors. To sum up, our main contributions are:

- We propose a novel deep generative model called BDCMF, which jointly learns latent factors of users and items from content, feedback and social matrices.
- We derive an efficient parallel variational EM-style algorithm from Bayesian point estimation perspective to infer latent factors of users and items.
- To obtain high-quality latent factors, we derive a full Bayesian variational posterior estimation algorithm that can infer posterior distribution of latent factors.
- Comprehensive experiments conducted on two large real-world datasets show BDCMF can significantly outperform state-of-the-art hybrid MF methods for CF.

Related Work

Matrix factorization (MF) is one of the most used approaches in collaborative filtering (CF) and other applications such as rank aggregation (Liang et al. 2014; 2018a). Since the traditional MF suffers from matrix sparsity problem, some previous works explore items' content information to improve the performance. Ma et al. (2008) proposed SocRec, which incorporates social information into probabilistic MF. Wang and Blei (2011) proposed CTR, which explores LDA to capture text topics, and integrates it into probabilistic MF. Wang, Wang, and Yeung (2015) proposed CDL which incorporates SDAE (Stack Denoising Auto-Encoder) into MF framework. To consider both user social information and item content information, many CTR-based and CDL-based methods have been proposed. Purushotham, Liu, and Kuo (2012) proposed CTR-SMF, which incorporates CTR and social matrix factorization to improve performance. In C-CTR-SMF2 (Chen et al. 2014), social context information was integrated into CTR. de Souza da Silva, Langseth, and Ramampiaro (2017) proposed PoissonMF-CS, which utilizes Poisson matrix factorization to model users' preferences and items' contents. Ren et al. proposed sCVR (Ren et al. 2017) to predict item ratings based on user opinions and social relations. Recently, some previous works utilize neural networks to model latent factors of users and items, due to their non-linear modelling abilities. These include NeuMF (Neural Matrix factorization) (He et al. 2017) and CDAE (Collaborative Denoising Auto-Encoder) (Wu et al. 2016). However, the stacked neural network structures of them make them difficult to train and incur high computational cost. In contract, our BDCMF incorporates neural network into Bayesian generative framework, which makes it have non-linear modelling abilities and to be trained effectively.

Problem Definition and Notations

Let $\mathbf{R} \in \{0,1\}^{N\times M}$ be a user-item matrix, where N and M are the number of users and items, respectively. $R_{ij} = 1$ denotes the implicit feedback from user i over item j is observed and $R_{ij} = 0$ otherwise. We use $\mathbf{R}_{\cdot j}$ to represent the j-th column of $\mathbf{R}$. Let $\mathbf{S} = \{S_{ik}\}^{N\times N}$ denote the social matrix of a social network graph $\mathcal{G}$, where $S_{ik} = 1$ if user i is associated with user k and $S_{ik} = 0$ otherwise. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M] \in \mathbb{R}^{L\times M}$ represent item content matrix, where L denotes the dimension of content vector $\mathbf{x}_j$, and $\mathbf{x}_j$ be the content information of item j. For example, if item j is a product or a music, the content x_j can be bag-of-word representation of its tags. We use $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N] \in \mathbb{R}^{D\times N}$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M] \in \mathbb{R}^{D\times M}$ to denote user and item latent matrices, respectively, where D denotes the latent dimension. $\mathbf{G} = [\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_N] \in \mathbb{R}^{D\times N}$ denotes the social factor latent matrix. $\mathbf{I}_D$ represents identity matrix with dimension D. The problem we aim to address is: given a user-item matrix $\mathbf{R}$, item content matrix $\mathbf{X}$ and social matrix $\mathbf{S}$, infer user latent matrix $\mathbf{U}$, social factor latent matrix $\mathbf{G}$ and item latent matrix $\mathbf{V}$ such that missing value R_{ij} in $\mathbf{R}$ can be effectively predicted.

Bayesian Deep Collaborative Matrix Factorization

In this section, we propose a **B**ayesian **D**eep **C**ollaborative **M**atrix **F**actorization (BDCMF) for recommendation, the goal of which is to infer user latent matrix $\mathbf{U}$, social factor latent matrix $\mathbf{G}$, and item latent matrix $\mathbf{V}$ given item content matrix $\mathbf{X}$, user social matrix $\mathbf{S}$, and feedback matrix $\mathbf{R}$.

The Proposed Model

Similar to (Ma et al. 2008), we consider user-item and social matrices, $\mathbf{R}$ and $\mathbf{S}$, sharing the same user latent matrix $\mathbf{U}$, such that $\mathbf{R}$ is factorized by user and item latent matrices, $\mathbf{U}$ and $\mathbf{V}$, via probabilistic matrix factorization (PMF) (Mnih and Salakhutdinov 2008), and $\mathbf{S}$ is factorized by user latent and social factor latent matrices, $\mathbf{U}$ and $\mathbf{G}$. For items' content information, since it can be very complex and various, we don't know its real distribution. However, we know any distribution can be generated by mapping simple Gaussian through a sufficiently complicated function (Doersch 2016). In our proposed model, we consider item contents to be generated by their latent content vector through a generative network. The generative process of BDCMF is:

1. For each user i, draw user latent vector $\mathbf{u}_i \sim \mathcal{N}(0, \lambda_u \mathbf{I}_D)$.
2. For each social factor k, draw its latent vector, i.e., the social latent vector $\mathbf{g}_k \sim \mathcal{N}(0, \lambda_g \mathbf{I}_D)$.
3. For each item j:
   (a) Draw item content latent vector $\mathbf{z}_j \sim \mathcal{N}(0, \mathbf{I}_D)$.
   (b) Draw item content vector $p_\theta(\mathbf{x}_j|\mathbf{z}_j)$.
   (c) Draw item latent offset $\mathbf{k}_j \sim \mathcal{N}(0, \lambda_v \mathbf{I}_D)$ and set the item latent vector as $\mathbf{v}_j = \mathbf{z}_j + \mathbf{k}_j$.

4. For each user-item pair (i, j) in $\mathbf{R}$, draw R_{ij}:

$$R_{ij} \sim \mathcal{N}(\mathbf{u}_i^\top \mathbf{v}_j, c_{ij}^{-1}). \tag{1}$$

5. For each user-social factor pair (i, k) in $\mathbf{S}$, draw S_{ik}:

$$S_{ik} \sim \mathcal{N}(\mathbf{u}_i^\top \mathbf{g}_k, \lambda_q^{-1} e_{ik}^{-1}). \tag{2}$$

In the process, $\lambda_v, \lambda_u, \lambda_g$, and λ_q are the free parameters, respectively. Specifically, λ_q is a trade-off between the social matrix and user-item matrix for user latent factor (will be discussed later). Similar to (Wang and Blei 2011; Wang, Wang, and Yeung 2015), c_{ij} in Eq.1 and e_{ik} in Eq.2 serves as confident parameters for R_{ij} and S_{ik}, respectively:

$$c_{ij} = \begin{cases} \varphi_1 & \text{if} \quad R_{ij} = 1, \\ \varphi_2 & \text{if} \quad R_{ij} = 0, \end{cases} \tag{3}$$

$$e_{ik} = \begin{cases} \varphi_3 & \text{if} \quad S_{ik} = 1, \\ \varphi_4 & \text{if} \quad S_{ik} = 0, \end{cases} \tag{4}$$

where $\varphi_1 > \varphi_2 > 0$ and $\varphi_3 > \varphi_4 > 0$ are free parameters. In our model, we follow (Wang, Wang, and Yeung 2015; Purushotham, Liu, and Kuo 2012) to set $\varphi_1 = \varphi_3 = 1$ and $\varphi_2 = \varphi_4 = 0.1$. $p_\theta(\mathbf{x}_j|\mathbf{z}_j)$ represents item content information and $\mathbf{x}_j$ is generated from latent content vector $\mathbf{z}_i$ through a generative neural network parameterized by θ. It should be noted that the specific form of the probability $p_\theta(\mathbf{x}_j|\mathbf{z}_j)$ depends on the type of the item content vector. For instance, if $\mathbf{x}_j$ is binary vector, $p_\theta(\mathbf{x}_j|\mathbf{z}_j)$ can be a multivariate Bernoulli distribution $\mathrm{Ber}(F_\theta(\mathbf{z}_j))$ with $F_\theta(\mathbf{z}_j)$ being the highly no-linear function parameterized by θ.

According to the graphic model in Fig.1, the joint probability of $\mathbf{R}, \mathbf{S}, \mathbf{X}, \mathbf{U}, \mathbf{V}, \mathbf{G}, \mathbf{Z}$ can be represented as:

$$p(\mathcal{O}, \mathcal{Z}) = \prod_{i=1}^{N} \prod_{j=1}^{M} \prod_{k=1}^{N} p(\mathcal{O}_{ijk}, \mathcal{Z}_{ijk}) = \prod_{i=1}^{N} \prod_{j=1}^{M} \prod_{k=1}^{N} p(\mathbf{z}_j) p(\mathbf{g}_k) p(\mathbf{u}_i) p_\theta(\mathbf{x}_j|\mathbf{z}_j) p(\mathbf{v}_j|\mathbf{z}_j) p(R_{ij}|\mathbf{u}_i, \mathbf{v}_j) p(S_{ik}|\mathbf{u}_i, \mathbf{g}_k), \tag{5}$$

where $\mathcal{O} = \{\mathbf{R}, \mathbf{S}, \mathbf{X}\}$ is the set of all observed variables, $\mathcal{Z} = \{\mathbf{U}, \mathbf{V}, \mathbf{G}, \mathbf{Z}\}$ is the set of all latent variables needed to be inferred, and $\mathcal{O}_{ijk} = \{R_{ij}, S_{ik}, \mathbf{x}_j\}$ and $\mathcal{Z}_{ijk} = \{\mathbf{u}_i, \mathbf{v}_j, \mathbf{g}_k, \mathbf{z}_j\}$ for short.

Bayesian Point Estimation

Previous works (Wang, Wang, and Yeung 2015; Wang and Blei 2011) have shown that using an EM-style algorithm enables recommendation methods that integrate them to obtain high-quality latent vectors (in our case, $\mathbf{U}$ and $\mathbf{V}$). Inspired by these work, in this section, we first derive an EM-style algorithm called BDCMF-1 from the view of Bayesian point estimation. The marginal log likelihood can be given by:

$$\log p(\mathcal{O}) = \log \int p(\mathcal{O}, \mathcal{Z}) \mathrm{d}\mathcal{Z} \geq \int q(\mathcal{Z}) \log \frac{p(\mathcal{O}, \mathcal{Z})}{q(\mathcal{Z})} \mathrm{d}\mathcal{Z} = \int q(\mathcal{Z}) \log p(\mathcal{O}, \mathcal{Z}) - \int q(\mathcal{Z}) \log q(\mathcal{Z}) \equiv \mathcal{L}(q), \tag{6}$$

where we apply Jensen's inequality, and $q(\mathcal{Z})$ and $\mathcal{F}(q)$ are variational distribution and the evidence lower bound

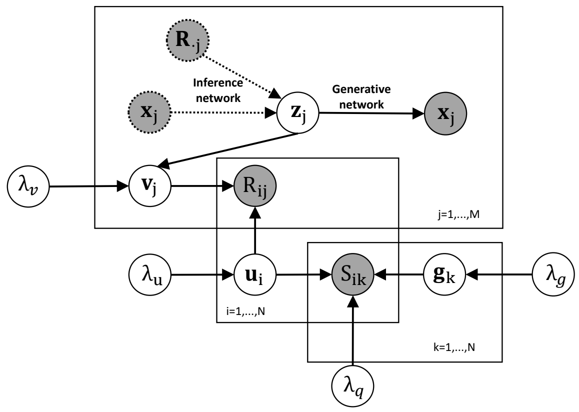

Figure 1: Graphical model of our BDCMF. Solid and dashed lines represent generative and inference process, respectively. Shaded nodes represent observed variables.

(ELBO), respectively. For variational distribution $q(\mathcal{Z})$, we consider variational distributions in it to be matrix-wise independent:

$$q(\mathcal{Z}) = q(\mathbf{U}) q(\mathbf{V}) q(\mathbf{G}) q(\mathbf{Z}) = \prod_{i=1}^{N} q(\mathbf{u}_i) \prod_{j=1}^{M} q(\mathbf{v}_j) \prod_{k=1}^{N} q(\mathbf{g}_k) \prod_{j=1}^{M} q(\mathbf{z}_j). \tag{7}$$

For Bayesian point estimation, we assume the variational distribution of $\mathbf{u}_i$ is:

$$q(\mathbf{u}_i) = \prod_{d=1}^{D} \delta(U_{id} - \hat{U}_{id}), \tag{8}$$

where $\{\hat{U}_{id}\}_{d=1}^{D}$ are variational parameters and δ is a Dirac delta function. Variational distributions of $\mathbf{v}_j$ and $\mathbf{g}_k$ are defined similarly. When U_{id} is discrete (The continuous situation will be discussed in next section), the entropy of $\mathbf{u}_i$ is:

$$H(\mathbf{u}_i) = - \int q(\mathbf{u_i}) \log q(\mathbf{u}_i) = \sum_{d=1}^{D} \sum_{U_{id}} \delta(U_{id} - \hat{U}_{id}) \log \delta(U_{id} - \hat{U}_{id}) = 0. \tag{9}$$

Similarly, $H(\mathbf{v}_j)$ and $H(\mathbf{g}_k)$ are 0 when the elements are discrete. Then the evidence lower bound $\mathcal{L}(q)$ (Eq. 6) can be written as:

$$\mathcal{L}_{\text{point}}(\hat{\mathbf{U}}, \hat{\mathbf{V}}, \hat{\mathbf{G}}, \theta, \phi) = \langle \log p(\mathbf{U}) p(\mathbf{G}) p(\mathbf{V}|\mathbf{Z}) p(\mathbf{X}|\mathbf{Z}) p(\mathbf{R}|\mathbf{U}, \mathbf{V}) p(\mathbf{S}|\mathbf{U}, \mathbf{G}) \rangle_q - \mathrm{KL}(q_\phi(\mathbf{Z}|\mathbf{X}) || p(\mathbf{Z})), \tag{10}$$

where $\langle \cdot \rangle$ is the statistical expectation with respect to the corresponding variational distribution. $\hat{\mathbf{U}} = \left\{ \hat{U}_{id} \right\}$, $\hat{\mathbf{V}} = \left\{ \hat{V}_{jd} \right\}$ and $\hat{\mathbf{G}} = \left\{ \hat{G}_{kd} \right\}$ are variational parameters corresponding to the variational distribution $q(\mathbf{U})$, $q(\mathbf{V})$ and $q(\mathbf{G})$, respectively.

For latent variables $\mathbf{Z}$, we use a generative network to represent the distribution $p_\theta(\mathbf{X}|\mathbf{Z})$. As discussed in (Kingma and Welling 2014), traditional variational-EM algorithm (Neal and Hinton 1998) is intractable to infer the posterior of $\mathbf{Z}$. Consequently, similar to VAE (Kingma and Welling 2014), we also introduce a variational distribution $q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{R})$ to approximate the true posterior distribution $p(\mathbf{Z}|\mathcal{O})$. $q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{R})$ is implemented by a inference neural network parameterized by ϕ (see Fig.1). Specifically, for $\mathbf{z}_j$ we have:

$$q(\mathbf{z}_j) = q_\phi(\mathbf{z}_j|\mathbf{x}_j, \mathbf{R}_{\cdot j}) = \mathcal{N}(\mu_j, \mathrm{diag}(\delta_j^2)), \tag{11}$$

where the mean μ_j and variance δ_j are the outputs of the inference neural network.

Directly maximizing the ELBO (Eq. 10) involves solving parameters $\hat{\mathbf{U}}$, $\hat{\mathbf{V}}$ $\hat{\mathbf{G}}$, θ and ϕ, which is intractable. Thus, we derive an iterative variational-EM (VEM) algorithm to maximize $\mathcal{L}_{\text{point}}(\hat{\mathbf{U}}, \hat{\mathbf{V}}, \hat{\mathbf{G}}, \theta, \phi)$, abbreviated $\mathcal{L}_{\text{point}}$.

Variational E-step. We first keep θ and ϕ fixed, then optimize evidence lower bound $\mathcal{L}_{\text{point}}$ with respect to $\hat{\mathbf{U}}$, $\hat{\mathbf{V}}$ and $\hat{\mathbf{G}}$. The updating rules of $\hat{\mathbf{u}}_i$, $\hat{\mathbf{v}}_j$ and $\hat{\mathbf{g}}_k$ are (we omit the derivation due to space limitations):

$$\hat{\mathbf{u}}_i \leftarrow (\hat{\mathbf{V}}\mathbf{C}_i\hat{\mathbf{V}}^\top + \lambda_q\hat{\mathbf{G}}\mathbf{E}_i\hat{\mathbf{G}}^\top + \lambda_u\mathbf{I}_D)^{-1}\cdot (\hat{\mathbf{V}}\mathbf{C}_i\mathbf{R}_i + \lambda_q\hat{\mathbf{G}}\mathbf{E}_i\mathbf{S}_i), \tag{12}$$

$$\hat{\mathbf{v}}_j \leftarrow (\hat{\mathbf{U}}\mathbf{C}_j\hat{\mathbf{U}}^\top + \lambda_v\mathbf{I}_D)^{-1}(\hat{\mathbf{U}}\mathbf{C}_j\mathbf{R}_j + \lambda_v\langle\mathbf{z}_j\rangle), \tag{13}$$

$$\hat{\mathbf{g}}_k \leftarrow (\lambda_q\hat{\mathbf{U}}\mathbf{E}_k\hat{\mathbf{U}}^\top + \lambda_g\mathbf{I}_D)^{-1}(\lambda_q\hat{\mathbf{U}}\mathbf{E}_k\mathbf{S}_k), \tag{14}$$

where $\mathbf{C}_i = \mathrm{diag}(c_{i1}, ...c_{iM})$, $\mathbf{E}_i = \mathrm{diag}(e_{i1}, ...e_{iN})$, $\mathbf{R}_i = [R_{i1}, ...R_{iM}]$ and $\mathbf{S}_i = [S_{i1}, ...S_{iN}]$. For item latent vector $\mathbf{v}_j$ and social latent vector $\mathbf{g}_k$, $\mathbf{C}_j$, $\mathbf{R}_j$, $\mathbf{E}_k$ and $\mathbf{S}_k$ are defined similarly. $\hat{\mathbf{u}}_i = [\hat{U}_{i1}, ...\hat{U}_{iD}]$, $\hat{\mathbf{v}}_j = [\hat{V}_{j1}, ...\hat{V}_{jD}]$ and $\hat{\mathbf{g}}_k = [\hat{G}_{k1}, ...\hat{G}_{kD}]$. For $\mathbf{z}_j$, its expectation is $\langle\mathbf{z}_j\rangle = \mu_j$, which is the output of the inference network.

It can be observed that λ_v governs how much the latent item vector $\mathbf{z}_j$ affects item latent vector $\mathbf{v}_j$. For example, if $\lambda_v = \infty$, it indicates we direct use latent item vector to represent item latent vector $\mathbf{v}_j$; if $\lambda_v = 0$, it means we do not embed any item content information into item latent vector. λ_q serves as a balance parameter between social network matrix and user-item matrix on user latent vector $\mathbf{u}_i$. For example, if $\lambda_q = \infty$, it means we only use the social network information to model user's preference; if $\lambda_q = 0$, we only use user-item matrix and item content information for prediction. Thus, λ_v and λ_q are regarded as collaborative parameters for item content, feedback and social matrices.

Variational M-step. Keep $\hat{\mathbf{U}}$, $\hat{\mathbf{V}}$ and $\hat{\mathbf{G}}$ fixed, we optimize the following $\mathcal{L}_{\text{point}}$ w.r.t. ϕ and θ (we only focus on terms containing ϕ and θ):

$$\mathcal{L}_{\text{point}} = \text{constant} + \sum\nolimits_{j=1}^{M} \mathcal{L}(\theta, \phi; \mathbf{x}_j, \mathbf{v}_j, \mathbf{R}_{\cdot\mathbf{j}}) = \text{constant} + \sum\nolimits_{j=1}^{M} -\frac{\lambda_v}{2}\langle(\mathbf{v}_j - \mathbf{z}_j)^\top(\mathbf{v}_j - \mathbf{z}_j)\rangle_{q(\mathcal{Z})} + \langle\log p_\theta(\mathbf{x}_j|\mathbf{z}_j)\rangle_{q_\phi(\mathbf{z}_j|\mathbf{x}_j,\mathbf{R}_{\cdot j})} - \mathrm{KL}(q_\phi(\mathbf{z}_j|\mathbf{x}_j, \mathbf{R}_{\cdot j})||p(\mathbf{z}_j)), \tag{15}$$

where M is the number of items and the constant term represents terms which do not contain θ and ϕ. For the expectation term $\langle p_\theta(\mathbf{x}_j|\mathbf{z}_j)\rangle_{q_\phi(\mathbf{z}_j|\mathbf{x}_j,\mathbf{R}_{\cdot j})}$, we can not solve it analytically. To handle this problem, we approximate it by the Monte Carlo sampling as follows:

$$\langle\log p_\theta(\mathbf{x}_j|\mathbf{z}_j)\rangle_{q_\phi(\mathbf{z}_j|\mathbf{x}_j,\mathbf{R}_{\cdot j})} = \frac{1}{L}\sum\nolimits_{l=1}^{L} p_\theta(\mathbf{x}_j|\mathbf{z}_j^l), \tag{16}$$

where L is the size of samplings, and $\mathbf{z}_j^l$ denotes the l-th sample, which is reparameterized to $\mathbf{z}_j^l = \epsilon_j^l \odot \mathrm{diag}(\delta_j^2) + \mu_j$. Here ϵ_j^l is drawn from $\mathcal{N}(0, \mathbf{I}_D)$ and $\odot$ is an element-wise multiplication. By using this reparameterization trick and Eq 11, $\mathcal{L}(\theta, \phi; \mathbf{x}_j, \mathbf{v}_j, \mathbf{R}_{\cdot\mathbf{j}})$ in Eq 15 can be estimated by:

$$\mathcal{L}(\theta, \phi; \mathbf{x}_j, \mathbf{v}_j, \mathbf{R}_{\cdot\mathbf{j}}) \simeq \tilde{\mathcal{L}}^j(\theta, \phi) = -\frac{\lambda_v}{2}(-2\mu_j^\top\hat{\mathbf{v}}_j + \mu_j^\top\mu_j + \mathrm{tr}(\mathrm{diag}(\delta_j^2))) + \frac{1}{L}\sum\nolimits_{l=1}^{L} p_\theta(\mathbf{x}_j|\mathbf{z}_j^l) - \mathrm{KL}(q_\phi(\mathbf{z}_j|\mathbf{x}_j, \mathbf{R}_{\cdot j})||p(\mathbf{z}_j)) + \text{constant}. \tag{17}$$

We can construct an estimator of $\mathcal{L}_{\text{point}}(\phi, \theta; \mathbf{X}, \mathbf{V}, \mathbf{R})$, based on minibatches:

$$\mathcal{L}_{\text{point}}(\theta, \phi) \simeq \tilde{\mathcal{L}}^P(\theta, \phi) = \frac{M}{P}\sum\nolimits_{j=1}^{P} \tilde{\mathcal{L}}^j(\theta, \phi). \tag{18}$$

As discussed in (Kingma and Welling 2014), the number of samplings L per item j can be set to 1 as long as the minibatch size P is large enough, e.g., $P = 100$. We can update θ and ϕ by using the gradient $\nabla_{\theta,\phi}\tilde{\mathcal{L}}^P(\theta, \phi)$.

We iteratively update $\mathbf{U}$, $\mathbf{V}$, $\mathbf{G}$, θ, and ϕ until it converges. Overview of our Bayesian point estimation is summarized in algorithm 1.

Bayesian Posterior Estimation

We assume $\hat{\mathbf{U}}$, $\hat{\mathbf{V}}$ and $\hat{\mathbf{G}}$ are discrete variables. When these variables are continuous, as discussed in pervious work (Welling, Chemudugunta, and Sutter 2008), the Bayesian point estimation will lead overly to optimistic estimates of test-set log-likelihood. This problem is exactly that pervious well-known recommendation models such CTR-based and CDL-based models (Wang, Wang, and Yeung 2015; Wang and Blei 2011; Purushotham, Liu, and Kuo 2012; Chen et al. 2014; Zhang et al. 2016) which use point estimation algorithm suffer from. In addition, point estimators leads to higher variance in prediction and ignore uncertainty in the model parameters. Unlike CTR-based and CDL-based recommendation methods, BDCMF can be easily estimated by a full Bayesian posterior estimator.

Here, we derived Bayesian posterior estimation learning algorithm called BDCMF-2. For Bayesian posterior estimation, we also assume variational distributions are matrix-wise independent:

$$q(\mathcal{Z}) = q(\mathbf{U})q(\mathbf{V})q(\mathbf{G})q(\mathbf{Z}) = \prod\nolimits_{i=1}^{N} q(\mathbf{u}_i)\prod\nolimits_{j=1}^{M} q(\mathbf{v}_j)\prod\nolimits_{k=1}^{N} q(\mathbf{g}_k)\prod\nolimits_{j=1}^{M} q(\mathbf{z}_j), \tag{19}$$

For variational distributions $p(\mathbf{z}_j)$, we also introduce a variational distribution $q_\phi(\mathbf{Z}|\mathbf{X})$ to approximate the true posterior distribution $p(\mathbf{Z}|\mathcal{O})$ like Bayesian point estimation. $q_\phi(\mathbf{Z}|\mathbf{X})$ is parameterized by ϕ, an inference neural network (see Fig.1). For $\mathbf{z}_j$:

$$q(\mathbf{z}_j) = q_\phi(\mathbf{z}_j|\mathbf{x}_j, \mathbf{R}_{\cdot j}) = \mathcal{N}(\mu_j, \mathrm{diag}(\delta_j^2)). \tag{20}$$

It should be noted we do not assume the specific forms of variational distributions for $q(\mathbf{u}_i)$, $q(\mathbf{v}_j)$ and $q(\mathbf{g}_k)$ in Bayesian posterior estimation. Our goal is also to maximize the evidence lower bound $\mathcal{L}(q)$ (Eq. 6). The algorithm also contains variational E-step and M-step.

Variational E-step. we fix θ and ϕ, then optimize evidence lower bound (6) w.r.t $\mathbf{u}_i$, $\mathbf{v}_j$ and $\mathbf{g}_k$. Taking the derivative of lower bound (Eq. 6) w.r.t. $q(\mathbf{u}_i)$ and setting it to zero, then we can get:

$$q(\mathbf{u}_i) \propto \exp\left\{\langle \log p(\mathcal{O}, \mathcal{Z})\rangle_{q(\mathcal{Z}\setminus \mathbf{u}_i)}\right\} = \mathcal{N}(\overline{\mathbf{u}}_i, \mathbf{\Lambda}_i^u)\,, \quad (21)$$

where the expectation is taken with respect to the variational distributions over all latent variables excluding $\mathbf{u}_i$. Thus, the updating rules of $\mathbf{u}_i$ are:

$$q(\mathbf{U}) = \prod\nolimits_{i=1}^{M} \mathcal{N}(\overline{\mathbf{u}}_i, \mathbf{\Lambda}_i^u), \text{where} \quad (22)$$

$$\mathbf{\Lambda}_i^u \leftarrow (\langle \mathbf{V}\rangle \mathbf{C}_i \langle \mathbf{V}\rangle^\top + \sum\nolimits_{j=1}^{M} c_{ij}\mathbf{\Lambda}_j^v + \lambda_q(\langle \mathbf{G}\rangle \mathbf{E}_i \langle \mathbf{G}\rangle^\top$$
$$+ \sum\nolimits_{k=1}^{N} e_{ik}\mathbf{\Lambda}_k^g) + \lambda_u \mathbf{I}_D)^{-1}, \quad (23)$$

$$\overline{\mathbf{u}}_i \leftarrow \mathbf{\Lambda}_i^u(\langle \mathbf{V}\rangle \mathbf{C}_i \mathbf{R}_i + \lambda_q \langle \mathbf{G}\rangle \mathbf{E}_i \mathbf{S}_i). \quad (24)$$

Similarly, the updating rules of $\mathbf{V}$ and $\mathbf{G}$ are:

$$q(\mathbf{V}) = \prod\nolimits_{j=1}^{M} \mathcal{N}(\overline{\mathbf{v}}_j, \mathbf{\Lambda}_j^v), \text{where} \quad (25)$$

$$\mathbf{\Lambda}_j^v \leftarrow (\langle \mathbf{U}\rangle \mathbf{C}_j \langle \mathbf{U}\rangle^\top + \sum\nolimits_{i=1}^{N} c_{ij}\mathbf{\Lambda}_i^u + \lambda_v \mathbf{I}_D)^{-1}, \quad (26)$$

$$\overline{\mathbf{v}}_j \leftarrow \Lambda_j^v(\langle \mathbf{U}\rangle \mathbf{C}_j \mathbf{R}_j + \lambda_v \langle \mathbf{z}_j\rangle). \quad (27)$$

$$q(\mathbf{G}) = \prod\nolimits_{k=1}^{N} \mathcal{N}(\overline{\mathbf{g}}_k, \mathbf{\Lambda}_k^g), \text{where} \quad (28)$$

$$\mathbf{\Lambda}_k^g \leftarrow (\lambda_q(\langle \mathbf{U}\rangle \mathbf{E}_k \langle \mathbf{U}\rangle^\top + \sum\nolimits_{i=1}^{N} e_{ik}\mathbf{\Lambda}_i^u)$$
$$+ \lambda_g \mathbf{I}_D)^{-1}, \quad (29)$$

$$\overline{\mathbf{g}}_k \leftarrow \mathbf{\Lambda}_k^g(\lambda_q \langle \mathbf{U}\rangle \mathbf{E}_k \mathbf{S}_k). \quad (30)$$

Variational M-step. we only focus parameters θ and ϕ. The M-step is as same as Bayesian point estimation. By using the reparameterization trick as we did in Bayesian point estimation, the objection function w.r.t θ and ϕ is:

$$\mathcal{L}(q) = \text{constant} + \sum\nolimits_{j=1}^{M} \mathcal{L}(\theta, \phi; \mathbf{x}_j, \mathbf{v}_j, \mathbf{R}_{\cdot j}) = \text{constant}$$
$$+ \sum\nolimits_{j=1}^{M} -\frac{\lambda_v}{2}\langle (\mathbf{v}_j - \mathbf{z}_j)^\top (\mathbf{v}_j - \mathbf{z}_j)\rangle_{q(\mathcal{Z})} + \quad (31)$$
$$\langle \log p_\theta(\mathbf{x}_j|\mathbf{z}_j)\rangle_{q_\phi(\mathbf{z}_j|\mathbf{x}_j, \mathbf{R}_{\cdot j})} - \mathrm{KL}(q_\phi(\mathbf{z}_j|\mathbf{x}_j, \mathbf{R}_{\cdot j})||p(\mathbf{z}_j))\,.$$

We can optimize it as the same as we did in Bayesian point estimation.

Discussion

According to Algorithm 1, the time complexity of updating $\mathbf{g}_k$ is $O(ND^2 + D^3)$, where N is the number of users and D is the dimension of latent space. The time complexity of updating $\mathbf{v}_j$ is $O(ND^2 + D^3 + LP^2)$, where L is the number of layers in neural network and P is the average dimension of these layers. The time complexity of updating $\mathbf{u}_i$ is $O(MD^2 + ND^2 + D^3)$, where M is the number of items. Because $\hat{\mathbf{V}}\mathbf{C}_i\hat{\mathbf{V}}^\top$ in Eq. 12 can be rewritten as $\hat{\mathbf{V}}(\mathbf{C}_i - \varphi_2\mathbf{I})\hat{\mathbf{V}}^\top + \varphi_2\hat{\mathbf{V}}\hat{\mathbf{V}}^\top$, the complexity of updating $\mathbf{g}_k$ decreases from $O(ND^2 + D^3)$ to $O(N_0D^2 + D^3)$, where N_0 is observed rating numbers with respect to item j which is relatively small in sparse matrix. $\hat{\mathbf{U}}\mathbf{C}_j\hat{\mathbf{U}}^\top$ and $\hat{\mathbf{U}}\mathbf{E}_k\hat{\mathbf{U}}^\top$ can be rewritten, similarly. In fact, the latent dimensions D is also relatively small (<200). Therefore, our BDCMF can be very efficient for large datasets. Additionally, our model is flexible to handle different types of contents in different scenarios by replacing the architecture of neural networks for inferring items' latent content representations to the other neural networks such as recurrent neural network.

Algorithm 1 BDCMF-1 inference algorithm.

```
Require: user-item matrix $\mathbf{R}$, item content matrix $\mathbf{X}$, parameters $\lambda_v$, $\lambda_u$, $\lambda_g$, and $\lambda_q$, and $P = 100$ and $L = 1$.
 1: Randomly initialize variational parameters $\hat{\mathbf{u}}_i$, $\hat{\mathbf{v}}_j$, $\hat{\mathbf{g}}_k$ and network parameters $\theta$ and $\phi$.
 2: while not converged do
 3:    for $i = 1$ to $N$ do
 4:       update $\hat{\mathbf{u}}_i$ using Eq.12.
 5:    end for
 6:    for $j = 1$ to $M$ do
 7:       update $\hat{\mathbf{v}}_j$ using Eq.13.
 8:    end for
 9:    for $k = 1$ to $N$ do
10:       update $\hat{\mathbf{g}}_k$ using Eq.14.
11:    end for
12:    randomly draw $P$ triples, i.e., $\{(\mathbf{x}_p, \mathbf{v}_p, \mathbf{R}_{\cdot p})\}_{p=1}^P$ from the dataset.
13:    $\epsilon \leftarrow$ draw $P$ samples from $\mathcal{N}(0, \mathbf{I}_D)$, with each for a triple $(\mathbf{x}_p, \mathbf{v}_p, \mathbf{R}_{\cdot p})$.
14:    $\theta, \phi \leftarrow$ update $\theta, \phi$ using the gradient $\nabla_{\theta,\phi}\tilde{\mathcal{L}}^P(\theta, \phi)$.
15: end while
```

Prediction

After Algorithm 1 is converged, we predict the missing value R_{ij} in $\mathbf{R}$ by using the learned latent features $\mathbf{u}_i$ and $\mathbf{v}_j$:

$$R_{ij}^* = \langle R_{ij}\rangle = (\langle \mathbf{z}_j\rangle + \langle \mathbf{k}_j\rangle)^\top \langle \mathbf{u}_i\rangle = \langle \mathbf{v}_j\rangle^\top \langle \mathbf{u}_i\rangle\,. \quad (32)$$

For a new item that is not rated by any other users, the offset ϵ_j is zero, and we can predict R_{ij} by:

$$R_{ij}^* = \langle R_{ij}\rangle = \langle \mathbf{z}_j\rangle^\top \langle \mathbf{u}_i\rangle\,. \quad (33)$$

Experiments

Experimental Setup

Research Questions. The research questions guiding the remainder of the paper are: (**RQ1**) How does our proposed BDCMF-1 compare to state-of-the-art MF methods for CF on sparsity matrix? (**RQ2**) How do different parameter settings (e.g., the social parameter λ_q and content parameter λ_v) affect BDCMF-1? (**RQ3**) Does the Bayesian posterior estimation outperform Bayesian point estimation?

Datasets. In order to answer our research questions, we conduct experiments on two real-world datasets from

Table 1: Statistics of the datasets.

Datasets	lastfm-2k	delicious-2k
users	1,892	1,876
items	17,632	69,226
tags	11,946	53,388
user-items relations	92,834	104,799
user-user relations	25,434	15,328
user-tags-items	186,479	437,593

Lastfm [1] (*lastfm-2k*) and Delicious[2] (*delicious-2k*) collected by Brusilovsky et al. (2010). Both of the datasets contain user-item, user-user, and user-tag-item relations. We first transform datasets as implicit feedback. For Lastfm dataset, we consider the user-item feedback is 1 if the user has listened to the artist (item); otherwise, it is 0. Similarly, for the Delicious dataset, if the user has bookmarked a URL (item), the user-item feedback is 1; otherwise, the feedback is 0. After doing so, Lastfm and Delicious datasets only contain 0.27% and 0.08% observed feedbacks, respectively, and are extremely sparse. We use items bag-of-word tag representations as their items content information.

Baselines. For fair comparisons, like that in our BDCMF-1 and BDCMF-2 (in what follows, we use BDCMF to refer to these two models), most baselines we used also incorporate user social information or item content information into matrix factorization.

MF-based methods:

(1) PMF. This model (Mnih and Salakhutdinov 2008) is a famous MF method, and only uses user-item feedback matrix.

(2) SoRec. This model (Ma et al. 2008) jointly decomposes user-user social matrix and user-item feedback matrix to learn user and item latent representations.

Content-based methods:

(3) Collaborative topic regression (CTR). This model (Wang and Blei 2011) utilizes topic model and matrix factorization to learn latent representations of users and items.

(4) Collaborative deep learning (CDL). This model (Wang, Wang, and Yeung 2015) utilizes stack denoising autoencoder to learn latent items' content representations, and incorporates them into probabilistic matrix factorization.

Hybrid methods:

(5) CTR-SMF. This model (Purushotham, Liu, and Kuo 2012) incorporates topic modeling and probabilistic MF of social networks.

(6) PoissonMF-CS This model(de Souza da Silva, Langseth, and Ramampiaro 2017), jointly models use social trust, item content and user's preference using Poisson matrix factorization framework. It is a state-of-the-art MF method for Top-N recommendation on the Lastfm dataset.

Neural network-based method:

(7)Neural Matrix Factorization (NeuMF). This model (He et al. 2017) is a state-of-the-art CF method, which utilizes neural network to model the interaction between user and item features.

[1]http://www.lastfm.com.

[2]http://www.delicious.com.

Settings. For fair comparisons, We first set the parameters for PMF, SoRec, CTR, CTR-SMF, CDL, NeuMF via five-fold cross validation. For PMF, we set D, λ_u and λ_v as 50, 0.01 and 0.001, respectively. The parameter settings for SoRec are $\lambda_c = 10$, $\lambda_u = \lambda_v = \lambda_z = 0.001$. For CTR, we find it achieve best performance when D=50, λ_u=0.1, λ_v=100, a=1 and b=0.01. CTR-SMF yields the best performance when D=75, λ_u=0.1, λ_v=100, λ_q=100, a=1 and b=0.01. For CDL, it achieves the best performance when we set $a = 1$, $b = 0.01$, $D = 50$, λ_u=1, λ_v=10, $\lambda_n = 1000$, and λ_w=0.0001. For NeuMF, we set D=50, and the last hidden layer is 16. For PoissonMF-CS, we set λ_c=0.1 and λ_s=0.1. For our model BDCMF, we set D=50. We set λ_v=1, λ_q=10 for dataset Lastfm, and set λ_v=0.1, λ_q=10 for dataset Delicious. To evaluate our model performance on extreme sparse matrix, we use 90% of dataset to train our model and the remainder for testing. The dimensions of the layers of the network for BDCMF are (L+N)-200-100(z_j)-100-100-(L+N), where L and N are the dimension of $\mathbf{x}_j$ and $\mathbf{R}_{\cdot j}$. The '-' denotes the next layer in neural networks.

Evaluation Metrics. The metrics we used are Recall@K, NDCG@K and MAP@K which are common metrics for recommendation and their definitions can be found at (Croft, Metzler, and Strohman 2015; Liang and de Rijke 2016). We report the average performance over all users on the metrics.

Experimental Results and Discussions

Overall Performance (RQ1). Table 2 shows the performance of our BDCMF-1 and the baselines using the two datasets. According to Table 2, we have following findings: a) BDCMF-1 outperforms the baselines in terms of all matrices on Lastfm, which demonstrates the effectiveness of our method of inferring the latent factors of users and items. b) For more sparse dataset, Delicious, BDCMF-1 also achieves the best performance, which demonstrates our model can effectively handle matrix sparsity problem. c) We can see both methods that utilize content and social information (BDCMF-1 and PoissonMF-CS) outperform the others (CDL, CTR, SoRec and PMF), which demonstrates incorporating content and social information can effectively alleviate matrix sparse problem. d) Our BDCMF-1 outperforms state-of-the-art PoissonMF-CS, though they are both Bayesian generative model. The reason is that our BDCMF-1 incorporates neural network into Bayesian generative model, which makes it have powerful non-linearity to model item content's latent representation.

Impact of Parameters (RQ2). Fig. 2(a) and 2(b) show the contour of Recall@50 by varying content parameters λ_v and social parameters λ_q on Lastfm and Delicious datasets. As we can see, BDCMF-1 achieves the best recommendation performance when λ_v=0.1 and λ_q=10 in Lastfm. We also can see that the performance is worse when λ_v >10, as λ_v can control how much item content information is incorporated into item latent vector, and Lastfm is a social media dataset where people have similar preferences with

Table 2: Recommendation performance of BDCMF-1 and the baselines. The best and the second best performance scores per metric per dataset are marked with boldfaces and underlined, respectively.

	Lastfm Dataset				Delicious Dataset			
	Recall@20	Recall@50	NDCG@20	MAP@20	Recall@20	Recall@50	NDCG@20	MAP@20
PMF	0.0923	0.1328	0.0703	0.1083	0.0272	0.0561	0.0286	0.0312
SoRec	0.1088	0.1524	0.0721	0.1128	0.0384	0.0892	0.0323	0.0389
CTR	0.1192	0.1624	0.0799	0.1334	0.0563	0.1342	0.0487	0.0581
CTR-SMF	0.1232	0.1832	0.0823	0.1386	0.0781	0.1564	0.0534	0.0588
CDL	0.1346	0.2287	0.0928	0.1553	0.0861	0.1897	0.0726	0.0792
NeuMF	<u>0.1517</u>	0.2584	0.1036	<u>0.1678</u>	<u>0.1078</u>	<u>0.2267</u>	<u>0.0769</u>	<u>0.0928</u>
PoissonMF-CS	0.1482	<u>0.2730</u>	<u>0.1089</u>	0.1621	0.1012	0.2123	0.0742	0.0863
BDCMF-1(ours)	**0.1625**	**0.3026**	**0.1174**	**0.1701**	**0.1121**	**0.2341**	**0.0826**	**0.1021**

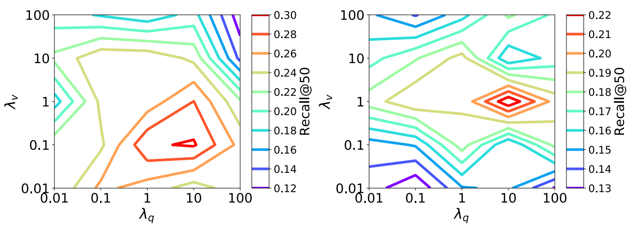
(a) Lastfm—Recall@50 (b) Delicious—Recall@50

Figure 2: The contour of Recall@50 by varying λ_v and λ_q on datasets Lastfm and Delicious.

their friends. λ_q can control how much social information is incorporated into user latent factor. Thus, our BDCMF-1 is very sensitive to λ_v and λ_q. For Delicious dataset, Fig. 2(b) shows BDCMF-1 achieves the best performance on Recall@50 when λ_v =1 and λ_q =10. Fig. 2(a) and 2(b) show that we can balance the content information and social information by varying λ_v and λ_q, leading to better recommendation performance.

To figure out the impact of the latent dimension D, we vary D and see the performance. As shown in Fig. 3, BDCMF-1 outperforms all the baselines with all dimensions, which demonstrates BDCMF-1 can learn better latent representations of users and items than the baselines regardless of their dimensional size.

The Effectiveness of Bayesian posterior estimation (RQ3). To figure out the effectiveness of Bayesian posterior estimation, we make comparison between BDCMF-1 and BDCMF-2 on Lastfm dataset. Results on Delicious show the same trend and thus we omit it here. Performance of BDCMF-1 and BDCMF-2 over iterations on Lastfm is observed as: a) With more iterations, the ELBO of BDCMF-1 and BDCMF-2 gradually increases. BDCMF-2 achieves bigger ELBO than BDCMF-1, which illustrates BDCMF-2 can fit data better (the margin likelihood of data is more higher). b) For recommendation performance, BDCMF-2 outperforms BDCMF-1 in terms of all metrics. Specifically, BDCMF-2 betters with a 10.4% relative improvement on Recall@20. c) When the number of iterations $>$ 30, BDCMF-1 suffers severe overfitting problem (Recall@20 scores begin to decrease, although the EBLO keeps increasing). In contrast, BDCMF-2 does not suffer this overfitting

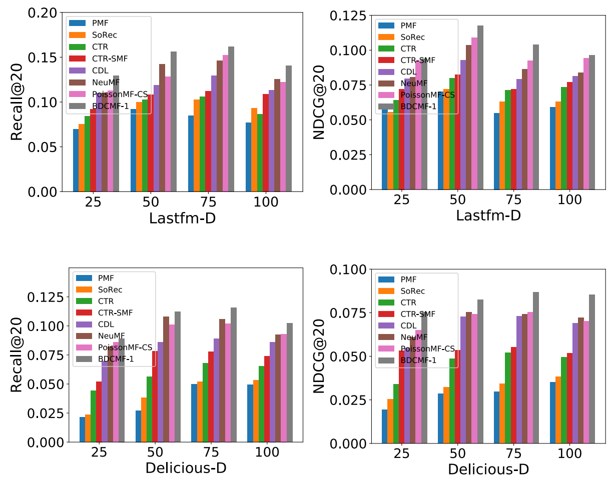
Figure 3: Recommendation performance on Lastfm and Delicious with various values of dimension D.

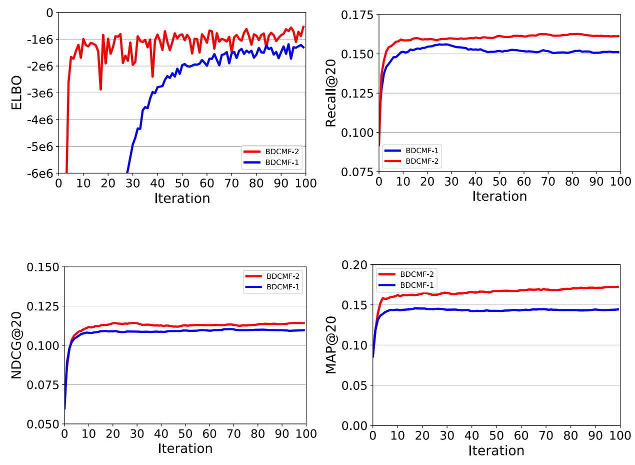
Figure 4: ELBO and recommendation performance of BDCMF methods w.r.t. the number of iterations on Lastfm (D=50, λ_v = 0.1 and λ_q = 10).

problem. These observations demonstrate BDCMF-2 can learn better latent representations than BDCMF-1.

Conclusions

In this paper we studied the problem of inferring latent factors of users and items in the social recommender system. We have proposed a novel Bayesian deep collaborative matrix factorization model, BDCMF, which incorporates various item content information and user social relationship into a full Bayesian framework. To model item's latent content vector, we use a generative network to model the generative process of item content. To effectively infer latent factors of users and items, we first derived a EM algorithm from a Bayesian point estimation perspective. Due to the full Bayesian nature of our proposed model and the drawbacks of Bayesian point estimation, we have inferred the full posterior distribution of the latent factors of users and items. We have conducted several experiments on two public datasets. Experimental results show that our BDCMF can effective infer latent factors of users and items, and the Bayesian posterior estimation is more robust than point estimation. In the future, we will utilize the proposed model to deal with other information retrieval task such as user profiling (Liang 2018; Liang et al. 2018b; Liang, Yilmaz, and Kanoulas 2018) and social network analysis.

References

Adams, R. P.; Dahl, G. E.; and Murray, I. 2014. Incorporating side information in probabilistic matrix factorization with gaussian processes. *Papeles De Poblacion* 3(14):33.

Brusilovsky, P.; Koren, Y.; Kuflik, T.; and Weimer, M. 2010. Workshop on information heterogeneity and fusion in recommender systems. In *RecSys*, 375–376.

Chen, C.; Zheng, X.; Wang, Y.; Hong, F.; and Lin, Z. 2014. Context-aware collaborative topic regression with social matrix factorization for recommender systems. In *AAAI*, 9–15.

Croft, W. B.; Metzler, D.; and Strohman, T. 2015. *Search engines: Information retrieval in practice*. Addison-Wesley Reading.

de Souza da Silva, E.; Langseth, H.; and Ramampiaro, H. 2017. Content-based social recommendation with poisson matrix factorization. In *ECML-PKDD*, 530–546.

Doersch, C. 2016. Tutorial on variational autoencoders. *CoRR* abs/1606.05908.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *WWW*, 173–182.

Kingma, D. P., and Welling, M. 2014. Auto-encoding variational bayes. In *ICLR*.

Li, X., and She, J. 2017. Collaborative variational autoencoder for recommender systems. In *KDD*, 305–314.

Li, S.; Kawale, J.; and Fu, Y. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *CIKM*, 811–820.

Liang, S., and de Rijke, M. 2016. Formal language models for finding groups of experts. *Information Processing & Management* 52(4):529–549.

Liang, S.; Ren, Z.; Weerkamp, W.; Meij, E.; and De Rijke, M. 2014. Time-aware rank aggregation for microblog search. In *CIKM*, 989–998.

Liang, S.; Markov, I.; Ren, Z.; and de Rijke, M. 2018a. Manifold learning for rank aggregation. In *WWW*, 1735–1744.

Liang, S.; Zhang, X.; Ren, Z.; and Kanoulas, E. 2018b. Dynamic embeddings for user profiling in twitter. In *KDD*, 1764–1773.

Liang, S.; Yilmaz, E.; and Kanoulas, E. 2018. Collaboratively tracking interests for user clustering in streams of short texts. *IEEE Transactions on Knowledge and Data Engineering*.

Liang, S. 2018. Dynamic user profiling for streams of short texts. In *AAAI*, 5860–5867.

Lim, Y. J., and Teh, Y. W. 2007. Variational bayesian approach to movie rating prediction. In *Proceedings of KDD cup and workshop*, volume 7, 15–21.

Ma, H.; Yang, H.; Lyu, M. R.; and King, I. 2008. Sorec:social recommendation using probabilistic matrix factorization. In *CIKM*, 931–940.

Mnih, A., and Salakhutdinov, R. R. 2008. Probabilistic matrix factorization. In *NIPS*, 1257–1264.

Neal, R. M., and Hinton, G. E. 1998. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Nato Advanced Study Institute on Learning in Graphical MODELS*, 355–368.

Nguyen, T. T., and Lauw, H. W. 2017. Collaborative topic regression with denoising autoencoder for content and community co-representation. In *CIKM*, 2231–2234.

Park, S.; Kim, Y. D.; and Choi, S. 2013. Hierarchical bayesian matrix factorization with side information. In *IJCAI*, 1593–1599.

Purushotham, S.; Liu, Y.; and Kuo, C. C. J. 2012. Collaborative topic regression with social matrix factorization for recommendation systems. In *ICML*, 691–698.

Ren, Z.; Liang, S.; Li, P.; Wang, S.; and de Rijke, M. 2017. Social collaborative viewpoint regression with explainable recommendations. In *WSDM*, 485–494. ACM.

Wang, C., and Blei, D. M. 2011. Collaborative topic modeling for recommending scientific articles. In *KDD*, 448–456.

Wang, H.; Chen, B.; and Li, W. J. 2013. Collaborative topic regression with social regularization for tag recommendation. In *IJCAI*, 2719–2725.

Wang, H.; Wang, N.; and Yeung, D. Y. 2015. Collaborative deep learning for recommender systems. In *KDD*, 1235–1244.

Welling, M.; Chemudugunta, C.; and Sutter, N. 2008. Deterministic latent variable models and their pitfalls. In *SDM*, 196–207.

Wu, Y.; DuBois, C.; Zheng, A. X.; and Ester, M. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*, 153–162. ACM.

Xu, J.; Yao, Y.; Tong, H.; Tao, X.; and Lu, J. 2017. Hoorays: High-order optimization of rating distance for recommender systems. In *KDD*, 525–534.

Zhang, F.; Yuan, N. J.; Lian, D.; Xie, X.; and Ma, W. Y. 2016. Collaborative knowledge base embedding for recommender systems. In *KDD*, 353–362.