Scalable Variational Bayesian Factorization Machine

Avijit Saha[1], Rishabh Misra[2], Ayan Acharya[3], and Balaraman Ravindran[1]

[1] Department of CSE, Indian Institute of Technology Madras, India
{avijit, ravi}@cse.iitm.ac.in
[2] Department of CSE, Thapar University, India
rishabhmisra1994@gmail.com
[3] Department of ECE, University of Texas at Austin, USA
aacharya@utexas.edu

Abstract. Support Vector Machines (SVMs) are one of the most popular predictors in machine learning which work on the generic features extracted from the data. But SVMs fail on very sparse data as encountered in applications like recommender systems. On the other hand, factorization models like matrix factorization and tensor factorization consider interactions among all the variables and provably work well on sparse and high-dimensional data. However, designing a factorization model specific to a given problem requires expert's knowledge. Factorization machine (FM) is a generic framework which combines the advantages of feature based representation of SVMs and the interaction of variables in factorization models. In this work, a scalable variational Bayesian inference algorithm for FM is developed which converges faster than the existing state-of-the-art Markov-Chain-Monte-Carlo based inference algorithm. Additionally, for large scale learning, a stochastic variational Bayesian algorithm for FM is also introduced which utilizes stochastic gradient descent (SGD) to optimize the lower bound in variational approximation. The variational method based on SGD outperforms existing online algorithm for FM as validated by extensive experiments performed on numerous large-scale real world datasets.

Keywords: Recommender System, Matrix Factorization, Bayesian Inference, Variational Bayes, Online Learning, Scalability.

1 Introduction

SVMs, one of the most popular predictors, are feature based techniques where the data is represented as generic feature vectors. SVMs are solved with standard tools like LIBSVM [1], SVM-Light [2] and do not require expert's intervention to extract information from the data, apart from a prior knowledge of the regularization parameter and the slack variables. However, methods based on SVMs fail in domains with very sparse and high dimensional data, where instead, a class of algorithms called factorization model is widely used due to its prediction quality and scalability. Matrix factorization [3–7] is the simplest and most well studied factorization model. Though factorization models have been successful due to their simplicity, performance and scalability in several domains, deploying these models to new prediction problems is non-trivial and requires – 1) designing of the model and feature representation for the specific application; 2) deriving learning or inference algorithm; and 3) implementing the approach for that specific application – all of which are time consuming and often call for domain expertise.

Factorization Machine (FM) [8] is a generic framework which combines high prediction quality of factorization model with the flexibility of feature engineering. FM represents data as real-valued features like standard machine learning approaches, such as SVMs, and uses interactions between each pair of variables in a low-dimensional latent space. Interestingly, the framework of FM subsumes many successful factorization models like matrix factorization [3], SVD++ [9], TimeSVD++ [10], PITF [11] and FPMC [12]. The other advantages of FM include – 1) FM allows parameter estimation with extremely sparse data where SVMs fail; 2) FM has linear complexity, can be optimized in the primal and, unlike SVMs, does not rely on support vectors; and 3) FM is a general predictor that can work with any real valued feature vector, while other state-of-the-art factorization models work only on very restricted input data.

FM is usually learned using stochastic gradient descent (SGD) [3], an online algorithm in which the training instances are fed sequentially in a random order. FM that uses SGD for learning is conveniently addressed as SGD-FM in this paper. SGD obviates the need to store the entire dataset in memory and hence is often preferred for large scale learning due to memory and speed considerations [13]. Though SGD is scalable and enjoys local convergence guarantee [14], it often overfits the data and requires manual tuning of learning rate and regularization parameters. Alternative methods to solve FM include Bayesian Factorization Machine [15] which provides state-of-the-art performance using Markov-Chain-Monte-Carlo (MCMC) sampling as the inference mechanism and avoids expensive manual tuning of the learning rate and regularization parameters (this framework is addressed as BFM-MCMC from here onwards). However, BFM-MCMC is a batch learning method and is less straight forward to apply to datasets as large as the KDD music dataset [16]. Also, it is difficult to preset the values of burn-in and collection iterations and gauge the convergence of the MCMC inference framework, a known problem with sampling based techniques.

Variational inference, an alternative to MCMC sampling, transforms a complex inference problem into a high-dimensional optimization problem [17–19]. Typically, the optimization is solved using a coordinate ascent algorithm and hence is more scalable compared to MCMC sampling [20]. In this paper, motivated by the scalability of variational methods, a batch **v**ariational **B**ayesian **F**actorization **M**achine (VBFM) is proposed. Empirically, VBFM is found to converge faster than BFM-MCMC and performs as good as BFM-MCMC asymptotically. The convergence is also easy to track when the objective associated with the variational approximation in VBFM stops changing significantly. Additionally, in this paper, an **o**nline **v**ariational **B**ayesian **F**actorization **M**achine

(OVBFM) framework is introduced which uses SGD for maximizing the lower bound obtained from the variational approximation and performs much better than the existing online algorithm for FM that uses SGD. To summarize, the paper makes the following contribution:

1. The VBFM framework is proposed which converges faster than BFM-MCMC.
2. The OVBFM framework is introduced which exploits the advantages of online learning and performs much better than SGD-FM.
3. Extensive experiments on real-world datasets validate the superiority of both VBFM and OVBFM.

The remainder of the paper is structured as follows. Section 2 presents background and related work which is followed by a description of the model in Section 3. In Section 4, the inference mechanism is elaborated for both VBFM and OVBFM. In section 5, the models are evaluated on real world data. Finally, the conclusion and opportunity for future work are presented in section 6.

2 Background and Related Work

In this section, brief descriptions of the basics of factorization models, FM and stochastic variational methods are provided.

2.1 Factorization Model

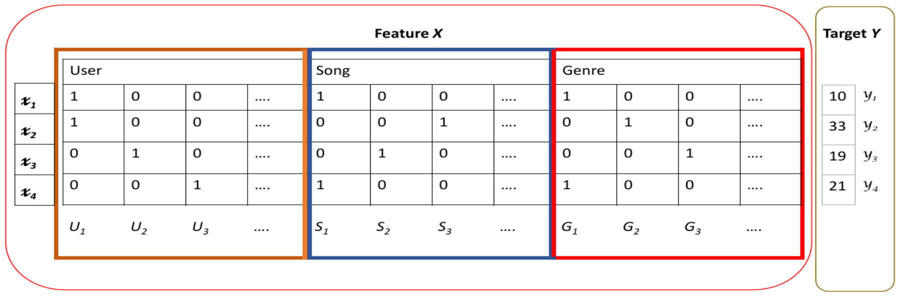
Matrix factorization [3–7] is the simplest and most well studied factorization model and has been applied in solving numerous problems related to analysis of dyadic data, such as in topic modeling [21], recommender systems [3] and network analysis [22]. Tensor factorization [23–26] is an extension of matrix factorization where the data is represented as three dimensional array, signifying interactions of three different variables. Formally, matrix factorization recovers a low-rank latent structure of a matrix by approximating it as a product of two low rank matrices. A given matrix, such as a document-by-word matrix $\boldsymbol{X} \in \mathbb{R}^{N\times D}$, is usually decomposed into two low rank matrices $\boldsymbol{U} \in \mathbb{R}^{N\times K}$ and $\boldsymbol{V} \in \mathbb{R}^{D\times K}$ such that:

$$\boldsymbol{X} \sim \boldsymbol{U}\boldsymbol{V}^{\dagger}. \tag{1}$$

Many specialized factorization models have further been proposed to deal with non-categorical variables. For example, SVD++ uses neighborhood of a user for analysis of user-movie rating data [9], TimeSVD++ [10] and Bayesian Temporal collaborative filtering with Bayesian probabilistic tensor factorization (BPTF) [24] discretize time which is a continuous variable, and Factorizing Personalized Markov Chains for Next-Basket Recommendation (FPMC) [12] considers the purchase history of users to recommend items. Numerous learning techniques have also been proposed for factorization models which include SGD [3], alternating least-squares [27], variational Bayes [28–30] and MCMC based inference [6].

2.2 Factorization Machine (FM)

FM combines the advantages of feature based methods like SVMs with factorization models. Popular feature based methods like SVMs can be solved using standard tools like LIBSVM [1] and SVM-Light [2]. But feature based methods fail in very sparse domains like recommender systems with very sparse and high-dimensional data where the factorization models are very successful. More precisely, FM learns a function $f : \mathbb{R}^D \to T$ which is a mapping from a real valued feature vector $\boldsymbol{X} \in \mathbb{R}^D$ to a target domain T. The training data for FM consists of N tuples, each of which contains the covariate $\boldsymbol{x}_n$ and the response variable y_n. Also FM assumes that the average number of non-zero entries in the feature vectors is much less than D.



	Feature X												Target Y	
	User				Song				Genre					
$\boldsymbol{x}_1$	1	0	0	....	1	0	0	....	1	0	0	....	10	y_1
$\boldsymbol{x}_2$	1	0	0	....	0	0	1	....	0	1	0	....	33	y_2
$\boldsymbol{x}_3$	0	1	0	....	0	1	0	....	0	0	1	....	19	y_3
$\boldsymbol{x}_4$	0	0	1	....	1	0	0	....	1	0	0	....	21	y_4
	U_1	U_2	U_3	....	S_1	S_2	S_3	....	G_1	G_2	G_3	....		

Fig. 1. Feature representation of FM for a song-count dataset with three types of variables: user, song and genre. An example of four data instances are shown. Row 1 shows a data instance where user U_1 listens to song S_1 of genre G_1 10 times.

Example

Consider a song-count dataset where the response variable is the number of times a user has listened to a particular song, which has an associated genre. Let us denote user, song and genre by U, S and G respectively. If the training data is composed of the set of points $\{(U_1, S_1, G_1, 10), (U_1, S_3, G_2, 33), (U_2, S_2, G_3, 19), (U_3, S_1, G_1, 21)\}$, then Fig 1 shows the corresponding feature representation of FM where the i^{th} column indicates the data corresponding to the i^{th} variable. Given the feature representation of Fig 1, the model equation for FM is:

$$y = w_0 + \sum_{i=1}^{D} x_i w_i + \sum_{i=1}^{D} \sum_{j=i+1}^{D} x_i x_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle, \tag{2}$$

where w_0 is the global bias, w_i is the bias associated with i^{th} variable and $\boldsymbol{v}_i$ is the latent factor of dimension K associated with the i^{th} variable. The objective is to estimate the parameters $w_0 \in \mathbb{R}$, $\boldsymbol{w} \in \mathbb{R}^D$ and $\boldsymbol{V} \in \mathbb{R}^{D \times K}$.

FM subsumes many factorization models by feature engineering and has also been used for context aware recommendation [31]. One of the more popular learning algorithms for FM is SGD [8] which requires costly search over the parameter space to find the best values of the learning rate and the regularization parameters. To mitigate such expensive tuning of parameters, learning algorithms based on adaptive stochastic gradient descent and alternating least square [32] have also been proposed which automatically select the learning rate and the regularization parameters. On the other hand, BFM-MCMC [15] is a generative approach whose inference is performed using MCMC sampling for which tuning of parameters is less of a concern, yet it produces state-of-the-art performance for several applications. Scalable algorithm for FM has further been proposed to take advantages of recurring patterns in the data [33]. However, such algorithm is complementary to OVBFM and both can exploit advantages of each other when combined.

2.3 Stochastic Variational Inference Methods

OVBFM is inspired by the successful application of stochastic variational inference (SVI) in matrix factorization [34], topic modeling [19, 35, 36] and network modeling [37]. Specifically, SVI samples a complete data instance, such as a document, and updates all the model parameters. However, OVBFM is different from SVI, as OVBFM considers an individual matrix entry and just updates the parameters associated with that entry, due to the constraints imposed by the generic nature of FM. To reduce the variance during online training, instead of updating the model parameters with every new training instance, a mini-batch version of OVBFM is considered in all the experiments reported here.

3 Model Description

Consider a training set containing N tuples of the form $(\boldsymbol{x}_n, y_n)_{n=1}^N$ where each tuple consists of the covariate $\boldsymbol{x}_n$ and the response variable y_n. The data is further represented in the form of Fig. 1 where the i^{th} column represents the i^{th} variable. The observed data y_n is assumed to be generated as follows:

$$y_n = w_0 + \sum_{i=1}^{D} w_i x_{nj} + \sum_{i=1}^{D} \sum_{j=i+1}^{D} x_{ni} x_{nj} < \boldsymbol{v}_i, \boldsymbol{v}_j > + \epsilon_n, \tag{3}$$

where w_0 is the global bias, w_i is the bias associated with the i^{th} variable and $\boldsymbol{v}_i$ is the latent factor of dimension K associated with the i^{th} variable. The uncertainty in the model is absorbed by the noise ϵ_n which is generated as $\epsilon_n \sim \mathcal{N}(0, \alpha^{-1})$, where α is the precision parameter. Independent prior is imposed on each of the latent variables as follows:

$$p(w_0) = \mathcal{N}(w_0 | 0, \sigma_0^{-1}), \tag{4}$$
$$p(w_i) = \mathcal{N}(w_i | 0, \sigma_{w_{c_i}}^{-1}), \tag{5}$$
$$p(v_{i,k}) = \mathcal{N}(v_{i,k} | 0, \sigma_{v_{c_i,k}}^{-1}). \tag{6}$$

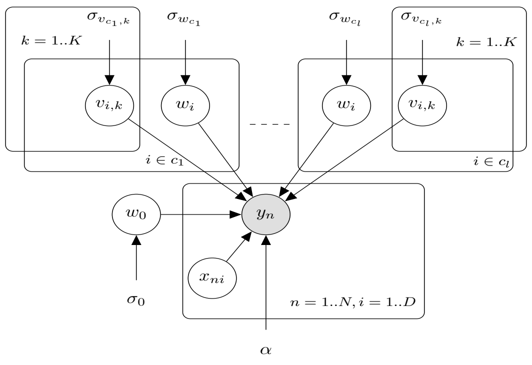


Fig. 2. Graphical model representation of VBFM.

Here c_i denotes the group the i^{th} variable belongs to. For example, in Fig 1, the users, movies and genres form three separate groups. If users form the first group, then according to this notation, $\{U_1, U_2, \cdots\}$ belong to group c_1. Fig. 2 shows a graphical model for VBFM with l different groups.

Let $\boldsymbol{X} \in \mathbb{R}^{N\times D}$ be the sparse feature representation of input data and $\boldsymbol{Y}$ be the vector of corresponding response variables, as shown in Fig 1. For notational convenience, the set of parameters in the model is denoted by $\boldsymbol{\theta} = \{\alpha, \sigma_0, \{\sigma_{w_{c_i}}\}_i, \{\sigma_{v_{c_i,k}}\}_{i,k}\}$ and the set of latent variables is represented concisely by $\boldsymbol{Z} = \{w_0, \{w_i\}_i, \boldsymbol{V}\}$. The joint distribution of the observations and the hidden variables can be written as:

$$p(\boldsymbol{Y}, \boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}) = \mathcal{N}(w_0|0, \sigma_0) \prod_{i=1}^{D} \mathcal{N}(w_i|0, \sigma_{w_{c_i}}^{-1}) \prod_{i=1}^{D}\prod_{k=1}^{K} \mathcal{N}(v_{i,k}|0, \sigma_{v_{c_i,k}}^{-1}) \prod_{n=1}^{N} \mathcal{N}(y_n|\boldsymbol{x}_n, \boldsymbol{Z}, \alpha^{-1}). \quad (7)$$

4 Approximate Inference

4.1 Batch Variational Inference

Given a training set, the objective is to maximize the likelihood of the observations given by $p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta})$. The marginal log-likelihood can be written as:

$$\ln p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta}) = \mathcal{L}(q(\boldsymbol{Z}), \boldsymbol{\theta}) + KL(q||p), \quad (8)$$

where,

$$\mathcal{L}(q(\boldsymbol{Z}), \boldsymbol{\theta}) = \int q(\boldsymbol{Z}) \ln\left(\frac{p(\boldsymbol{Z}, \boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta})}{q(\boldsymbol{Z})}\right) d\boldsymbol{Z}, \quad (9)$$

$$KL(q||p) = -\int q(\boldsymbol{Z}) \ln\left(\frac{p(\boldsymbol{Z}|\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{\theta})}{q(\boldsymbol{Z})}\right) d\boldsymbol{Z}. \quad (10)$$

$KL(q||p)$ is the Kullback-Leibler divergence between $q(\boldsymbol{Z})$ and $p(\boldsymbol{Z}|\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{\theta})$, and is always non-negative. Thus $\ln p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta})$ is lower-bounded by the term

$\mathcal{L}(q(\boldsymbol{Z}), \boldsymbol{\theta})$, also known as **e**vidence **l**ower **bo**und (ELBO) [17, 18]. While maximizing the ELBO $\mathcal{L}(q(\boldsymbol{Z}), \boldsymbol{\theta})$, note that the optimum is achieved at $q(\boldsymbol{Z}) = p(\boldsymbol{Z}|\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{\theta})$. But optimizing $\mathcal{L}(q(\boldsymbol{Z}), \boldsymbol{\theta})$ w.r.t. $q(\boldsymbol{Z})$ is intractable in practice [28]. Therefore, for large scale applications, variational approximation is adopted in which a fully factorized variational distribution is considered:

$$q(\boldsymbol{Z}) = q(w_0) \prod_{i=1}^{D} q(w_i) \prod_{i=1}^{D} \prod_{k=1}^{K} q(v_{i,k}), \tag{11}$$

$$q(w_0) = \mathcal{N}(w_0|\mu_0^{'}, \sigma_0^{'}), \tag{12}$$

$$q(w_i) = \mathcal{N}(w_i|\mu_{w_i}^{'}, \sigma_{w_i}^{'}), \tag{13}$$

$$q(v_{i,k}) = \mathcal{N}(v_{i,k}|\mu_{v_{i,k}}^{'}, \sigma_{v_{i,k}}^{'}). \tag{14}$$

The variational parameters $\{\mu_0^{'}, \sigma_0^{'}, \{\mu_{w_i}^{'}, \sigma_{w_i}^{'}\}_i, \{\mu_{v_{i,k}}^{'}, \sigma_{v_{i,k}}^{'}\}_{i,k}\}$ are conveniently denoted by $\boldsymbol{\phi}$. The ELBO in Eq. (9) can be calculated as follows:

$$\mathcal{L}(q(\boldsymbol{Z}), \boldsymbol{\theta}) = \sum_{n=1}^{N} F_n + F^0 + \sum_{i=1}^{D} F_i^w + \sum_{i=1}^{D} \sum_{k=1}^{K} F_{i,k}^v, \tag{15}$$

$$F_n = E_q \left[\textstyle\sum_{n=1}^{N} \ln \mathcal{N}(y_n|\boldsymbol{x}_n, \boldsymbol{Z}, \alpha^{-1}) \right] = -\tfrac{1}{2} \ln 2\pi\alpha^{-1} - \tfrac{\alpha}{2}((y_n - \bar{y}_n)^2 + T_n), \tag{16}$$

$$\bar{y}_n = \mu_0^{'} + \sum_{i=1}^{D} \mu_{w_i}^{'} x_{ni} + \sum_{i=1}^{D} \sum_{j=i+1}^{D} x_{ni} x_{nj} \sum_{k=1}^{K} \mu_{v_{i,k}}^{'} \mu_{v_{j,k}}^{'}, \tag{17}$$

$$T_n = \sigma_0^{'2} + \sum_{i=1}^{D} \sigma_{w_i}^{'} x_{ni}^2 + \sum_{i=1}^{D} \sum_{j=i+1}^{D} x_{ni}^2 x_{nj}^2 \sum_{k=1}^{K} \left(\mu_{v_{i,k}}^{'2} \sigma_{v_{j,k}}^{'} + \mu_{v_{j,k}}^{'2} \sigma_{v_{i,k}}^{'} + \sigma_{v_{i,k}}^{'} \sigma_{v_{j,k}}^{'} \right), \tag{18}$$

$$F^0 = E_q[\ln \mathcal{N}(w_0|0, \sigma_0^{-1}) - \ln \mathcal{N}(w_0|\mu_0^{'}, \sigma_0^{'})] = \tfrac{1}{2} + \tfrac{1}{2} \ln \sigma_0 \sigma_0^{'} - \tfrac{\sigma_0}{2}(\mu_0^{'2} + \sigma_0^{'}), \tag{19}$$

$$F_i^w = E_q[\ln \mathcal{N}(w_i|0, \sigma_{w_{c_i}}^{-1}) - \ln \mathcal{N}(w_i|\mu_{w_i}^{'}, \sigma_{w_i}^{'})] = \tfrac{1}{2} + \tfrac{1}{2} \ln \sigma_{w_i}^{'} \sigma_{w_{c_i}} - \tfrac{\sigma_{w_{c_i}}}{2}(\mu_{w_i}^{'2} + \sigma_{w_i}^{'}), \tag{20}$$

$$F_{i,k}^v = E_q[\ln \mathcal{N}(v_{i,k}|0, \sigma_{v_{c_i},k}^{-1}) - \ln \mathcal{N}(v_{i,k}|\mu_{v_{i,k}}^{'}, \sigma_{v_{i,k}}^{'})] = \tfrac{1}{2} + \tfrac{1}{2} \ln \sigma_{v_{i,k}}^{'} \sigma_{v_{c_i},k} - \tfrac{\sigma_{v_{c_i},k}}{2}(\mu_{v_{i,k}}^{'2} + \sigma_{v_{i,k}}^{'}). \tag{21}$$

Let $\mathcal{Q}$ be the set of all distributions having a fully factorized form as given in Eq. (11). The optimal distribution that produces the tightest possible lower bound $\mathcal{L}$ is given by:

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q(\boldsymbol{Z})||p(\boldsymbol{Z}|\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{\theta})). \tag{22}$$

The update rule corresponding to a variational parameter describing q^* can be obtained by setting the derivative of Eq. (15) w.r.t. that variational parameter to zero. For example, the update equation for the variational parameters associated with $q^*(v_{i,k})$ can be written as follows:

$$\sigma_{v_{i,k}}^{'} = \left(\sigma_{v_{c_i},k} + \alpha \sum_{n=1}^{N} x_{ni}^2 \left(\left(\sum_{l=1 \& l \neq i}^{D} x_{nl} \mu_{v_{l,k}}^{'} \right)^2 + \sum_{l=1 \& l \neq i}^{D} x_{nl}^2 \sigma_{v_{l,k}}^{'} \right) \right)^{-1}, \tag{23}$$

$$\mu'_{v_{i,k}} = \sigma'_{v_{i,k}} \alpha \sum_{n=1}^{N} x_{ni} \sum_{l=1\&l\neq i}^{D} x_{nl}\mu'_{v_{l,k}} \left(y_n - \bar{y}_n + x_{ni}\mu'_{v_{i,k}} \sum_{l=1\&l\neq i}^{D} x_{nl}\mu'_{v_{l,k}} \right). \quad (24)$$

Straightforward implementation of Eq. (23) and (24) requires $O(KND^2)$ complexity. However, using a simple trick we can reduce the complexity to $O(N)$. To that end, we first show the calculation of $\bar{y}_n$ and T_n:

$$\bar{y}_n = \mu'_0 + \sum_{i=1}^{D} \mu'_{w_i} x_{ni} + \frac{1}{2} \sum_{k=1}^{K} \left(\left(\sum_{i=1}^{D} x_{ni}\mu'_{v_{i,k}} \right)^2 - \sum_{i=1}^{D} x_{ni}^2 \mu'^2_{v_{i,k}} \right), \quad (25)$$

$$T_n = \sigma'^2_0 + \sum_{i=1}^{D} \sigma'_{w_i} x_{ni}^2 + \sum_{k=1}^{K} \left(\sum_{i=1}^{D} x_{ni}^2 \mu'^2_{v_{i,k}} \left(\sum_{i=1}^{D} \sigma'_{v_{i,k}} x_{ni}^2 \right) - \sum_{i=1}^{D} x_{ni}^4 \mu'^2_{v_{i,k}} \sigma'_{v_{i,k}} \right) + \frac{1}{2} \sum_{k=1}^{K} \left(\left(\sum_{i=1}^{D} x_{ni}^2 \sigma_{v_{i,k}} \right)^2 - \sum_{i=1}^{D} x_{ni}^4 \sigma^2_{v_{i,k}} \right). \quad (26)$$

Now, $\bar{y}_n$ and T_n can be computed in $O(KD)$ time. However, the complexity to update Eq. (23) and (24) is still $O(KND)$. We can further reduce the complexity to $O(N)$ by pre-computing the quantities $R_n = (y_n - \bar{y}_n)$ and T_n for all the training points. The update equations, with R_n and T_n included, can be written as follows:

$$\sigma'_{v_{i,k}} = \left(\sigma_{v_{c_i,k}} + \alpha \sum_{n=1}^{N} x_{ni}^2 \left(S_1(i,k)^2 + S_2(i,k) \right) \right)^{-1}, \quad (27)$$

$$\mu'_{v_{i,k}} = \sigma'_{v_{i,k}} \alpha \sum_{n=1}^{N} x_{ni} S_1(i,k) \left(R_n + x_{ni}\mu'_{v_{i,k}} S_1(i,k) \right), \quad (28)$$

where, $S_1(i,k) = \sum_{i=1}^{D} (x_{ni}\mu'_{v_{i,k}} - x_{ni}\mu'_{v_{i,k}})$, $S_2(i,k) = \sum_{i=1}^{D} (x_{ni}^2 \sigma'_{v_{i,k}} - x_{ni}^2 \sigma'_{v_{i,k}})$, and $S_3(i,k) = \sum_{l=1}^{D} (x_{ni}^2 \mu'^2_{v_{i,k}} - x_{ni}^2 \mu'^2_{v_{i,k}})$. Ω_i denotes the set of data instances that have the i^{th} variable. The same trick works for all other parameters as well. Both R_n and T_n can be updated iteratively as each of the hyper-parameters are updated iteratively by setting the derivative of Eq. (15) w.r.t. the corresponding hyper-parameter to zero. The detail procedure is presented in Algorithm 1.

4.2 Stochastic Variational Inference

Stochastic approximation methods follow noisy gradient of a target function with decreasing step sizes. Such noisy gradient is calculated only for the sub-sampled data, the computation of which is often cheaper. Scaling of the objective function is necessary in such case to ensure that the expectation of the noisy

Algorithm 1 **v**ariational **B**ayesian **F**actorization **M**achine (VBFM)

```
Require: $\alpha, \sigma_0, \sigma_{w_{c_i}}, \sigma_{v_{c_{i,k}}}$
Ensure: Randomly Initialize $\sigma_0^{'}$, $\mu_0^{'}$, $\sigma_{w_i}^{'}$, $\mu_{w_i}^{'}$, $\sigma_{v_{i,k}}^{'}$, $\mu_{v_{i,k}}^{'}$
Ensure: Compute $R_n$ and $T_n$ for all the training data points.
 1: for $t = 1$ to I do
 2:     // Update $w_0$'s parameter
 3:     $\sigma_{old} = \sigma_0^{'}$
 4:     $\mu_{old} = \mu_0$
 5:     $\sigma_0^{'} = (\sigma_0 + \alpha N)^{-1}$
 6:     $\mu_0^{'} = \sigma_0^{'}\alpha \sum_{n=1}^{N} \left(R_n + \mu_0^{'}\right)$
 7:     for $n = 1$ to N do
 8:         $R_n = R_n + \mu_{old} - \mu_0^{'}$
 9:         $T_n = R_n + \sigma_0^{'} - \sigma_{old}$
10:     end for
11:     // Update $w_i$'s parameter
12:     for $i = 1$ to D do
13:         $\sigma_{old} = \sigma_{w_i}^{'}$
14:         $\mu_{old} = \mu_{w_i}^{'}$
15:         $\sigma_{w_i}^{'} = \left(\sigma_{w_{c_i}} + \alpha \sum_{n=1}^{N} x_{ni}^2\right)^{-1}$
16:         $\mu_{w_i}^{'} = \sigma_{w_i}^{'}\alpha \sum_{n=1}^{N} x_{ni} \left(R_n + x_{ni}\mu_{w_i}^{'}\right)$
17:         for $n$ in $\Omega_i$ do
18:             $R_n = R_n + x_{ni}(\mu_{old} - \mu_{w_i}^{'})$
19:             $T_n = T_n + x_{ni}(\sigma_{w_i}^{'} - \sigma_{old})$
20:         end for
21:     end for
22:     // Update $v_{i,k}$'s parameter
23:     for $k = 1$ to K do
24:         for $i = 1$ to D do
25:             $\sigma_{old} = \sigma_{v_{i,k}}^{'}$
26:             $\mu_{old} = \mu_{v_{i,k}}^{'}$
27:             $\sigma_{v_{i,k}}^{'} = \left((\sigma_{v_{c_{i,k}}} + \alpha \sum_{n=1}^{N} x_{ni}^2 \left(S_1(i,k)^2 + S_2(i,k)\right)\right)^{-1}$
28:             $\mu_{v_{i,k}}^{'} = \sigma_{v_{i,k}}^{'}\alpha \sum_{n=1}^{N} x_{ni}S_1(i,k) \left[R_n + x_{ni}\mu_{v_{j,f}}S_1(i,k)\right]$
29:             for $n$ in $\Omega_i$ do
30:                 $R_n = R_n + x_{ni}S_1(i,k)(\mu_{old} - \mu_{v_{i,k}}^{'})$
31:                 $T_n = T_n + (x_{ni}^2 S_2(i,k) + x_{ni}^2 S_3(i,k))(\sigma_{v_{i,k}}^{'} - \sigma_{old})$
32:             end for
33:         end for
34:     end for
35:     // Update parameters
36:     $\alpha = \frac{N}{\sum_{n=1}^{N} R_n^2 + T_n}$
37:     $\sigma_0 = \frac{1}{\mu_0^2 + \sigma_0}$
38:     for $i = 1$ to $|c|$ do
39:         $\sigma_{w_{c_i}} = \frac{\sum_{j\in c_i} 1}{\sum_{j\in c_i}\left(\mu_{w_j}^{'2} + \sigma_{w_j}^{'}\right)}$
40:     end for
41:     for $k = 1$ to $K$ do
42:         for $i = 1$ to $|c|$ do
43:             $\sigma_{v_{c_i,k}} = \frac{\sum_{j\in c_i} 1}{\sum_{j\in c_i}\left(\mu_{v_{j,k}}^{'2} + \sigma_{v_{j,k}}^{'}\right)}$
44:         end for
45:     end for
46: end for
```

gradient is equal to the gradient of the original target function. Therefore, in the implementation, after sub-sampling a data instance n uniformly at random from the given dataset, the noisy estimate of $\mathcal{L}(q(\boldsymbol{Z}), \boldsymbol{\theta})$ can be computed as follows:

$$\mathcal{L}_{\text{noisy}}(q(\boldsymbol{Z}), \boldsymbol{\theta}) = {s_n}^{-1}F_n + F^0 + \sum_{i=1;i\in n}^{D} F_i^w + \sum_{i=1;i\in n}^{D}\sum_{k=1}^{K} F_{i,k}^v, \qquad (29)$$

where s_n is the rescaling constant. Eq. (29) is the rescaled version of Eq. (15). Rescale factors for w_0, w_i and $v_{i,k}$ are set to N, $|\Omega_i|$ and $|\Omega_i|$, respectively. The variational parameters associated with $q(\boldsymbol{Z})$ are updated by making a small step in the direction of the gradient of Eq. (29). Since natural gradient leads to faster convergence [38, 19], natural parameters of $q(\boldsymbol{Z})$ are considered for the updates. Such parameters are represented as: $\bar{v}_{i,k} = \frac{\mu'_{v_{i,k}}}{\sigma'_{v_{i,k}}}$, $\hat{v}_{i,k} = \frac{1}{\sigma'_{v_{i,k}}}$ and $\mathring{v}_{i,k} = \{\bar{v}_{i,k}, \hat{v}_{i,k}\}$, with $\mathring{v}_{i,k}$ denoting the natural parameter corresponding to $v_{i,k}$. Also, the natural gradient of $\mathcal{L}_{\text{noisy}}(q(\boldsymbol{Z}), \boldsymbol{\theta})$ w.r.t. $\mathring{v}_{i,k}$ is given by $\nabla\mathcal{L}'(\mathring{v}_{i,k})$. As the model is conditionally conjugate [19], $\nabla\mathcal{L}'(\mathring{v}_{i,k}) = \mathring{v}^*_{i,k} - \mathring{v}_{i,k}$, where $\mathring{v}^*_{i,k} = \{\bar{v}^*_{i,k}, \hat{v}^*_{i,k}\}$ is the value of $\mathring{v}_{i,k}$ that maximizes Eq. (29). Therefore, update equation for $\mathring{v}_{i,k}$ can be written as:

$$\mathring{v}^{\text{new}}_{i,k} = \mathring{v}^{\text{old}}_{i,k} + \eta^i_v(\mathring{v}^*_{i,k} - \mathring{v}^{\text{old}}_{i,k}) = (1 - \eta^i_v)\mathring{v}^{\text{old}}_{i,k} + \eta^i_v \mathring{v}^*_{i,k}, \tag{30}$$

where η^i_v is the step size corresponding to $\mathring{v}_{i,k}$. This step size is updated using Robbins-Monro conditions [19] which ensures convergence (more details can be found in the Appendix). To reduce the variance due to noisy estimate of the gradient, a mini-batch version of online algorithm is considered with a batch size of s number of points. To update a parameter, for example $\mathring{v}_{i,k}$, $\bar{v}^*_{i,k}$ and $\hat{v}^*_{i,k}$ are computed and stored for all the data instances with non-zero feature values in the i^{th} column. The update of $\mathring{v}_{i,k}$ can then be derived as follows:

$$\mathring{v}^{\text{new}}_{i,k} = (1 - \eta^i_v)\mathring{v}^{\text{old}}_{i,k} + \eta^i_v \mathring{v}^{\text{avg}}_{i,k}, \tag{31}$$

where,

$$\mathring{v}^{\text{avg}}_{i,k} = \sum_{n=1}^{n_i} \mathring{v}^{*,n}_{i,k}/n_i. \tag{32}$$

Here n_i is the number of non-zero entries in the i^{th} column of the design matrix constructed from the current batch. Algorithm 2 describes the detail procedure of OVBFM algorithm. Details of the update equations can be found in the Appendix.

5 Experiments

5.1 Dataset

In this section, empirical results on four real-world datasets are presented that validate the effectiveness of the proposed models. Except Netflix, all the other datasets are publicly available. The details of these datasets are provided in Table 1. For Movielens 1m and 10m datasets, the train-test split provided by Movielens is used for the experiments. For Netflix, the probe data is used as the test set. In case of KDD Music dataset, standard train-test split is used for analysis.

5.2 Methods of Comparison

VBFM and OVBFM are compared against BFM-MCMC [15] and SGD-FM [32]. A version of BFM-MCMC is also considered as another baseline which allows 20 burn-in iterations for the sampler and is named as BFM-MCMC-Burnin.

Algorithm 2 **o**nline **v**ariational **B**ayesian **F**actorization **M**achine (OVBFM)

```
Require: $\alpha, \sigma_0, \sigma_{w_{c_i}}, \sigma_{v_{c_i,k}}, \eta^i \ \forall i, k$
Ensure: Randomly Initialize $\sigma_0^{'}$, $\mu_0^{'}$, $\sigma_{w_i}^{'}$, $\mu_{w_i}^{'}$, $\sigma_{v_{i,k}}^{'}$, $\mu_{v_{i,k}}^{'}$ $\forall i, k$
 1: for $t = 1$ to $T$ do
 2:   for $s = 1$ to $M$ do
 3:     Compute $R_n$ and $T_n$ $\forall n \in s$
 4:     // Update $w_0$'s parameter
 5:     for $i = 1$ to $n$ do
 6:       Update $\mathring{w}_0^{\text{avg}}$ using (32)
 7:     end for
 8:     Update $\mathring{w}_0$ using (31)
 9:     Update $R_n$ and $T_n$ like algorithm 1 for current batch
10:     // Update $w_i$'s parameter
11:     for $i = 1$ to $D$ do
12:       for $n = 1$ to $\Omega_i$ do
13:         Update $\mathring{w}_i^{\text{avg}}$ using (32)
14:       end for
15:       Update $\mathring{w}_i$ using (31)
16:       Update $R_n$ and $T_n$ like algorithm 1 for current batch
17:     end for
18:     // Update $v_{i,k}$'s parameter
19:     for $k = 1$ to $K$ do
20:       for $i = 1$ to $D$ do
21:         for $n = 1$ to $\Omega_i$ do
22:           Update $\mathring{v}_{i,k}^{\text{avg}}$ using (32)
23:         end for
24:         Update $\mathring{v}_{i,k}$ using (31)
25:         Update $R_n$ and $T_n$ like algorithm 1 for current batch
26:       end for
27:     end for
28:     Update $\eta_w^0$, $\eta_w^i$, $\eta_v^i$ using Robbins-Monro sequence
29:     // Update hyper-parameters
30:     $\alpha = (1 - \eta_w^0)\alpha + \eta_w^0(\frac{|s|}{\sum_{n=1}^{|s|} R_n^2 + T_n})$
31:     $\sigma_0 = (1 - \eta_w^0)\sigma_0 + \eta_w^0(\frac{1}{\mu_0^2 + \sigma_0})$
32:     for $i = 1$ to $|c|$ do
33:       $\sigma_{w_{c_i}} = (1 - \eta_w^i)\sigma_{w_{c_i}} + \eta_w^i(\frac{\sum_{j \in c_i} 1}{\sum_{j \in c_i} \left(\mu_{w_j}^{\prime 2} + \sigma_{w_j}^{\prime}\right)})$
34:     end for
35:     for $k = 1$ to $K$ do
36:       for $i = 1$ to $|c|$ do
37:         $\sigma_{v_{c_i,k}} = (1 - \eta_v^i)\sigma_{v_{c_i,k}} + \eta_v^i(\frac{\sum_{j \in c_i} 1}{\sum_{j \in c_i} \left(\mu_{v_{j,k}}^{\prime 2} + \sigma_{v_{j,k}}^{\prime}\right)})$
38:       end for
39:     end for
40:   end for
41: end for
```

Table 1. Dataset Description

Dataset	No. of User	No. of Movie	No. of Entries
Movielens 1m	6040	3900	**1m**
Movielens 10m	71567	10681	**10m**
Netflix	480189	17770	**100m**
KDD Music	1000990	624961	**263m**

5.3 Parameter Selection and Experimental Setup

The variational parameters $\mu_0^{'}, \mu_{w_i}^{'}, \mu_{v_{i,k}}^{'}$ are initialized using a standard normal distribution and $\sigma_0^{'}, \sigma_{w_i}^{'}, \sigma_{v_{i,k}}^{'}$ are initialized to 0.02 for both VBFM and OVBFM. The parameters $\boldsymbol{\theta}$ of the model are initialized to 1.0 for both VBFM and OVBFM. Additionally, in OVBFM, all the η's are initialized to 1.0 and decayed further using Robbins-Monro sequence for all the experiments. The number of batches for OVBFM is chosen by cross validation. In BFM-MCMC and

BFM-MCMC-Burnin, the parameters are initialized according to the values suggested in [15]. In SGD-FM, BFM-MCMC and BFM-MCMC-Burnin, parameters in $\boldsymbol{Z}$ are initialized using a normal distribution with zero mean and standard deviation of value 0.01. As performance of SGD-FM is susceptible to the learning rate and regularization parameter, they are chosen by cross-validation. In Movielens 1m, 10m and Netflix, the best performances are achieved with the following values of learning rate and regularization parameter – $(0.001, 0.01)$, $(0.0001, 0.01)$ and $(0.001, 0.01)$ respectively. For KDD music dataset, experiments are run for SGD-FM with three different learning rates 0.0001, 0.00005 and 0.00001, but the regularization parameter is kept fixed at 0.01.

Three sets of experiments are run for each of Movielens 1m, 10m and Netflix datasets corresponding to three different values of $K \in \{20, 50, 100\}$. Since the performances of all the proposed and baseline methods depend on the number of latent factors K, it is necessary to investigate how the models work with a range of values of K. As we run experiments on an Intel I5 machine with 16GB RAM, employing batch algorithms on KDD music data is not possible. Hence, for KDD music dataset, we only compare performances of SGD-FM and OVBFM for $K = 20$ and $K = 50$. All of the proposed and baseline methods are allowed to run for 100 iterations. In case of BFM-MCMC-Burnin, 20 burn-in iterations are followed by 80 collection iterations. Root Mean Square Error (RMSE) [3] is used as the evaluation metric for all the experiments. The code for VBFM and OVBFM will be publicly available upon acceptance of the paper[4].

5.4 Results

Left, middle and right columns in Fig. 3, 4 and 5 correspond to the results for $K = 20$, 50 and 100 respectively. In Fig. 6, left and right columns show results on KDD music dataset with $K = 20$ and 50 respectively. In all the plots, $x-$axis represents the number of iterations and the $y-$axis presents the RMSE value. For all the experiments on Movielens 1m, 10m and Netflix dataset, VBFM is found to converge faster than both BFM-MCMC and BFM-MCMC-Burnin. Though the over-all performance varies for all of these three methods depending on the datasets and the number of latent factors used, the differences among their asymptotic behaviors are negligible. In few cases, VBFM performs slightly better than the other methods and in other cases, the other methods outperform VBFM. For all the experiments on Movielens 1m, 10m and Netflix dataset, OVBFM performs much better than SGD-FM. Also it is evident from the graph that SGD overfits the data quite often. In KDD music dataset, OVBFM performs better than SGD-FM and the gap in RMSE is more significant for $K = 50$. Overfitting with SGD is more problematic with higher values of K in the KDD music dataset. Also for SGD, there is an additional difficulty of tuning the learning rate for each dataset. For very small values of learning rate, SGD underfits the data and for very large values it overfits the data. On the contrary, the performance of OVBFM is quite robust w.r.t. the variance in the learning rate.

[4] https://github.com/avijit1990, https://github.com/rishabhmisra

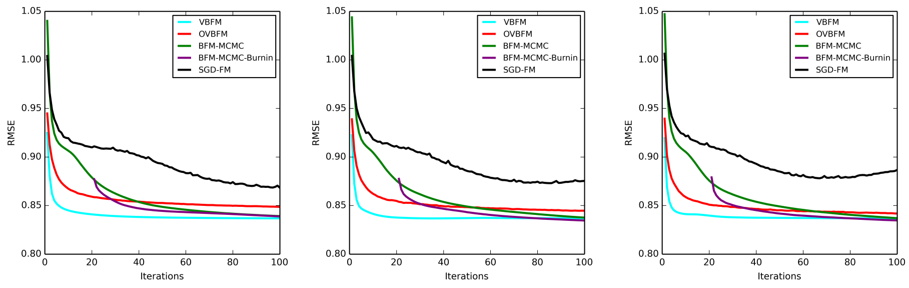


Fig. 3. Results on Movielens 1m dataset

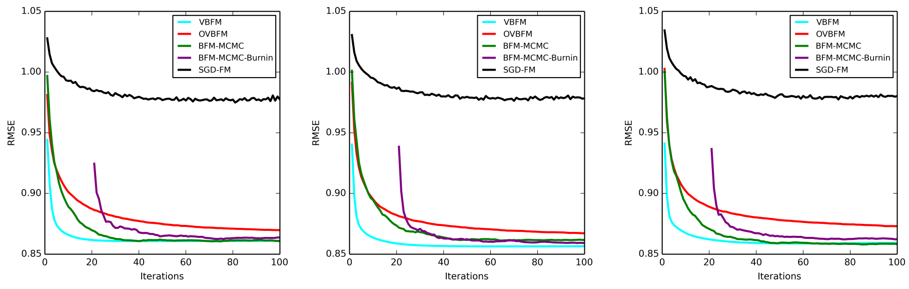


Fig. 4. Results on Movielens 10m dataset

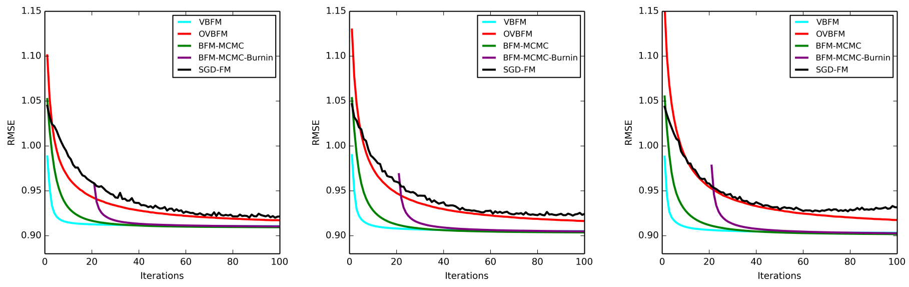


Fig. 5. Results on Netflix dataset

6 Conclusion and Future Work

A variational Bayesian inference algorithm for Factorization Machine (VBFM) is proposed which performs as good as the state-of-the-art MCMC inference algorithm for FM and converges faster than that algorithm. An online variational Bayesian algorithm for FM (OVBFM) is further introduced which combines the advantages of online learning with variational approximation and performs much better than the existing online algorithm for FM based on SGD. Experiments on several real-world datasets show the effectiveness of the proposed approaches.

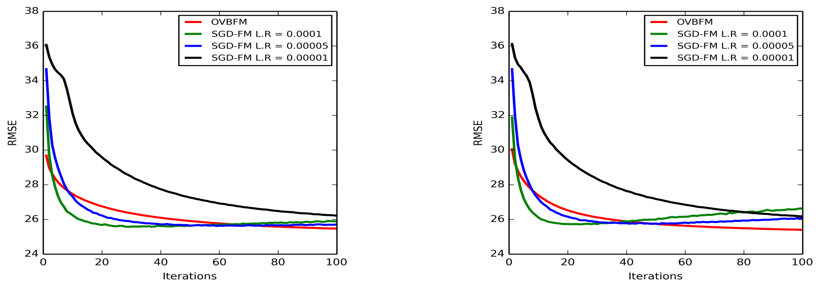


Fig. 6. Results on KDD Music dataset

It would be interesting to extend the proposed methods in applications where one can actively query labels for few data instances [30], leading to solve the cold-start problem. It is hypothesized that this would be a non-trivial problem to solve because of the generic framework that FM entails.

Acknowledgement

We thank Yong-Deok Kim for his useful comments. This project is supported by Ericsson India Global Services Ltd.

References

1. C.-C. Chang and C.-J. Lin, “Libsvm: A library for support vector machines,” *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, May 2011.
2. T. Joachims, “Advances in kernel methods,” ch. Making Large-scale Support Vector Machine Learning Practical, pp. 169–184, Cambridge, MA, USA: MIT Press, 1999.
3. Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, pp. 30–37, Aug. 2009.
4. D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Proc. of NIPS*, 2001.
5. R. Salakhutdinov and A. Mnih, “Probabilistic matrix factorization,” in *Proc. of NIPS*, 2007.
6. R. Salakhutdinov and A. Mnih, “Bayesian probabilistic matrix factorization using markov chain monte carlo,” in *Proc. of ICML*, pp. 880–887, 2008.
7. P. Gopalan, J. M. Hofman, and D. M. Blei, “Scalable recommendation with poisson factorization,” *CoRR*, vol. abs/1311.1704, 2013.
8. S. Rendle, “Factorization machines,” in *Proc. of ICDM*, 2010.
9. Y. Koren, “Factorization meets the neighborhood: A multifaceted collaborative filtering model,” in *Proc. of KDD*, pp. 426–434, 2008.
10. Y. Koren, “Collaborative filtering with temporal dynamics,” in *Proc. of KDD*, pp. 447–456, 2009.
11. S. Rendle and L. Schmidt-Thieme, “Pairwise interaction tensor factorization for personalized tag recommendation,” in *Proc. of WSDM*, pp. 81–90, 2010.

12. S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proc. of WWW*, pp. 811–820, 2010.
13. J. Silva and L. Carin, "Active learning for online Bayesian matrix factorization.," in *Proc. of KDD*, pp. 325–333, ACM, 2012.
14. M.-A. Sato, "Online model selection based on the variational Bayes," *Neural Comput.*, vol. 13, pp. 1649–1681, July 2001.
15. S. R. Christoph Freudenthaler, Lars Schmidt-Thieme, "Bayesian factorization machines," in *Proc. of NIPS Workshop on Sparse Representation and Low-rank Approximation*, 2011.
16. G. Dror, N. Koenigstein, Y. Koren, and M. Weimer, "The yahoo! music dataset and kdd-cup '11.," in *Proc. of KDD Cup*, vol. 18 of *JMLR Proceedings*, pp. 8–18, JMLR.org, 2012.
17. M. Beal, "Variational algorithms for approximate bayesian inference," in *PhD. Thesis, Gatsby Computational Neuroscience Unit, University College London.*, 2003.
18. D. Tzikas, A. Likas, and N. Galatsanos, "The variational approximation for Bayesian inference," *IEEE Signal Processing Magazine*, vol. 25, pp. 131–146, Nov. 2008.
19. M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *J. Mach. Learn. Res.*, vol. 14, pp. 1303–1347, May 2013.
20. M. D. Hoffman, D. M. Blei, and F. R. Bach, "Online learning for latent dirichlet allocation," in *Proc. of NIPS*, pp. 856–864, 2010.
21. S. Arora, R. Ge, and A. Moitra, "Learning topic models – going beyond svd," in *Proc. of FOCS*, pp. 1–10, 2012.
22. M. Zhou, "Infinite edge partition models for overlapping community detection and link prediction," in *Proc. of AISTATS (to appear)*, 2015.
23. A. Shashua and T. Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," in *Proc. of ICML*, pp. 792–799, 2005.
24. L. Xiong, X. Chen, T. K. Huang, J. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with bayesian probabilistic tensor factorization," in *Proc. of SDM*, 2010.
25. J. C. Ho, J. Ghosh, S. R. Steinhubl, W. F. Stewart, J. C. Denny, B. A. Malin, and J. Sun, "Limestone: High-throughput candidate phenotype generation via tensor factorization," *Journal of Biomedical Informatics*, vol. 52, pp. 199–211, 2014.
26. E. C. Chi and T. G. Kolda, "On tensors, sparsity, and nonnegative factorizations.," *SIAM J. Matrix Analysis Applications*, vol. 33, no. 4, pp. 1272–1299, 2012.
27. Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *Proc. 4th Intl Conf. Algorithmic Aspects in Information and Management, LNCS 5034*, 2008.
28. Y. J. Lim and Y. W. Teh, "Variational bayesian approach to movie rating prediction," in *Proc. of KDDCup*, 2007.
29. Y. Kim and S. Choi, "Scalable variational bayesian matrix factorization with side information," in *Proc. of AISTATS*, pp. 493–502, 2014.
30. J. Silva and L. Carin, "Active learning for online bayesian matrix factorization," in *Proc. of KDD*, pp. 325–333, 2012.
31. S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *Proc. of SIGIR*, pp. 635–644, 2011.
32. S. Rendle, "Factorization machines with libFM," *ACM Trans. Intell. Syst. Technol.*, vol. 3, pp. 57:1–57:22, May 2012.

33. S. Rendle, “Scaling factorization machines to relational data,” in *Proc. of VLDB*, pp. 337–348, 2013.
34. J. M. Hernández-Lobato, N. Houlsby, and Z. Ghahramani, “Stochastic inference for scalable probabilistic modeling of binary matrices,” in *Proc. of ICML*, pp. 379–387, 2014.
35. C. Wang and D. M. Blei, “Truncation-free online variational inference for bayesian nonparametric models,” in *Proc. of NIPS*, pp. 413–421, 2012.
36. T. Broderick, N. Boyd, A. Wibisono, A. Wilson, and M. Jordan, “Streaming variational bayes,” in *Proc. of NIPS*, pp. 1727–1735, 2013.
37. P. K. Gopalan, S. Gerrish, M. Freedman, D. M. Blei, and D. M. Mimno, “Scalable inference of overlapping communities,” in *Proc. of NIPS*, pp. 2249–2257, 2012.
38. S. Amari, “Natural gradient works efficiently in learning,” *Neural Comput.*, vol. 10, pp. 251–276, Feb. 1998.

Appendix

Robbins-Monro Update Rules

Learning rates η_w^0, η_w^i and η_v^i are updated each time the corresponding parameters get updated. For this, we have used Robbins-Monro sequence. In particular, let t_{w_0}, t_{w_i} and $t_{v_{i,k}}$ be the number of times corresponding parameters get updated. Then the update rules can be written as follows:

$$\eta_w^0 = (1 + t_{w_0})^{-\lambda}, \eta_w^i = (1 + t_{w_i})^{-\lambda} \ \forall i \in \{1, 2, \cdots, D\}, \tag{33}$$

$$\eta_v^i = (1 + t_{v_{i,k}})^{-\lambda} \ \forall i \in \{1, 2, \cdots, D\} \text{ and } \forall k \in \{1, 2, \cdots, K\}, \tag{34}$$

where $\lambda \in (0.5, 1)$. For all the experiments, minimum value of λ produced best results. Therefore λ is set at 0.5.

Update Rules for OVBFM

– Update rule for the parameters of $\mathring{w}_0$ given the n^{th} data point is as follows:

$$\sigma_{\hat{w}_0} = (1 - \eta_w^0)\sigma_{\hat{w}_0} + \eta_w^0(\sigma_0 + N\alpha), \mu_{\bar{w}_0} = (1 - \eta_w^0)\mu_{\bar{w}_0} + \eta_w^0 \left(N\alpha(R_n + \mu_0^{'})\right). \tag{35}$$

– Update rule for the parameters of $\mathring{w}_i$ given the n^{th} data point is as follows:

$$\sigma_{\hat{w}_i} = (1 - \eta_w^i)\sigma_{\hat{w}_i} + \eta_w^i(\sigma_{w_{c_i}} + |\Omega_i|\alpha x_{ni}^2), \tag{36}$$

$$\mu_{\bar{w}_i} = (1 - \eta_w^i)\mu_{\bar{w}_i} + \eta_w^i \left[|\Omega_i|\alpha x_{ni}\left(R_n - x_{ni}\mu_{w_i}^{'}\right)\right]. \tag{37}$$

– Update rule for the parameters of $\mathring{v}_{i,k}$ given the n^{th} data point is as follows:

$$\sigma_{\hat{v}_{i,k}} = (1 - \eta_v^i)\sigma_{\hat{v}_{i,k}} + \eta_v^i \left[\sigma_{v_{c_i,k}} + |\Omega_i|\alpha x_{ni}^2 \left(S_1(i,k)^2 + S_2(i,k)\right)\right], \tag{38}$$

$$\mu_{\bar{v}_{i,k}} = (1 - \eta_v^i)\mu_{\bar{v}_{i,k}} + \eta_v^i \left[|\Omega_i|\alpha x_{ni} S_1(i,k) \left(R_n + x_{ni}\mu_{v_{i,k}}^{'} S_1(i,k)\right)\right]. \tag{39}$$