Stochastic Inference for Scalable Probabilistic Modeling of Binary Matrices

Jóse Miguel Hernández Lobato*
Department of Engineering
University of Cambridge
jmh233@cam.ac.uk

Neil Houlsby*
Department of Engineering
University of Cambridge
nmth2@cam.ac.uk

Zoubin Ghahramani
Department of Engineering
University of Cambridge
zoubin@eng.cam.ac.uk

Abstract

Fully observed large binary matrices appear in a wide variety of contexts. To model them, probabilistic matrix factorization (PMF) methods are an attractive solution. However, current batch algorithms for PMF can be inefficient since they need to analyze the entire data matrix before producing any parameter updates. We derive an efficient stochastic inference algorithm for PMF models of fully observed binary matrices. Our method exhibits faster convergence rates than more expensive batch approaches and has better predictive performance than scalable alternatives. The proposed method includes new data subsampling strategies which produce large gains over standard uniform subsampling. We also address the task of automatically selecting the size of the minibatches of data and we propose an algorithm that adjusts this hyper-parameter in an online manner.

1 Introduction

Many machine learning methods have been developed for modeling matrices with a large number of missing entries. Matrix factorization (MF) approaches are probably the most successful because of their simplicity and often superior predictive performance. These methods assume that the partially observed data matrix $\mathbf{X}$ is well approximated by a low rank matrix $\mathbf{U}\mathbf{V}^{\mathrm{T}}$. We will focus on probabilistic approaches in which fast approximate inference is usually implemented using variational Bayes (VB) [7, 10, 8]. The resulting techniques are computationally efficient because their cost depends only on the number of entries observed in $\mathbf{X}$, which is usually low, and not on the size of $\mathbf{X}$ which can be large. Many real-world datasets are binary, that is, the entries of $\mathbf{X}$ take values in $\{0, 1\}$. In these cases, $\mathbf{X}$ is fully observed and the aforementioned probabilistic approaches are infeasible in practice because they are based on batch variational algorithms that require processing all the entries in $\mathbf{X}$ before producing even a single update to the variational parameters.

We address the problem of scalable learning with probabilistic MF models that are accurate enough to produce state-of-the-art predictions on large binary matrices. To meet this challenge we propose a novel stochastic inference algorithm. Stochastic methods have the advantage that, with large datasets, they can make reasonably accurate predictions before a batch algorithm has generated a single parameter update. The proposed approach is based on a recently developed method called stochastic variational inference (SVI) [3]. MF models present specific challenges for SVI that are not encountered in models currently addressed by this technique because: i) We subsample individual matrix entries instead of complete data instances (e.g. an entire document in LDA). In standard SVI all the variational parameters are updated each time a data instance is subsampled. With matrices, we have different parameters for each row and column in $\mathbf{X}$ and each time we subsample a matrix entry, we update only the variational parameters associated with the row and column of that entry. This makes the data sub-sampling strategy more important because it determines which parameters

*Both authors contributed equally.

are updated and how often. For this reason, we develop novel data subsampling strategies with different sampling probabilities across the rows and columns of $\mathbf{X}$. ii) Parameter estimates in MF models often exhibit heavy-tailed empirical distributions [6]. These heavy-tails can significantly reduce the convergence speed of stochastic algorithms in practice. A solution is to use minibatches to reduce the effect of outliers in the noisy estimates of the gradients. However, the best minibatch size S can be dataset-dependent. To avoid having to hand-tune S to each dataset, which is common practice, we propose a method that adaptively selects the value of S online.

Our algorithm improves upon the state-of-the-art in probabilistic MF because i) we handle fully observed matrices and learn by subsampling individual matrix entries, ii) we use likelihood functions for binary data and not for continuous data, iii) flexible priors and additional bias parameters can be easily incorporated with our method, and iv) we use improved subsampling strategies and propose a rule to automatically select the minibatch size.

2 A Probabilistic Model for Binary Matrices

We describe a probabilistic model for the generation of an $L \times M$ sparse binary matrix $\mathbf{X}$. We assume that there are two low rank matrices or latent factors $\mathbf{U} \in \mathbb{R}^{L \times D}$ and $\mathbf{V} \in \mathbb{R}^{M \times D}$, where $D \ll \min(L, M)$, such that $\mathbf{X}$ is obtained as a function of $\mathbf{U}$, $\mathbf{V}$ and some additive noise. In particular, $\mathbf{X} = \Theta[\mathbf{U}\mathbf{V}^{\mathrm{T}} + z + \mathbf{E}]$, where $\Theta[\cdot]$ applies the Heaviside step function to the entries of a matrix, $z \in \mathbb{R}$ is a global bias parameter and $\mathbf{E}$ is an $L \times M$ additive noise matrix whose entries e_{ij} are i.i.d. with cumulative distribution function given by the logistic function $\sigma(x) = 1/[1 + \exp(-x)]$. This results in the likelihood

$$p(\mathbf{X}|\mathbf{U}, \mathbf{V}, z) = \prod_{i=1}^{L}\prod_{j=1}^{M} p(x_{i,j}|\mathbf{u}_i, \mathbf{v}_j, z) = \prod_{i=1}^{L}\prod_{j=1}^{M} \left[\sigma(\mathbf{u}_i\mathbf{v}_j^{\mathrm{T}} + z)^{x_{i,j}} \cdot \sigma(-\mathbf{u}_i\mathbf{v}_j^{\mathrm{T}} - z)^{1-x_{i,j}}\right] , \quad (1)$$

where $\mathbf{u}_i$ and $\mathbf{v}_j$ are the i-th and j-th rows of $\mathbf{U}$ and $\mathbf{V}$, respectively. We specify fully factorized Gaussian priors for $\mathbf{U}$, $\mathbf{V}$ and z. We can also incorporate a local bias to each row and column by adding fixed columns of ones to $\mathbf{U}$ and $\mathbf{V}$. The posterior distribution for $\mathbf{U}$, $\mathbf{V}$ and z is

$$p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}) = p(\mathbf{X}|\mathbf{U}, \mathbf{V}, z)p(\mathbf{U})p(\mathbf{V})p(z)/p(\mathbf{X}) . \quad (2)$$

where $p(\mathbf{U}) = \prod_{i=1}^{L}\prod_{d=1}^{D} \mathcal{N}(u_{i,d}|\bar{u}^0_{i,d}, \tilde{u}^0_{i,d})$, $p(\mathbf{V}) = \prod_{j=1}^{M}\prod_{d=1}^{D} \mathcal{N}(v_{j,d}|\bar{v}^0_{j,d}, \tilde{v}^0_{j,d})$, and $p(z) = \mathcal{N}(z|\bar{z}^0, \tilde{z}^0)$. We can make predictions about the expected value $x^\star_{i,j}$ that an entry $x_{i,j}$ in $\mathbf{X}$ would have taken if there were no additive noise $\mathbf{E}$. For this, we use

$$p(x^\star_{i,j}|\mathbf{X}) = \int \left[\sigma(\mathbf{u}_i\mathbf{v}_j^{\mathrm{T}} + z)^{x^\star_{i,j}} \cdot \sigma(-\mathbf{u}_i\mathbf{v}_j^{\mathrm{T}} - z)^{1-x^\star_{i,j}}\right] p(\mathbf{U}, \mathbf{V}, z|\mathbf{X})\, d\mathbf{U}d\mathbf{V}dz . \quad (3)$$

The computation of (2) and (3) is intractable and we have to use approximations. In the following section we show how to use VB [5] for computing approximations to (2) and (3).

2.1 Variational Bayes for Binary Matrices

VB approximates the exact posterior (2) with a simpler distribution $q(\mathbf{U}, \mathbf{V}, z)$ that is tractable. We choose $q(\mathbf{U}, \mathbf{V}, z)$ to be a fully factorized Gaussian:

$$q(\mathbf{U}, \mathbf{V}, z) = \left[\prod_{i=1}^{L}\prod_{d=1}^{D} \mathcal{N}(u_{i,d}|\bar{u}_{i,d}, \tilde{u}_{i,d})\right] \left[\prod_{j=1}^{M}\prod_{d=1}^{D} \mathcal{N}(v_{j,d}|\bar{v}_{j,d}, \tilde{v}_{j,d})\right] \mathcal{N}(z|\bar{z}, \tilde{z}) . \quad (4)$$

We use $\Phi = \{\{\{\bar{u}_{i,d}, \tilde{u}_{i,d},\}_{i=1}^{L}, \{\bar{v}_{j,d}, \tilde{v}_{j,d}\}_{j=1}^{M}\}_{d=1}^{D}, \bar{z}, \tilde{z}\}$ to denote the variational parameters that are adjusted so that $q(\mathbf{U}, \mathbf{V}, z)$ is as similar as possible to $p(\mathbf{U}, \mathbf{V}, z|\mathbf{X})$. This is done maximizing the evidence lower bound (ELBO) $\mathcal{L}(\Phi) = \mathbb{E}_q[\log p(\mathbf{U}, \mathbf{V}, z, \mathbf{X})] - \mathbb{E}_q[\log q(\mathbf{U}, \mathbf{V}, z)]$. However, $\mathbb{E}_q[\log p(\mathbf{U}, \mathbf{V}, z, \mathbf{X})]$ cannot be evaluated analytically. To address this, we use the Gaussian lower bound on the logistic function described in [4]. We choose this approximation because it yields Gaussian complete conditional distributions[1]. Exponential family complete conditionals will allow us to use stochastic inference methods based on natural gradients, which improve convergence [3]. We lower bound $\sigma(a)^{x_{i,j}} \cdot \sigma(-a)^{1-x_{i,j}}$ in (1) with

[1] A complete conditional is the conditional distribution of a variable given all of the other variables.

$\tau(a,\xi) = \exp\{ax_{i,j}\}\sigma(\xi)\exp\{-\frac{a+\xi}{2} + \lambda(\xi)(a^2-\xi^2)\}$, where $\lambda(\xi) = (0.5-\sigma(\xi))/(2\xi)$ and ξ is adjusted to make the lower bound tight at $x = \pm\xi$. When we replace each $p(x_{i,j}|\mathbf{u}_i, \mathbf{v}_j, z)$ in (1) with an instantiation of $\tau(a,\xi)$ that includes its own parameter $\xi_{i,j}$, we obtain a new lower bound

$$\mathcal{L}'(\Phi,\Xi) = \sum_{i=1}^{L}\sum_{j=1}^{M}\alpha_{i,j} + \sum_{i=1}^{L}\sum_{d=1}^{D}\beta_{i,d} + \sum_{j=1}^{M}\sum_{d=1}^{D}\gamma_{j,d} + \kappa\,, \tag{5}$$

where $\Xi = \{\{\{\xi_{i,j}\}_{i=1}^{L}\}_{j=1}^{M}\}$, $\alpha_{i,j} = \log\sigma(\xi_{i,j}) - 0.5[\mu_{i,j}(1-2x_{i,j}) + \xi_{i,j}] + \lambda(\xi_{i,j})(\mu_{i,j}^2 + s_{i,j}^2 - \xi_{i,j}^2)$, $\beta_{i,d} = \rho(\tilde{u}_{i,d}, \tilde{u}_{i,d}^0, \bar{u}_{i,d}, \bar{u}_{i,d}^0)$, $\gamma_{j,d} = \rho(\tilde{v}_{j,d}, \tilde{v}_{j,d}^0, \bar{v}_{j,d}, \bar{v}_{j,d}^0)$, $\kappa = \rho(\tilde{z}, \tilde{z}, \bar{z}^0, \bar{z}^0)$, $\rho(a,b,c,d) = -0.5 - 0.5\log a/b + [(c-d)^2 + a][2b]^{-1}$, $\mu_{i,j} = \sum_d \bar{u}_{i,d}\bar{v}_{j,d}$ and $s_{i,j}^2 = \sum_d \bar{u}_{i,d}^2\tilde{v}_{j,d} + \tilde{u}_{i,d}\bar{v}_{j,d}^2 + \tilde{u}_{i,d}^2\tilde{v}_{j,d}$. One could tune q by the alternative maximization of $\mathcal{L}'$ with respect to Φ and Ξ. Given Φ, Ξ is optimized by setting $\xi_{i,j} = [\mu_{i,j}^2 + s_{i,j}^2]^{0.5}$.

Given Ξ, Φ can be optimized by doing an iteration of gradient descent [10]. This reference describes a state-of-the-art batch method for the optimization of the ELBO in MF models with Gaussian likelihood. The resulting batch algorithm is infeasible when $\mathbf{X}$ is very large since each iteration requires the examination of all the entries in $\mathbf{X}$ before updating any parameters. For massive matrices, we propose a method based on stochastic variational inference (SVI) [3] for optimizing $\mathcal{L}'$.

2.2 SVI for Binary Matrices

Stochastic optimization methods follow noisy estimates of the gradient of a target function which consists of the sum of many terms. Noise arises because the target function is approximated by summing over a reduced set of terms which are randomly subsampled. We apply stochastic optimization to $\mathcal{L}'(\Phi) = \max_\Xi \mathcal{L}'(\Phi,\Xi)$. For this, we iterate over the following process. Firstly, we randomly select indexes $i \in \{1,\ldots,L\}$ and $j \in \{1,\ldots,M\}$ with probability $p(i,j)$. Secondly, we optimize $\xi_{i,j}$ by setting $\xi_{i,j} = [\mu_{i,j}^2 + s_{i,j}^2]^{0.5}$. Thirdly, we compute a noisy estimate of $\mathcal{L}'(\Phi)$:

$$\mathcal{L}'_{\text{noisy}}(\Phi_{i,j}) = [c_{i,j}^{\alpha}]^{-1}\alpha_{i,j} + \sum_{d=1}^{D}\beta_{i,d} + \sum_{d=1}^{D}\gamma_{j,d} + \kappa\,, \tag{6}$$

where $c_{i,j}^{\alpha}$ is a re-scaling constant. Finally, we update $\Phi_{i,j} = \{\{\bar{u}_{i,d}, \tilde{u}_{i,d}, \bar{v}_{j,d}, \tilde{v}_{j,d}\}_{d=1}^{D}, \{\bar{z}, \tilde{z}\}\}$ by making a small step in the direction of the gradient of (6). Intuitively, (6) is an appropriately re-scaled version of (5) that includes only those terms which have the same indexes i and j as the subsampled matrix entry $x_{i,j}$. Importantly, the constant $c_{i,j}^{\alpha}$ is chosen to guarantee that the expectation under the data-sampling strategy $p(i,j)$ of the gradient of (6) with respect to the elements of $\Phi_{i,j}$ is the same as the gradient of $\mathcal{L}'(\Phi)$ with respect to those elements[2].

Instead of standard gradients, one can achieve faster convergence using natural gradients [1]. For this, we work with the natural parameters of (4): $\dot{u}_{i,d} = \bar{u}_{i,d}/\tilde{u}_{i,d}$, $\ddot{u}_{i,d} = 1/\tilde{u}_{i,d}$, and similarly for $\dot{v}_{j,d}$, $\ddot{v}_{j,d}$, $\dot{z}$ and $\ddot{z}$. Let $\mathring{\mathbf{u}}_{i,d} = (\dot{u}_{i,d}, \ddot{u}_{i,d})$ and let $\nabla\mathcal{L}'(\mathring{\mathbf{u}}_{i,d})$ denote the natural gradient of (6) with respect to $\mathring{\mathbf{u}}_{i,d}$. Then $\nabla\mathcal{L}'(\mathring{\mathbf{u}}_{i,d}) = \mathring{\mathbf{u}}_{i,d}^{\star} - \mathring{\mathbf{u}}_{i,d}$, where $\mathring{\mathbf{u}}_{i,d}^{\star} = (\dot{u}_{i,d}^{\star}, \ddot{u}_{i,d}^{\star})$ is the value of $\mathring{\mathbf{u}}_{i,d}$ that maximizes (6) when all the other natural parameters are kept fixed to their current values. Note that $\mathring{\mathbf{u}}_{i,d}^{\star}$ is a noisy estimate of the maximizer of the exact ELBO (5) with respect to $\mathring{\mathbf{u}}_{i,d}$ when all the other natural parameters are kept fixed. The resulting stochastic update for $\mathring{\mathbf{u}}_{i,d}$ is

$$\mathring{\mathbf{u}}_{i,d}^{\text{new}} = \mathring{\mathbf{u}}_{i,d}^{\text{old}} + \rho_i^u\nabla\mathcal{L}'(\mathring{\mathbf{u}}_{i,d}) = (1-\rho_i^u)\mathring{\mathbf{u}}_{i,d}^{\text{old}} + \rho_i^u\mathring{\mathbf{u}}_{i,d}^{\star}, \tag{7}$$

where ρ_i^u is the size of the step taken in the direction of the natural gradient. The corresponding stochastic updates for $\mathring{\mathbf{v}}_{j,d}$ and $\mathring{z}$ are similar.

The resulting Stochastic Inference method for Binary Matrices (SIBM) works by iterating over the following steps: i) randomly subsample an entry $x_{i,j}$ from $\mathbf{X}$ with probability $p(i,j)$ and ii) perform a small update of the variational parameters that approximate the posterior distribution of the i-th row of $\mathbf{U}$, the j-th row of $\mathbf{V}$ and the global bias z. In practice, each time we sample the indices i and j, we first update $\mathring{\mathbf{z}}$, then all the $\mathring{\mathbf{v}}_{j,d}$ and finally all the $\mathring{\mathbf{u}}_{i,d}$. Each of these operations is performed using the updated parameter values produced by the previous operations. Furthermore, we recompute the optimal value for $\xi_{i,j}$ whenever any of the natural parameters change.

[2] This property is required to ensure that the algorithm converges to a minimum of the exact target function.

2.3 The Sampling Distribution

We investigate the performance of different choices of $p(i,j)$, the probability distribution used to subsample the entries of $\mathbf{X}$. Our objective will be to predict the location of those entries in $\mathbf{X}$ that would have taken value one but actually took value zero by effect of the additive noise matrix $\mathbf{E}$. Real-world binary matrices are typically highly sparse. This means that when using a uniform strategy $p(i,j) = 1/(LM)$, most of the sampled entries $x_{i,j}$ take value zero. As a result, SIBM may take many iterations to obtain good predictive performance. We propose two alternative strategies for which we can compute the appropriate rescaling $c_{i,j}^{\alpha}$ in (6).

To ensure that we see enough ones, we propose to sample zeros and ones with equal probability, that is, $p(i,j) = 1/(2\sum_{a=1}^{L}\sum_{b=1}^{M}\mathbf{I}[x_{i,j} = x_{a,b}])$, where $\mathbf{I}[\cdot]$ is the indicator function. Now, each time that an entry is sampled we obtain a zero or a one with equal probability. However, another characteristic of real-world binary matrices is that the frequency of ones and zeros is unbalanced; they can change considerably across rows or columns. In practice, it will take SIBM a long time to accurately model those ones located in rows/columns with many zeros. Any entry sampled in those rows/columns will usually take value zero and sampling a zero there is unlikely to be useful since SIBM can learn quickly that these rows/columns are very sparse. Therefore, we account for this by biasing the class-balanced strategy so that the probability of sampling a one at location (i,j) is proportional to i) the number of zeros found in the i-th row and ii) the number of zeros found in the j-th column. A similar bias is introduced for the zeros. The result is the sampling distribution

$$p(i,j) = [r_i^{(1-x_{i,j})}c_j^{(1-x_{i,j})}][2\sum_{a=1}^{L}\sum_{b=1}^{M}\mathbf{I}[x_{i,j} = x_{a,b}]r_a^{(1-x_{a,b})}c_b^{(1-x_{a,b})}]^{-1}\,, \tag{8}$$

where $r_i^{(0)}$ and $r_i^{(1)}$ are the number of zeros and ones in the i-th row of $\mathbf{X}$ and $c_j^{(0)}$ and $c_j^{(1)}$ count the number of zeros and ones in the j-th column. These counts are lower thresholded at 1 so that $p(i,j) \neq 0$. We find in all of our experiments that the final sampling strategy given in 8 performs best, and so we use this strategy in all experiments presented here.

2.4 Automatic Minibatch Size

Stochastic methods often use minibatches to reduce variance in the noisy estimates of the natural gradient and help the algorithm converge to better local optima. When using a minibatch of size S, we randomly subsample S entries from $\mathbf{X}$. We now store the parameter udpate values that maximize the noisy ELBO (6) ($\mathring{\mathbf{u}}_{i,d}^{\star,s}$ and $\mathring{\mathbf{v}}_{j,d}^{\star,s}$) for each sample $s = 1,\ldots,S$. After subsampling S entries, we update each $\mathring{\mathbf{u}}_{i,d}$ if at least one of the last S subsampled entries belongs to the i-th row of $\mathbf{X}$. The minibatch update rule is the same as in(7), but replacing $\mathbf{u}_{i,d}^{\star}$ with $\mathbf{u}_{i,d}^{\star,\text{avg}} = \frac{1}{n(i)}\sum_{s=1}^{n(i)}\mathring{\mathbf{u}}_{i,d}^{\star,s}$. where $n(i)$ is the number of entries from the i-th row found in the last S subsampled entries.

An important question is how to choose the minibatch size S. This relevant because in matrix factorization models we have that parameter values are often heavy tailed [6]. In our stochastic method, this results in heavy tailed noisy estimates of the natural gradients. The choice of S represents a trade-off between the reduction of these heavy tails and slow convergence due to excessively large minibatches. To avoid having to hand-tune S to each dataset or run expensive cross validation searches to fix this parameter, we propose an adaptive algorithm that selects S appropriately to the statistics of the data during learning. Our approach is to choose S so that we bound the magnitude of the error in the noisy gradient. Let $\mathring{\mathbf{u}}_{i,d}^{\star,\star}$ be the value of $\mathring{\mathbf{u}}_{i,d}$ that maximizes the exact ELBO (5). We obtain a probabilistic bound on the relative error of $\mathring{\mathbf{u}}_{i,d}^{\star,\text{avg}}$ with respect to the global maximizer of the ELBO, $\mathring{\mathbf{u}}_{i,d}^{\star,\star}$ using Markov's inequality:

$$\delta = p\left[\frac{\|\mathring{\mathbf{u}}_{i,d}^{\star,\text{avg}} - \mathring{\mathbf{u}}_{i,d}^{\star,\star}\|_2^2}{\|\mathring{\mathbf{u}}_{i,d}^{\star,\star}\|_2^2} \geq \theta\right] \leq \frac{\mathbb{E}[\|\mathring{\mathbf{u}}_{i,d}^{\star,\text{avg}} - \mathring{\mathbf{u}}_{i,d}^{\star,\star}\|_2^2]}{\theta\|\mathring{\mathbf{u}}_{i,d}^{\star,\star}\|_2^2} = \frac{\|\text{Var}[\mathring{\mathbf{u}}_{i,d}^{\star}]\|_1}{\theta\|\mathbb{E}[\mathring{\mathbf{u}}_{i,d}^{\star}]\|_2^2}\mathbb{E}\left[\frac{1}{n(i)}\right] \approx \frac{\|\text{Var}[\mathring{\mathbf{u}}_{i,d}^{\star}]\|_1}{\theta S p(i)\|\mathbb{E}[\mathring{\mathbf{u}}_{i,d}^{\star}]\|_2^2}\,,$$

where $\text{Var}[\mathring{\mathbf{u}}_{i,d}^{\star}]$ is a vector with the variances of the entries in $\mathring{\mathbf{u}}_{i,d}^{\star}$, $p(i)$ is the probability of sampling an element from the i-th row of $\mathbf{X}$, that is, $p(i) = \sum_j p(i,j)$. We have approximated $\mathbb{E}\left[1/n(i)\right]$ by $1/[p(i)S]$ and we have used the fact that $\mathring{\mathbf{u}}_{i,d}^{\star,\star} = \mathbb{E}[\mathring{\mathbf{u}}_{i,d}^{\star,\text{avg}}]$. We solve for S and obtain a minibatch size that approximately limits the probability that the relative error of $\mathring{\mathbf{u}}_{i,d}^{\star,\text{avg}}$ is larger than θ:

$$S = \|\text{Var}[\mathring{\mathbf{u}}_{i,d}^{\star}]\|_1\left[\theta\delta p(i)\|\mathbb{E}[\mathring{\mathbf{u}}_{i,d}^{\star}]\|_2^2\right]^{-1}\,. \tag{9}$$

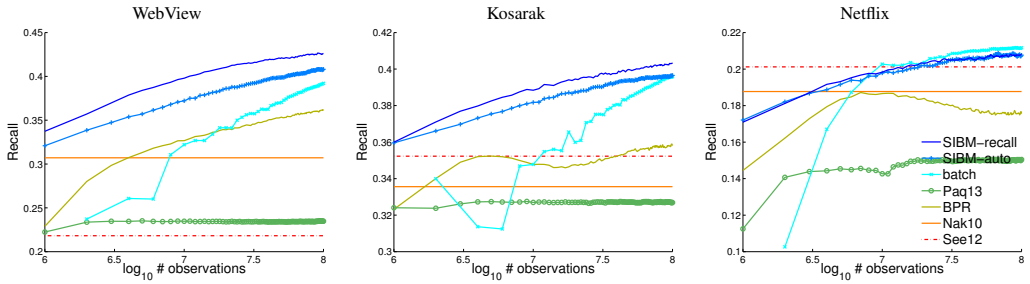

Figure 1: Average recall for each method versus number of samples drawn from $\mathbf{X}$.

Intuitively, the resulting minibatch size increases with the inverse of the signal to noise ratio (SNR) in the estimate $\mathring{\mathbf{u}}^{\star}_{i,d}$ of the global maximizer of the exact ELBO (5), that is, $\mathring{\mathbf{u}}^{\star,\star}_{i,d}$. If the SNR decreases, this rule chooses large minibatches to mitigate the large relative errors.

The proposed approach requires choosing a single dataset-independent parameter, that is, the product of θ and δ, as opposed to hand-tuning S to each dataset. By making $\theta\delta$ small we limit the expected deviation of $\mathring{\mathbf{u}}^{\star,\text{avg}}_{i,d}$ from $\mathring{\mathbf{u}}^{\star,\star}_{i,d}$. In practice $\theta\delta = 2$ leads to good performance. Note that (9) requires knowing $\mathbb{E}[\mathring{\mathbf{u}}^{\star}_{i,d}]$ and $\text{Var}[\mathring{\mathbf{u}}^{\star}_{i,d}]$. We estimate these quantities online using exponentially weighted moving averages. Also, note that rule (9) provides a different minibatch size for each of the $\mathring{\mathbf{u}}_{i,d}$. The rule for each of the $\mathring{\mathbf{v}}_{j,d}$ is similar. In practice, we select S to be the average of the minibatch sizes selected for each of these parameters.

3 Experiments and Discussion

We consider six datasets that include i) a synthetic dataset generated by sampling $\mathbf{X}$ from the generative model assumed by SIBM. ii) Purchase data from a retail store (retail) [2], iii) click data from an online news portal (Kosarak). We include two datasets from the 2000 KDD Cup, iv) point of sale data from a retailer (POS), v) click data from an e-commerce website (WebView), and vi) the Netflix data, treating 4-5 star ratings as ones. We pre-process the original datasets to be able to compare to the computationally expensive batch approach. We keep the 1000 columns with the highest number of ones and discard rows with fewer than 10 ones. We then randomly subsample 2000 rows. Each matrix is randomly split into a training matrix and a set of test entries with value one. The training matrix is generated by randomly removing a one entry from each row in the original matrix and adding this entry to the test set. Predictive performance is evaluated using recall at 10; for this we use (3) to rank the zeros by their probability of actually taking value one in the noise free matrix. We compare the version of SIBM that selects the minibatch size S automatically (SIBM-auto) with a version of SIBM in which S is selected via cross-validation to maximize recall on a validation set (SIBM-recall). We also compare to i) the batch algorithm (batch) that maximizes (5) [10], ii) the analytic solution for variational MF with Gaussian likelihood [8] (Nak10) and iii) the extension of this method to binary matrices [12] (See12), iv) a scheme described in [9] (Paq13) that subsamples the zeros, and finally, v) one of the best performing non-variational Bayesian algorithms, BPR [11].

Figure 1 shows the average recall on 20 realizations of the experiments vs. the number of entries from $\mathbf{X}$ that are observed on three of the datasets. Results for all datasets are shown in Table 1. Other than the analytic solutions (Nak10 and See12), all algorithms have linear cost in the number of observations. It is hard to quantify the number of entries observed by Nak10 and See12, which are based on iterative calls to an SVD subroutine. Therefore, their performance is presented as a constant line[3]. The figures show that that SIBM converges faster than batch and sometimes also to better solutions, such as in the WebView dataset. SIBM-auto produces the greatest improvements during the first iterations of the learning process. These first iterations are the part of the plots that are most relevant for large scale learning. With massive data, only a few passes over the available data instances are possible. It is then when stochastic methods are most useful. In general, the results of SIBM-auto are very close to those of the gold-standard SIBM-recall. The analytic algorithms (Nak10, See12) usually obtain poor predictive performance due to the simplistic modelling assumptions that they make. Paq13 performs poorly because this method subsamples the zeros and

[3]This is a generous assumption for See12, see wall-clock times in Figure 2.

Dataset	SIBM recall	SIBM auto	batch	Paq13	BPR	Nak10	See12
Synthetic	**0.368**	0.360	0.314	0.234	0.321	0.250	0.295
Netflix	0.198	0.198	**0.203**	0.143	0.187	0.188	**0.201**
Kosarak	**0.388**	0.382	0.348	0.327	0.348	0.336	0.352
POS	**0.373**	**0.371**	0.351	0.354	0.345	0.295	0.350
WebView	**0.398**	0.372	0.322	0.235	0.327	0.307	0.218
Retail	**0.234**	0.230	0.229	**0.233**	0.223	0.152	0.228

Table 1: Avg. recall after observing 10^7 samples. Best results in bold, second best underlined.

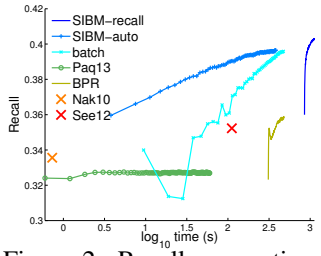

Figure 2: Recall versus time on Kosarak.

does not account for the bias introduced by the subsampling process. As a result, it converges to suboptimal solutions. BPR converges to worse solutions than SIBM and batch. We also plotted recall versus wall-clock time; Figure 2 gives an example on the Kosarak dataset. The results for recall vs. time and recall vs. number of observed entries are similar; the main difference being that SIBM-recall and BPR are heavily penalized due to the additional time required by these methods to run a cross validation search for selecting the minibatch size (SIBM-recall) and regularization parameters (BPR), respectively.

In summary, our stochastic inference method represents an extension of the method stochastic variational inference (SVI) to matrix factorization models, a class of models not addressed before by SVI. The proposed method has the following advantages with respect to existing probabilistic solutions: i) we can handle fully observed matrices, ii) learning occurs by subsampling the matrix entries, iii) we use likelihood functions for binary data instead of for continuous data, iv) flexible priors and additional bias parameters can be easily incorporated in the method.

References

[1] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[2] Tom Brijs, Gilbert Swinnen, Koen Vanhoof, and Geert Wets. Using association rules for product assortment decisions: a case study. In *KDD*, pages 254–260, 1999.

[3] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

[4] T Jaakkola and M Jordan. A variational approach to bayesian logistic regression models and their extensions. In *Sixth International Workshop on Artificial Intelligence and Statistics*, 1997.

[5] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. In *Learning in Graphical Models*, volume 89, pages 105–161. Springer Netherlands, 1998.

[6] Balaji Lakshminarayanan, Guillaume Bouchard, and Cedric Archambeau. Robust bayesian matrix factorisation. In *AISTATS*, pages 425–433, 2011.

[7] Yew Jin Lim and Yee Whye Teh. Variational bayesian approach to movie rating prediction. In *Proceedings of KDD Cup and Workshop*, pages 15–21, 2007.

[8] Shinichi Nakajima, Masashi Sugiyama, and Ryota Tomioka. Global analytic solution for variational bayesian matrix factorization. *NIPS*, 23:1759–1767, 2010.

[9] Ulrich Paquet and Noam Koenigstein. One-class collaborative filtering with random graphs. *WWW*, pages 999–1008, 2013.

[10] T. Raiko, A. Ilin, and K. Juha. Principal component analysis for large scale problems with lots of missing values. In *Machine Learning: ECML 2007*, volume 4701 of *Lecture Notes in Computer Science*, pages 691–698. Springer Berlin / Heidelberg, 2007.

[11] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461. AUAI Press, 2009.

[12] Matthias Seeger and Guillaume Bouchard. Fast variational bayesian inference for non-conjugate matrix factorization models. *JMLR - Proceedings Track*, 22:1012–1018, 2012.