Fast and Flexible Variational Bayes for Sparse Binary Matrices

September 13, 2012

Abstract

1 Introduction

We seek to factorise the binary matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$ whose elements are binary ($x_{ij} \in \{-1, +1\}$) as the product of two lower rank matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{S} \in \mathbb{R}^{K \times J}$, i.e. $\mathbf{X} = \mathbf{AS}$. We have observed only the elements O from $\mathbf{X}$, of which we denote $\mathcal{O}^+$ and $\mathcal{O}^-$ the sets of $+1$s and -1s respectively. These matrices are often sparse, i.e. $\mathcal{O}^+ \ll \mathcal{O}^-$. Probilistic methods for this factorisation are popular because of their ability to handle noise in a formal manner, and deal with missing entries.

However, performing Bayesian inference is intractable so Variational Bayes is used for approximate inference. This may be solved analytically, as in (Nakajima et al., 2010), or by gradient descent, as in (Raiko et al., 2007). The analytic solution has the advantage that it is faster, requiring only $O(\mathcal{O}^+)$ computations, however, it is more restrictive. Additional complexity e.g. non-zero mean priors can be added in a straightforward manner to the gradient descent-based approach, however, the complexity is $O(\mathcal{O})$ i.e. linear in the *total* number of elements, rather than just the $+1$s. Both these approaches use a Gaussian likelihood function, which is somewhat unsatisfactory for a binary matrix, where a classification likelihood would be expected to perform better.

Recent work has extended the analytic solution to arbitrary likelihood functions (Seeger and Bouchard, 2012); however it performs some crude approximations and our experiments do not indicate that it offers a large advantage over the Gaussian likelihood. In this work we extend the gradient descent approach to binary matrices seeking $O(\mathcal{O}^+)$ complexity. We present both a solution for Gaussian and classification likelihoods.

In Section 2 we present the current Gaussian gradient descent approach (Raiko et al., 2007), in Section 3 we extend this to a Sigmoidal likelihood. We reduce the complexity to $O(\mathcal{O}^+)$ in Section 4 and demonstrate the performance of the algorithm on some large matrices in Section 6. Section 7 contains future extension.

2 Gaussian Likelihood

We provide a short review of the work of (Raiko et al., 2007). The model is as follows:

$$\begin{aligned}
P(\mathbf{X}|\mathbf{A},\mathbf{S}) &= \prod_{(i,j)\in O} p(x_{ij}|\mathbf{A},\mathbf{S}) = \prod_{(i,j)\in O} \mathcal{N}(x_{ij}|\sum_{k=1}^{K} a_{ik}s_{kj}) \\
P(\mathbf{A}) &= \prod_{ik} N(a_{ik}|0, v_{a_k}) \\
P(\mathbf{S}) &= \prod_{kj} N(a_{kj}|0, v_{s_k})\,.
\end{aligned}$$

Performing Bayesian inference (computing $P(\mathbf{A}, \mathbf{S}|\mathbf{X})$) is intractable, therefore Variational Bayes is used to approximate the posterior with a simpler distribution. This distribution is a fully factorised Gaussian $Q(\mathbf{A}, \mathbf{S})$:

$$Q(\mathbf{A}, \mathbf{S}) = Q(\mathbf{A})Q(\mathbf{S}) = \left[\prod_{ik} \mathcal{N}(a_{ik}; \bar{a}_{ik}, \tilde{a}_{ik})\right] \left[\prod_{kj} \mathcal{N}(s_{kj}; \bar{s}_{kj}, \tilde{s}_{kj})\right] . \tag{1}$$

The approximate posterior is found by maximising the Variational Free Energy (2).o The Variational Free Energy lower bounds the negative marginal likelihood $P(\mathbf{X})$ for all $Q(\mathbf{A}, \mathbf{S})$. We seek to maximise this cost function following cost with respect to the parameters of the approximation ($\bar{a}_{ik}, \tilde{a}_{ik}, \bar{s}_{kj}, \tilde{s}_{kj}$) and the hyper-parameters (v_x).

$$\mathcal{C} = \mathbb{E}_{Q(\mathbf{A},\mathbf{S})} \left[\log \frac{Q(\mathbf{A}, \mathbf{S})}{P(\mathbf{X}, \mathbf{A}, \mathbf{S})}\right] = \sum_{(i,j)\in O} C_{x_{ij}} + \sum_{ik} C_{a_{ik}} + \sum_{kj} C_{s_{kj}}, \tag{2}$$

The individual terms in (2) take the following form:

$$C_{x_{ij}} = \frac{(x_{ij} - \sum_k \bar{a}_{ik}\bar{s}_{kj})^2 + \sum_k(\tilde{a}_{ik}\bar{s}_{kj}^2 + \bar{a}_{ik}^2\tilde{s}_{kj} + \tilde{a}_{ik}\tilde{s}_{kj})}{2v_x} + \frac{\log 2\pi v_x}{2} \tag{3}$$

$$C_{a_{ik}} = \mathbb{E}_{Q(\mathbf{A})} [\log Q(\mathbf{A}) - \log P(\mathbf{A})] = \frac{\bar{a}_{ik}^2 + \tilde{a}_{ij}}{2v_{a_k}} - \frac{1}{2}\log \frac{\tilde{a}_{ik}}{v_{a_k}} - \frac{1}{2} \tag{4}$$

$$C_{s_{kj}} = \mathbb{E}_{Q(\mathbf{S})} [\log Q(\mathbf{S}) - \log P(\mathbf{S})] = \frac{\bar{s}_{kj}^2 + \tilde{s}_{kj}}{2v_{s_k}} - \frac{1}{2}\log \frac{\tilde{s}_{kj}}{v_{s_k}} - \frac{1}{2} . \tag{5}$$

These are differentiated with respect to all of the parameters for optimisation, for details see (Raiko et al., 2007).

3 Binary Likelihood

For binary matrices it is more appropriate to use a sigmoidal likelihood:

$$P(\mathbf{X}|\mathbf{A}, \mathbf{S}) = \prod_{(i,j)\in O} p(x_{ij}|\mathbf{A}, \mathbf{S}) = \prod_{(i,j)\in O} \sigma\left(x_{ij} \sum_{k=1}^{K} a_{ik}s_{kj}\right) , \tag{6}$$

where

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

.

The same variational approximation is made as for the Gaussian case (1), and the terms in the cost corresponding to the prior ($C_{a_{ik}}$ and $C_{s_{kj}}$) are the same, however, the likelihood term is now intractable.

There are a number of possible approximations available to circumnavigate this intractability (INSERT A REVIEW). With scalability in mind we choose a relatively simple and fast approximation from (Jaakkola and Jordan, 1997). This makes an approximation to the sigmoid, that has a single fitting parameters; the cost now becomes:

$$
\begin{aligned}
C_{x_{ij}} &= \mathbb{E}_{Q(\mathbf{A})Q(\mathbf{S})}\left[-\log P(x_{ij}|\mathbf{A},\mathbf{S})\right] \\
&= -\int \prod_k \mathcal{N}(a_{ik};\bar{a}_{ik},\tilde{a}_{ik})\mathcal{N}(s_{kj};\bar{s}_{kj},\tilde{s}_{kj}) \log\sigma\left(x_{ij}\sum_k a_{ik}s_{kj}\right) d\mathbf{A}d\mathbf{S} \\
&\approx -\int \prod_k \mathcal{N}(a_{ik};\bar{a}_{ik},\tilde{a}_{ik})\mathcal{N}(s_{kj};\bar{s}_{kj},\tilde{s}_{kj}) \log\tau\left(x_{ij}\sum_k a_{ik}s_{kj},\zeta_{ij}\right) d\mathbf{A}d\mathbf{S} \\
&= -\log\sigma(\zeta_{ij}) + \frac{\zeta_{ij}}{2} - \frac{x_{ij}\sum_k \bar{a}_{ik}\bar{s}_{kj}}{2} - \lambda(\zeta_{ij})\left(\sum_k\left(\bar{a}_{ik}^2\tilde{s}_{kj} + \tilde{a}_{ik}\bar{s}_{kj}^2 + \tilde{a}_{ik}\tilde{s}_{kj}\right) + \left(\sum_k \bar{a}_{ik}\bar{s}_{kj}\right)^2 - \zeta_{ij}^2\right).
\end{aligned} \tag{7}
$$

where $\tau(z,\zeta) = \sigma(\zeta)\exp\left\{\frac{z-\zeta}{2} + \lambda(\zeta)\left(z^2-\zeta^2\right)\right\} \leq \sigma(z)$, and $\lambda(\zeta) = \frac{1/2-\sigma(\zeta)}{2\zeta}$. This introduces another parameter ζ_{ij} for every element in $\mathcal{O}$. We maximise over these parameters also to make the approximation as tight as possible. Because τ lower bounds σ, the Free Energy still lower bounds the negative log likelihood.

Because we will be modelling sparse matrices mostly, we introduce another variable b into the model that explicitly captures the sparsity. This is added into the likelihood as follows, and has prior:

$$
\begin{aligned}
P(\mathbf{X}|\mathbf{A},\mathbf{S},b) &= \prod_{(i,j)\in O}\sigma\left(x_{ij}\sum_k a_{ik}s_{kj} + b\right) \\
P(b) &= \mathcal{N}(b;0,v_b)\,.
\end{aligned}
$$

The cost now includes another term C_b, and (7) becomes:

$$
\begin{aligned}
C_b &= \mathbb{E}_{Q(b)}\left[\log Q(b) - \log P(b)\right] = \frac{\bar{b}^2+\tilde{b}}{2v_b} - \frac{1}{2}\log\frac{\tilde{b}}{v_b} - \frac{1}{2} \\
C_{x_{ij}} &= -\log\sigma(\zeta_{ij}) + \frac{\zeta_{ij}}{2} - \frac{x_{ij}\left(\sum_k \bar{a}_{ik}\bar{s}_{kj} + \bar{b}\right)}{2} \\
&\quad - \lambda(\zeta_{ij})\left(\sum_k\left(\bar{a}_{ik}^2\tilde{s}_{kj} + \tilde{a}_{ik}\bar{s}_{kj}^2 + \tilde{a}_{ik}\tilde{s}_{kj}\right) + \left(\sum_k \bar{a}_{ik}\bar{s}_{kj}\right)^2 + 2\bar{b}\sum_k \bar{a}_{ik}\bar{s}_{kj} + (\bar{b}^2+\tilde{b}) - \zeta_{ij}^2\right).
\end{aligned}
$$

3.1 Optimisation

Firstly we need to compute the following derivatives:

$$\frac{\partial \mathcal{C}}{\partial \tilde{a}_{ik}} = \frac{1}{2v_{a_k}} - \frac{1}{2\tilde{a}_{ik}} - \sum_{j|(i,j)\in O} \lambda(\zeta_{ij})(\bar{s}_{kj}^2 + \tilde{s}_{kj})$$

$$\frac{\partial \mathcal{C}}{\partial \tilde{s}_{kj}} = \frac{1}{2v_{s_k}} - \frac{1}{2\tilde{s}_{kj}} - \sum_{i|(i,j)\in O} \lambda(\zeta_{ij})(\bar{a}_{ik}^2 + \tilde{a}_{ik})$$

$$\frac{\partial \mathcal{C}}{\partial \bar{a}_{ik}} = \frac{\bar{a}_{ik}}{v_{a_k}} - \sum_{j|(i,j)\in O} \frac{x_{ij}\bar{s}_{kj}}{2} + 2\lambda(\zeta_{ij}) \left[\bar{a}_{ik}\tilde{s}_{kj} + \bar{s}_{kj}\sum_l \bar{a}_{il}\bar{s}_{lj} + \bar{b}\bar{s}_{kj} \right]$$

$$\frac{\partial \mathcal{C}}{\partial \bar{s}_{kj}} = \frac{\bar{s}_{kj}}{v_{s_k}} - \sum_{i|(i,j)\in O} \frac{x_{ij}\bar{a}_{ik}}{2} + 2\lambda(\zeta_{ij}) \left[\tilde{a}_{ik}\bar{s}_{kj} + \bar{a}_{ik}\sum_l \bar{a}_{il}\bar{s}_{lj} + \bar{b}\bar{a}_{ik} \right]$$

$$\frac{\partial^2 \mathcal{C}}{\partial \bar{a}_{ik}^2} = \frac{1}{v_{a_k}} - \sum_{j|(i,j)\in O} 2\lambda(\zeta_{ij}) \left[\tilde{s}_{kj} + \bar{s}_{kj}^2 \right]$$

$$\frac{\partial^2 \mathcal{C}}{\partial \bar{s}_{kj}^2} = \frac{1}{v_{s_k}} - \sum_{i|(i,j)\in O} 2\lambda(\zeta_{ij}) \left[\tilde{a}_{ik} + \bar{a}_{ik}^2 \right]$$

$$\frac{\partial \mathcal{C}}{\partial \tilde{b}} = \frac{1}{2v_b} - \frac{1}{2\tilde{b}} - \sum_{(i,j)\in O} \lambda(\zeta_{ij})$$

$$\frac{\partial \mathcal{C}}{\partial \bar{b}} = \frac{\bar{b}}{v_b} - \sum_{(i,j)\in O} \frac{x_{ij}}{2} + 2\lambda(\zeta_{ij}) \left[\sum_k \bar{a}_{ik}\bar{s}_{kj} + \bar{b} \right]$$

$$\frac{\partial \mathcal{C}}{\partial v_{a_k}} = \frac{1}{2v_k^a} - \sum_i \frac{\bar{a}_{ik}^2 + \tilde{a}_{ik}}{2(v_{a_k})^2}$$

$$\frac{\partial \mathcal{C}}{\partial v_{s_k}} = \frac{1}{2v_k^s} - \sum_j \frac{\bar{s}_{kj}^2 + \tilde{s}_{kj}}{2(v_{s_k})^2}$$

The variance parameters of $\tilde{a}_{ik}, \tilde{s}_{kj}$ are updates by setting the gradient to zero:

$$\tilde{a}_{ik} \leftarrow \left[\frac{1}{v_{a_k}} - 2\sum_{j|(i,j)\in \mathcal{O}} \lambda(\zeta_{ij})(\bar{s}_{kj}^2 + \tilde{s}_{kj}) \right]^{-1}$$

$$\tilde{s}_{kj} \leftarrow \left[\frac{1}{v_{s_k}} - 2\sum_{i|(i,j)\in \mathcal{O}} \lambda(\zeta_{ij})(\bar{a}_{ik}^2 + \tilde{a}_{ik}) \right]^{-1}$$

The mean parameters cannot be solved for analytically, so gradient descent with a partial Newton update as used, as in (Raiko et al., 2007):

$$\bar{a}_{ik} \leftarrow -\gamma \left(\frac{\partial^2 \mathcal{C}}{\partial \bar{a}_{ik}^2} \right)^{-\alpha} \frac{\partial \mathcal{C}}{\partial \bar{a}_{ik}},$$

and the same for $\bar{s}_{kj}$. γ and α are tunable learning parameters.

The 'sparsity' parameters can both be optimised analytically by setting the gradient to zero:

$$\tilde{b} \leftarrow \left[\frac{1}{v_b} - 2\sum_{(i,j)\in O} \lambda(\zeta_{ij})\right]^{-1}$$
$$\bar{b} \leftarrow \frac{\sum_{(ij)\in O} x_{ij}/2 + 2\lambda(\zeta_{ij})\sum_k \bar{a}_{ik}\bar{s}_{kj}}{1/v_b - 2\sum_{(i,j)\in O}\lambda(\zeta_{ij})}$$

The prior variance parameters can be optimised also:

$$v_{a_k} \leftarrow \frac{1}{I}\sum_i \bar{a}_{ik}^2 + \tilde{a}_{ik}$$
$$v_{s_k} \leftarrow \frac{1}{J}\sum_i \bar{s}_{kj}^2 + \tilde{s}_{kj}$$

Usually v_{a_k} are left at unity due to invariance in the scale between the two matrices.

The additional parameters ζ_{ij} can again be optimised analytically:

$$\zeta_{ij} \leftarrow \sqrt{\sum_k \left(\bar{a}_{ik}^2\tilde{s}_{kj} + \tilde{a}_{ik}\bar{s}_{kj}^2 + \tilde{a}_{ik}\tilde{s}_{kj}\right) + \left(\sum_k \bar{a}_{ik}\bar{s}_{kj}\right)^2 + 2\bar{b}\sum_k \bar{a}_{ik}\bar{s}_{kj} + (\bar{b}^2 + \tilde{b})}\,.$$

One epoch of optimisation updates the parameters in the following order $\{\zeta_{ij}\}$, $\{\tilde{a}_{ik}\}$, $\{\tilde{s}_k j\}$, $\{\bar{a}_{ik}, \bar{s}_{kj}\}$, $\tilde{b}$, $\bar{b}$, $\{v_{a_k}\}$, then $\{v_{s_k}\}$.

4 'Sparse' Implementation

4.1 Gaussian Case

We present a simple trick to reduce first the complexity of computing the Free Energy in the Gaussian case. Computing the priors (4) and (5) takes $O(IK + KJ)$ computations, K may be chosen independently of I, J and so this term grows only linearly in the size of the number of parameters, therefore we assume that this term is negligible compared to the $O(\mathcal{O})$ operations required to compute (3).

The speedup comes from splitting the computation of (3) into terms in $+1$s and -1s:

$$\begin{aligned}
C_x &= \sum_{(i,j)\in O} C_{x_{ij}} \\
&= \sum_{(i,j)\in O^+} \mathbb{E}_{Q(\mathbf{A},\mathbf{S},b)}\left[-\log P(x_{ij}=+1|\mathbf{A},\mathbf{S},b)\right] + \sum_{(i,j)\in O^-} \mathbb{E}_{Q(\mathbf{A},\mathbf{S},b)}\left[-\log P(x_{ij}=-1|\mathbf{A},\mathbf{S},b)\right] \quad (8)\\
&= \sum_{(i,j)\in O^+} \mathbb{E}_{Q(\mathbf{A},\mathbf{S},b)}\left[-\log P(x_{ij}=+1|\mathbf{A},\mathbf{S},b)\right] - \sum_{(i,j)\in O^-} \mathbb{E}_{Q(\mathbf{A},\mathbf{S},b)}\left[-\log P(x_{ij}=-1|\mathbf{A},\mathbf{S},b)\right] \\
&\quad + \sum_{(i,j)\in O} \mathbb{E}_{Q(\mathbf{A},\mathbf{S},b)}\left[-\log P(x_{ij}=-1|\mathbf{A},\mathbf{S},b)\right] \quad (9)\\
& \quad (10)
\end{aligned}$$

The first two terms in (9) require $O(\mathcal{O}^+)$ operations. The third looks like it will take more, however, note that all of the parameters in i and j in (3) either linearly or quadratically. For the linear combinations rearrangements like the following can be performed:

$$\sum_{i,j}\sum_k \bar{a}_{ik}^2\tilde{s}_{kj} = \sum_k\sum_i \bar{a}_{ij}^2 \sum_j \tilde{s}_{kj}\,,$$

which only requires $O(IJ + KJ)$ operations. The quadratic term $\sum_{i,j}(x_{ij} - \sum_k \bar{a}_{ik}\bar{s}_{kj})^2$ can be dealt with similarly with complexity $O(IK^2 + JK^2)$, which is still usually small compared to $O(\mathcal{O}^+)$ as it scales only linearly in the dimensions of the matrix, rather than quadratically. This very simple trick allows the computations to be performed exactly with complexity scaling only with the number of sparse elements in the matrix.

All the updates operation in Section 3.1 are admissible to the same trick, so the whole process has the same complexity as computing the free energy.

4.2 Sigmoidal Case

With the sigmoid, the method needs to be adapted, there are a total of $\mathcal{O}$ fitting parameters, so no rearrangement of sums can avoid the need to look at $\mathcal{O}$ elements. One solution was to fit the parameter to some average value, and then the trick can be applied. However, empirically the performance was found to be not very robust. Therefore it is proposed to work with (8) rather than (9), calculating the term for the $\mathcal{O}^-$ directly.

An approximation is required, we subsample the elements of the matrix. When sums of the form $\sum_{(i,j)\in\mathcal{O}^-}$ are required, e.g. in computing (7), N elements are randomly sampled and the sum computed over these elements and then scaled up by $|\mathcal{O}^-|/N$. When sums over the rows are required ($\sum_{j|(i,j)\in\mathcal{O}}$) e.g. when updating $\bar{a}_{ij}$ then N_r elements are sampled from each row the result is scaled up by J/N_r. Similarly for sums over the columns N_c samples are collected. The complexity becomes $O(N + IN_r + JN_c + \mathcal{O}^+)$, if the number of samples remains small compared to the dimensions of the matrix, this again reduces to $O(\mathcal{O}^+)$

NOTE: currently the samples are taken at the beginning of the experiment and kept fixed, it may be better to re-sample every so often, this is TO BE INVESTIGATED.

5 Analytic Approaches

For the Gaussian case the global maxima can be found analytically (Nakajima et al., 2010). This solution is subject to the restriction that the prior on the rows and columns of A and S respectively are spherical and zero mean. In order to be fast ($O(\mathcal{O}^+)$) this method is built upon a sparse SVD that can only observe the +1s, therefore it treats the -1 elements as zeros, one would expect this to make little difference, except that the effect of the prior is now slightly different than if the -1s are observed. Finally, it is possible to build in an 'offset parameter', b (see (7)) into the gradient descent approach, but this cannot be done with the analytic method.

An algorithm has been developed to adapt the analytic solution to sigmoidal (and other) likelihoods (Seeger and Bouchard, 2012). This approach iteratively uses the Gaussian solution of (Nakajima et al., 2010), then transforms the data based on the result. For the sigmoid case this method is built upon a cruder approximation to the sigmoid function than that used in (7). This method is subject to all the restrictions above for the Gaussian case, including the fact that to be fast it must only look at $\mathcal{O}^+$, therefore it cannot transform the -1 elements (they must always be passed to the Gaussian solver as 0s); therefore, unlike for our algorithm all the elements in $\mathcal{O}^-$ are remain unobserved.

6 Experiments

6.1 Artificial Data

Experiments are conducted on matrices generated from the Sigmoidal model. The Gaussian and Sigmoidal gradient descent approaches with 'sparse complexity' are implemented in C++, because they require multiple loops, and are denoted (GS-cc) and (SS-cc) respectively. The analytic algorithms of (Nakajima et al., 2010) and (Seeger and Bouchard, 2012) are implemented in MATLAB, because the time-limiting step is a sparse

SVD (MATLAB's inbuilt routine is written in FORTRAN, I think), and are denoted (GA-m) and (SA-m) respectively. For comparison, 'dense complexity' implementations of Gaussian and Sigmoid gradient descent schemes are implemented in MATLAB using vector operations (this was found to be slightly faster than using C++ loops), and are denoted (GD-m) and (SD-m). Note that (GD-m) will produce exactly the same result as (GS-cc), but (SD-m) is the 'gold standard' i.e. equivalent to sampling the whole of $\mathcal{O}^-$ (SS-cc). We finally compare to a sparse SVD also (SVD).

Three artificial matrices ($1k \times 1k$, $5k \times 5k$ and $10k \times 10k$) are investigated. These matrices are all around 1% sparse (i.e. $\mathcal{O}^+/\mathcal{O} \approx 0.01$). Performance is evaluated by leaving out $M = 3$ +1s from each row using precision@N, averaged over the rows. Figure 1 presents the results alongside actual run times. The 'dense' implementations of gradient descent can only be run in sensible time (and memory) on the smallest of the datasets.

Firstly, it can be observed that the Sigmoidal models with gradient descent significantly outperform the competitors. Secondly it can be seen that the sparse implementations are significantly faster than their dense counterparts. Finally, despite (on the small matrix), the sampling Sigmoid gradient descent algorithm being outperformed by the full counterpart on the small dataset, it still outperforms all the others on the larger datasets, with comparable running times. A nice feature is that the number of samples can be tuned to trade of performance with computational time.

6.2 Extensions and Real Datasets

Before running experiments on real datasets, performance of the gradient descent algorithms can be improved by adding a couple of 'bells and whistles'. Firstly every row and column can be given its own specific 'offset' parameter, similar to b. This can be added directly into the model by adding an additional column/row of parameters and a column/row of ones to $\mathbf{A}$ and $\mathbf{S}$ respectively.

Secondly, the restriction of a zero-mean spherical prior needn't be maintained, one can assign each parameter a prior of the form $p(a_{ik}) = \mathcal{N}(a_{ik}|m_{a_{ik}}, v_{a_{ik}})$. The parameters of the prior can also be optimised to maximise the free energy also.

The analytic solution does not currently cater to either of these extensions. We apply the first extension to the fast gradient descent algorithms

Results on a 2000×1000 real dataset (Miguel, you sent me this dataset a while back, is it basket data?) are presented in Figure (2). Interestingly, on this dataset the additional row and column-wise offsets assist (SS), but not (GS). Although the running times are a little slower (SS) and (SSe) yield by far the best performance.

NOTE: running time per iteration may be unfair, as typically most iterations are required for the Gaussian gradient descent based approaches, and least for (SA). (SA) requires only one iteration, a single SVD, it is just included for the time being for comparison

7 Future Work

We can now factorise sparse binary matrices with up to $100M$ elements in just a few minutes. However, one can add even more speed by changing the batch optimisation into a continuous, or batch approach (Hoffman et al., 2010). TO IMPLEMENT.

Due to the fact that the priors can be inferred this algorithm could provide a core for a more sophisticated model. Suppose there are many matrices e.g. in a time series, a GP prior could be places over the parameter matrices, and a alternating inference scheme could be employed.

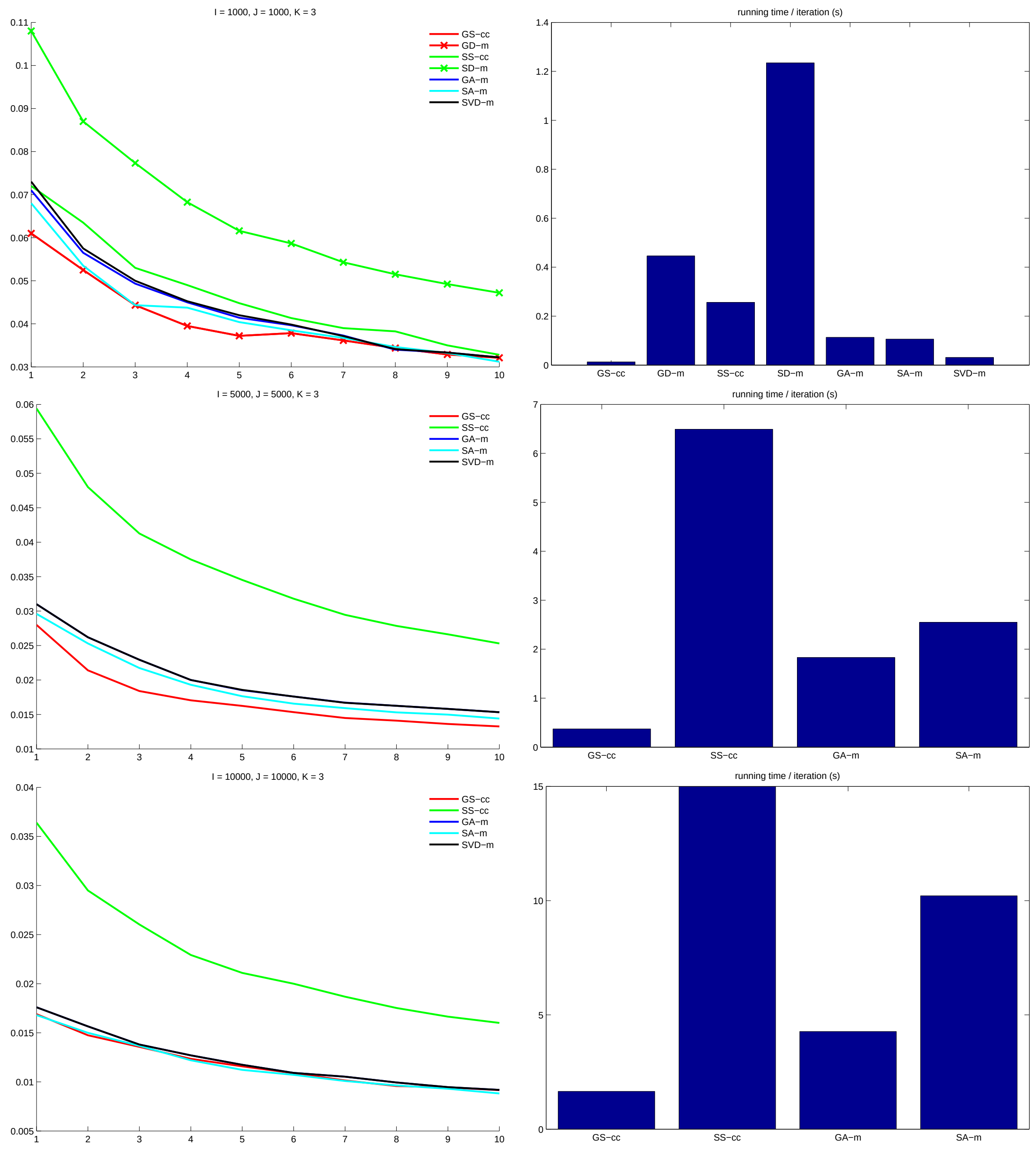

Figure 1: Precisions and running times (per iteration) on artificial matrices.

References

Hoffman, M., Blei, D., and Bach, F. (2010). Online learning for latent dirichlet allocation. *Advances in Neural Information Processing Systems*, 23:856–864.

Jaakkola, T. and Jordan, M. (1997). A variational approach to bayesian logistic regression models and their extensions. In *Proceedings of the sixth international workshop on artificial intelligence and statistics*.

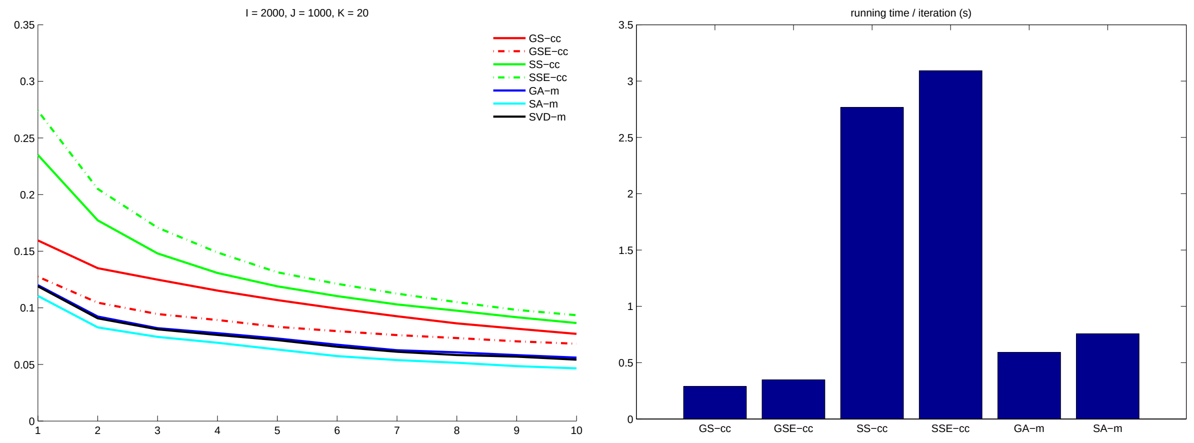

Figure 2: Precisions and running times (per iteration) on real matrix.

Citeseer.

Nakajima, S., Sugiyama, M., and Tomioka, R. (2010). Global analytic solution for variational bayesian matrix factorization. *NIPS2010*.

Raiko, T., Ilin, A., and Juha, K. (2007). Principal component analysis for large scale problems with lots of missing values. In Kok, J., Koronacki, J., Mantaras, R., Matwin, S., Mladenic, D., and Skowron, A., editors, *Machine Learning: ECML 2007*, volume 4701 of *Lecture Notes in Computer Science*, pages 691–698. Springer Berlin / Heidelberg.

Seeger, M. and Bouchard, G. (2012). Fast variational bayesian inference for non-conjugate matrix factorization models.