Supplementary Material for Stochastic Inference for Scalable Probabilistic Modeling of Binary Matrices

José Miguel Hernández-Lobato[1], Neil Houlsby[1] and Zoubin Ghahramani

1 Stochastic Update Rules

The stochastic update rules for $\mathring{\mathbf{u}}_{i,d}$, $\mathring{\mathbf{v}}_{j,d}$ and $\mathring{\mathbf{z}}$ are

$$\begin{aligned}
\mathring{\mathbf{u}}_{i,d}^{\text{new}} &= \mathring{\mathbf{u}}_{i,d}^{\text{old}} + \rho_i^u \nabla \mathcal{L}'(\mathring{\mathbf{u}}_{i,d}) = (1-\rho_i^u)\mathring{\mathbf{u}}_{i,d}^{\text{old}} + \rho_i^u \mathring{\mathbf{u}}_{i,d}^{\star}, \\
\mathring{\mathbf{v}}_{j,d}^{\text{new}} &= \mathring{\mathbf{v}}_{j,d}^{\text{old}} + \rho_j^v \nabla \mathcal{L}'(\mathring{\mathbf{v}}_{j,d}) = (1-\rho_j^v)\mathring{\mathbf{v}}_{j,d}^{\text{old}} + \rho_j^v \mathring{\mathbf{v}}_{j,d}^{\star}, \\
\mathring{\mathbf{z}}^{\text{new}} &= \mathring{\mathbf{z}}^{\text{old}} + \rho^z \nabla \mathcal{L}'(\mathring{\mathbf{z}}) = (1-\rho^z)\mathring{\mathbf{z}}^{\text{old}} + \rho^z \mathring{\mathbf{z}}^{\star},
\end{aligned}$$

where ρ_i^u, ρ_j^v and ρ^z are the sizes of the steps taken in the direction of the natural gradient. Our model has exponential family complete conditionals because of the Gaussian lower bound that we use to approximate the logistic function. A complete conditional is the conditional distribution of a variable given all of the other variables and observations (Hoffman et al., 2013). When the complete conditionals are in the exponential family the natural gradient of Equation (9) in the main document with respect to $\mathring{\mathbf{u}}_{i,d}$ is given by $\nabla \mathcal{L}'(\mathring{\mathbf{u}}_{i,d}) = \mathring{\mathbf{u}}_{i,d}^{\star} - \mathring{\mathbf{u}}_{i,d}$, 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 Equation (9) when all the other natural parameters are kept fixed at their current values. Similarly, the natural gradient of the variational objective with respect to $\mathring{\mathbf{v}}_{j,d}$ is given by $\nabla \mathcal{L}'(\mathring{\mathbf{v}}_{j,d}) = \mathring{\mathbf{v}}_{j,d}^{\star} - \mathring{\mathbf{v}}_{j,d}$, where $\mathring{\mathbf{v}}_{j,d}^{\star} = (\dot{v}_{j,d}^{\star}, \ddot{v}_{j,d}^{\star})$ and $\nabla \mathcal{L}'(\mathring{\mathbf{z}}) = \mathring{\mathbf{z}}^{\star} - \mathring{\mathbf{z}}$, where $\mathring{\mathbf{z}}^{\star} = (\dot{z}^{\star}, \ddot{z}^{\star})$. When we subsample the entry $x_{i,j}$ from $\mathbf{X}$, the values $\dot{u}_{i,d}^{\star}$, $\ddot{u}_{i,d}^{\star}$, $\dot{v}_{i,d}^{\star}$, $\ddot{v}_{i,d}^{\star}$, $\dot{z}^{\star}$, $\ddot{z}^{\star}$ that maximize Equation (9) in the main document are computed as follows.

$$\begin{aligned}
\dot{u}_{i,d}^{\star} =& \bar{u}_{i,d}^0/\tilde{u}_{i,d}^0 + \bar{v}_{j,d}\left[0.5(2x_{i,j}-1) + 2\lambda(\xi_{i,j})(\mu_{i,j} - \bar{u}_{i,d}\bar{v}_{j,d})\right]/p(i|j)\,, \\
\ddot{u}_{i,d}^{\star} =& 1/\tilde{u}_{i,d}^0 - 2\lambda(\xi_{i,j})(\bar{v}_{j,d}^2 + \tilde{v}_{j,d})/p(i|j)\,, \\
\dot{v}_{j,d}^{\star} =& \bar{v}_{j,d}^0/\tilde{v}_{j,d}^0 + \bar{u}_{i,d}\left[0.5(2x_{i,j}-1) + 2\lambda(\xi_{i,j})(\mu_{i,j} - \bar{u}_{i,d}\bar{v}_{j,d})\right]/p(j|i)\,, \\
\ddot{v}_{j,d}^{\star} =& 1/\tilde{v}_{j,d}^0 - 2\lambda(\xi_{i,j})(\bar{u}_{i,d}^2 + \tilde{u}_{i,d})/p(j|i)\,, \\
\dot{z}^{\star} =& \left[0.5(2x_{i,j}-1) + 2\lambda(\xi_{i,j})(\mu_{i,j} - \bar{z})\right]/p(i,j) + \bar{z}^0/\tilde{z}^0\,, \\
\ddot{z}^{\star} =& 1/\tilde{z}^0 - 2\lambda(\xi_{i,j})/p(i,j)\,.
\end{aligned}$$

2 Algorithmic Details

The overall work-flow of the SIBM method is presented in the main paper in Algorithm 1. Here we provide details of the algorithms used to compute the minibatch size automatically and to select the sizes of the steps taken in the direction of the noisy natural gradient (the Robbins-Monro schedule).

[1]Equal contributors.

2.1 Automatic Minibatch Selection

We compute the minibatch size $S^u_{i,d}$ for the variational parameters $\mathring{\mathbf{u}}^\star_{i,d}$ using

$$S^u_{i,d} = \frac{\|\mathrm{Var}[\mathring{\mathbf{u}}^\star_{i,d}]\|_1}{\theta\delta p(i)\|\mathbb{E}[\mathring{\mathbf{u}}^\star_{i,d}]\|_2^2}\,. \tag{1}$$

To compute (1) we need to estimate $\mathbb{E}[\mathring{\mathbf{u}}^\star_{i,d}]$ and $\mathrm{Var}[\mathring{\mathbf{u}}^\star_{i,d}]$. For this, we use exponentially weighted moving averages. Let $\bar{\mathbf{u}}_{i,d}$ and $\bar{\bar{\mathbf{u}}}_{i,d}$ denote respectively our estimates of the mean and mean squared value of $\mathring{\mathbf{u}}^\star_{i,d}$, the noisy maximizer of the lower bound on the ELBO. Each time we draw a sample in the i-th row of $\mathbf{X}$, we update these averages as

$$\begin{aligned}\bar{\mathbf{u}}_{i,d} &= (1-\hat{\rho}^u_i)\bar{\mathbf{u}}_{i,d} + \hat{\rho}^u_i\mathring{\mathbf{u}}^\star_{i,d}\,,\\ \bar{\bar{\mathbf{u}}}_{i,d} &= (1-\hat{\rho}^u_i)\bar{\bar{\mathbf{u}}}_{i,d} + \hat{\rho}^u_i[\mathring{\mathbf{u}}^\star_{i,d}\circ\mathring{\mathbf{u}}^\star_{i,d}]\,,\end{aligned}$$

where "$\circ$" denotes the Hadamard element-wise product. The interpolation weight $\hat{\rho}^u_i$ is selected as $\hat{\rho}^u_i = (1+\hat{t}^i_u)^{-\lambda}$, where $\hat{t}^i_u$ is the number of times that we have sampled an entry in the i-th row of $\mathbf{X}$ and we set $\lambda = 0.7$. The quantities $\mathbb{E}[\mathring{\mathbf{u}}^\star_{i,d}]$ and $\mathrm{Var}[\mathring{\mathbf{u}}^\star_{i,d}]$ are then estimated using $\mathbb{E}[\mathring{\mathbf{u}}^\star_{i,d}] \approx \bar{\mathbf{u}}_{i,d}$ and $\mathrm{Var}[\mathring{\mathbf{u}}^\star_{i,d}] \approx \bar{\bar{\mathbf{u}}}_{i,d} - \bar{\mathbf{u}}_{i,d}\circ\bar{\mathbf{u}}_{i,d}$. The minibatch size $S^v_{j,d}$ for the natural parameters $\mathring{\mathbf{v}}_{j,d}$ is obtained in a similar manner.

Every time we draw S subsamples (where S is the size of the minibatch), we re-update the minibatch size to the average of the sizes selected for the $\mathring{\mathbf{u}}_{i,d}$ and $\mathring{\mathbf{v}}_{j,d}$ parameters:

$$S^{\text{new}} = \frac{\sum_{i=1}^L\sum_{d=1}^D S^u_{i,d} + \sum_{j=1}^M\sum_{d=1}^D S^v_{j,d}}{DL+DM}\,,$$

where $S^u_{i,d}$ is the minibatch size selected for $\mathring{\mathbf{u}}_{i,d}$, as estimated using (1), and $S^v_{j,d}$ is the minibatch size selected for $\mathring{\mathbf{v}}_{j,d}$. Note that S^{new} is computed efficiently by exploiting the fact that $S^u_{i,d}$ and $S^v_{j,d}$ only change if the minibatch includes a sample in the i-th row or j-th column. To collect the initial statistics, we use $S = 5L$ for the first minibatch, subsequent values of S chosen by the algorithm are insensitive to this choice, as evidenced by our experiments.

2.2 Robbins-Monro Step Size Schedule

The step sizes ρ^u_i, ρ^v_j and ρ_z are reduced each time any of the variational parameters $\mathring{\mathbf{u}}_{i,d}$, $\mathring{\mathbf{v}}_{j,d}$ and $\mathring{\mathbf{z}}$ are updated, respectively. For this, we use a simple Robbins-Monro schedule (Robbins and Monro, 1951). In particular, let t^u_i, t^v_j, t^v_j be the number of times that each vector of natural parameters $\mathring{\mathbf{u}}_{i,d}$, $\mathring{\mathbf{v}}_{j,d}$ and $\mathring{\mathbf{z}}$ has been updated, respectively. After each stochastic update, the step sizes can be modified using $\rho^u_i = (1+t^u_i)^{-\lambda}$, $\rho^v_j = (1+t^v_j)^{-\lambda}$ and $\rho^z = (1+t^z)^{-\lambda}$, where $\lambda \in (0.5, 1]$. In our experiments we use $\lambda = 0.7$ since this value produced good overall results.

2.3 Description of MAP-recall

MAP-recall is a version of SIBM-recall that finds the *maximum a posteriori* (MAP) solution of the posterior distribution. This is performed by using standard stochastic gradient ascent with the same data subsampling strategy as in SIBM-recall. Therefore, MAP-recall employs the same re-scaling constants as SIBM ($c^\alpha_{i,j}$ in equation (9) in the main document) to guarantee that the expectation of the noisy gradient is the exact gradient of the posterior distribution. We performed two modifications to SIBM-recall to obtain MAP-recall. Firstly, we no longer use the variational parameters related to the variance of the posterior approximation since we only seek a point estimate of the model parameters. Secondly, MAP-recall uses standard gradients and not natural gradients. Natural gradients are not available when doing MAP inference because we are no longer minimizing the Kullback-Leibler divergence between probability distributions.

In MAP-recall we select the sizes of the steps taken in the direction of the noisy natural gradient using a Robbins-Monro schedule similar to the one used in SIBM-recall. In particular, let t^u_i, t^v_j, t^v_j be the number of

times that the parameters in the i-th row of $\mathbf{U}$, the j-th row of $\mathbf{V}$ and the bias parameter z have been updated, respectively. The step sizes are modified using $\rho_i^u = (t_0^u + t_i^u)^{-\lambda}$, $\rho_j^v = (t_0^v + t_j^v)^{-\lambda}$ and $\rho^z = (t_0^z + t^z)^{-\lambda}$. We fixed $\lambda = 1$, which worked better than $\lambda = 0.7$ (the one used in SIBM). We also hand-tuned t_0^z, t_0^y and t_0^v to yield the best possible overall results. In our experiments, MAP-recall was more sensitive to the values of t_0^z, t_0^y and t_0^v than SIBM, for which $t_0^z = t_0^y = t_0^v = 1$ works well.

3 Additional Results

This section contains all of the plots for the experiments detailed in the main text. These include plots of performance (measured using recall at 10) versus number of samples and versus wall-clock time. We also include plots showing the evolution of the lower bound on the ELBO, Equation (7) in the main document. The results of these plots are summarized in Tables 1 and 2 for the small and large datasets, respectively. The numbers in these tables correspond to results taken at a 'slice' just after observing a fixed number of samples. We take a slice rather than presenting performance after convergence of the algorithms because we are interested in scalable methods that produce good solutions on large datasets with a finite computational budget. Running the algorithms to convergence on massive matrices can take an infeasible amount of time. The best performing method (and those statistically indistinguishable) on each dataset is highlighted in bold, the second best is underlined. These tables reveal that SIBM-recall consistently performs best in terms of recall, with SIBM-auto coming very close. The method MAP-recall is usually outperformed by the other variational approaches SIBM-auto and SIBM-recall. When evaluated using cost, SIBM-auto performs best.

3.1 Number of Samples and Running Time

We present the performance of each algorithm versus the number of entries observed in the data matrix and wall-clock time. Each algorithm has a linear computational cost in the number of observed entries, except the algorithms based on the analytic solution with a Gaussian likelihood (Nak10 and See12). It is hard to quantify the number of samples observed by these solutions, so in Figures 1 and 3 these solutions are depicted using a horizontal line at their performance at convergence. In Figures 2 and 4 they are presented as a single point at the time taken to produce their final solution. It is important to note that these time plots are highly implementation-dependent. To be as fair as possible we implemented all algorithms in C.

Figures 1 and 2 show the learning curves on the small datasets against number of samples and time on the horizontal axis, respectively. Figure 1 is a replication of the plots presented in the main text. Figures 3 and 4 show the corresponding plots for the large datasets. Overall, with respect to the number of samples, SIBM-recall and SIBM-auto perform best. MAP-recall is usually outperformed by the other variational approaches SIBM-auto and SIBM-recall. Furthermore, the performance of MAP-recall deteriorates in some cases as the method converges to the MAP solution. This is indicative of overfitting problems. With respect to wall-clock time, SIBM-auto produces good solutions quickly, greatly outperforming the batch solution in the early stages of learning. In Figures 2 and 4 SIBM-recall, MAP-recall and BPR are heavily penalized because they require running a cross-validation search to select the minibatch size and the regularization parameters, respectively.

The time plots also reveal that although Nak10 is fast, See12 does not run very quickly. In almost all cases SIBM-auto produces better solutions before the algorithm See12 has converged. Finally, Paq13 runs quickly, but the solutions usually have poor predictive performance.

3.2 Evolution of the Lower Bound on the ELBO

Figures 5 and 6 show the evolution of the cost (negative ELBO) on the small datasets. We do not report the value of the lower bound on the ELBO on the large datasets because it is too expensive to compute. We only report the lower bound value for the stochastic methods and See12 because the other methods BPR, Paq13 and Nak10 do not yield comparable values. After observing many samples, the batch algorithm will converge to an optimum of the lower bound. However, early in learning (as measured both by number of

Table 1: Small datasets, recall and cost after observing 10^7 samples. Bold typeface indicates the best results (and those statistically indistinguishable), underlining denotes the second best.

	recall								**cost**$\times 10^{-5}$			
Dataset	**SIBM recall**	**SIBM auto**	**batch**	**MAP recall**	**Paq13**	**BPR**	**Nak10**	**See12**	**SIBM recall**	**SIBM auto**	**batch**	**See12**
Synthetic	**0.368**	0.360	0.314	0.347	0.234	0.321	0.250	0.295	**1.804**	**1.803**	1.821	4.313
Netflix	0.198	0.198	**0.203**	0.189	0.143	0.187	0.188	**0.201**	4.555	4.550	**4.383**	6.807
Kosarak	**0.388**	0.382	0.348	0.348	0.327	0.348	0.336	0.352	2.124	**1.963**	1.994	3.607
POS	**0.373**	**0.371**	0.351	0.353	0.354	0.345	0.295	0.350	**1.413**	**1.415**	1.437	2.674
WebView	**0.398**	0.372	0.322	0.374	0.235	0.327	0.307	0.218	1.672	**1.573**	1.630	2.886
Retail	0.234	0.230	0.229	**0.237**	0.233	0.223	0.152	0.228	1.557	**1.490**	1.511	2.430

Table 2: Large datasets, recall after observing 10^7 samples from WebView, Retail and 10^8 from others.

Dataset	**SIBM recall**	**SIBM auto**	**batch**	**MAP recall**	**Paq13**	**BPR**	**Nak10**	**See12**
Synthetic	**0.387**	0.367	0.324	0.368	0.249	0.374	0.262	0.266
Netflix	**0.203**	0.193	0.190	0.192	0.146	0.190	0.190	0.199
Kosarak	**0.391**	0.372	0.346	0.368	0.327	0.370	0.319	0.341
POS	0.373	0.368	0.348	0.352	0.352	**0.374**	0.289	0.347
WebView	**0.390**	0.343	0.359	0.360	0.235	0.326	0.303	0.213
Retail	**0.235**	0.230	0.233	**0.239**	0.235	**0.237**	0.149	0.228

samples and time) the stochastic algorithm achieves much better values. In most cases SIBM achieves a reasonably good solution before batch has completed a single iteration.

3.3 Sampling Strategies

Figure 7 depicts the performance of the three sampling strategies on the small datasets. On all of the datasets S-Biased performs best, followed by S-Balanced. As expected, on sparse binary matrix data, uniform sampling (S-Uniform) yields slow convergence.

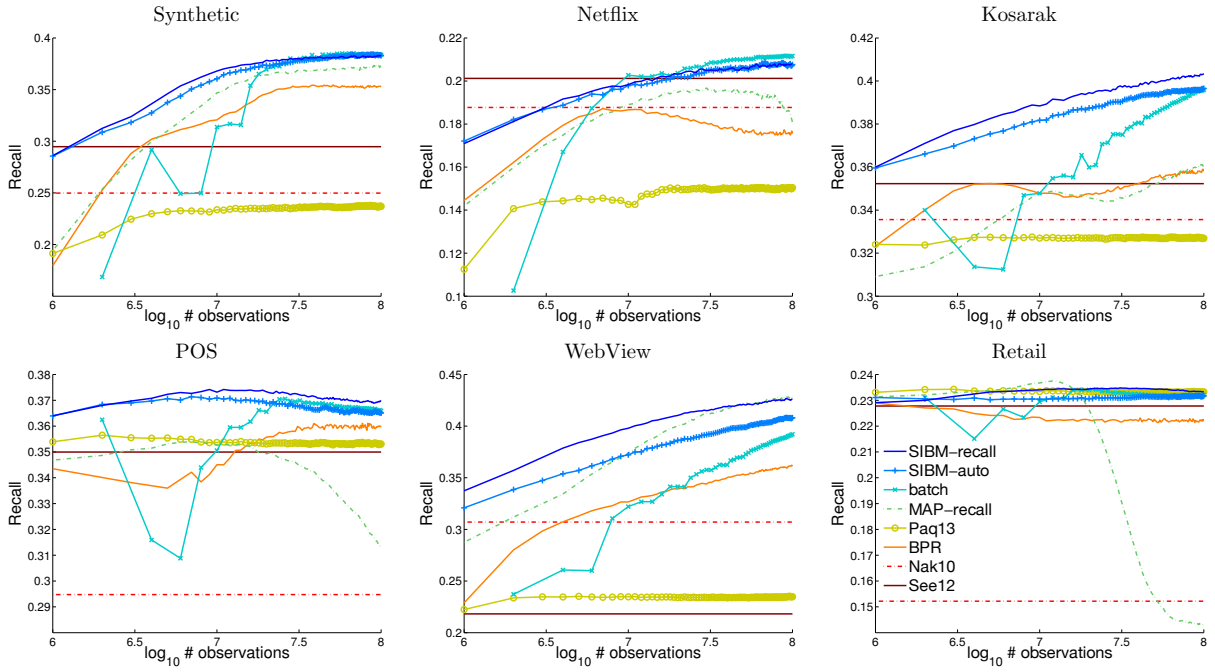

Figure 1: Average recall versus number of samples for each method on each small dataset.

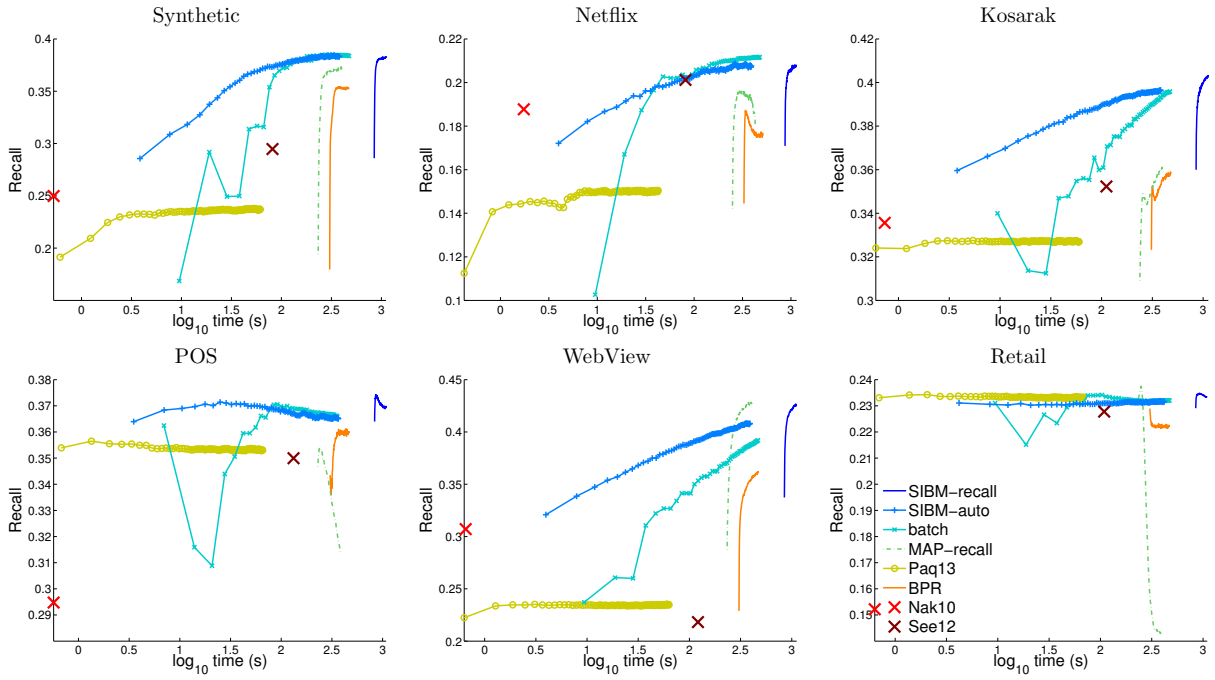

Figure 2: Average recall versus time for each method on each small dataset.

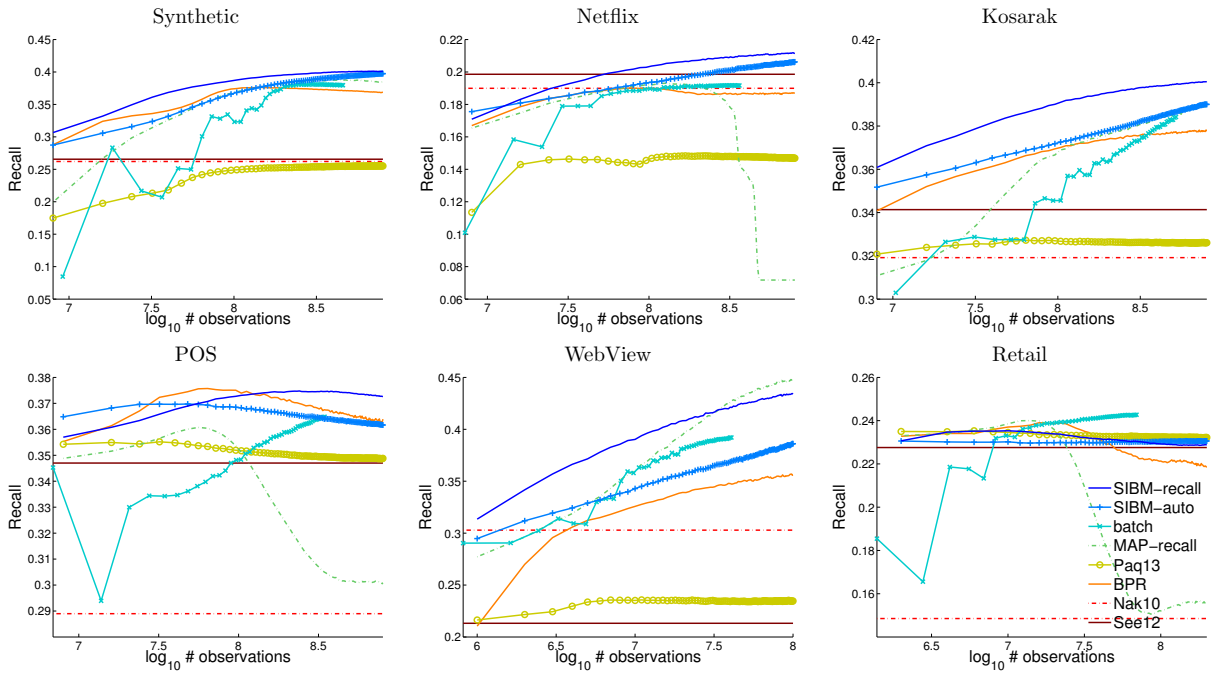

Figure 3: Average recall versus number of samples for each method on each large dataset.

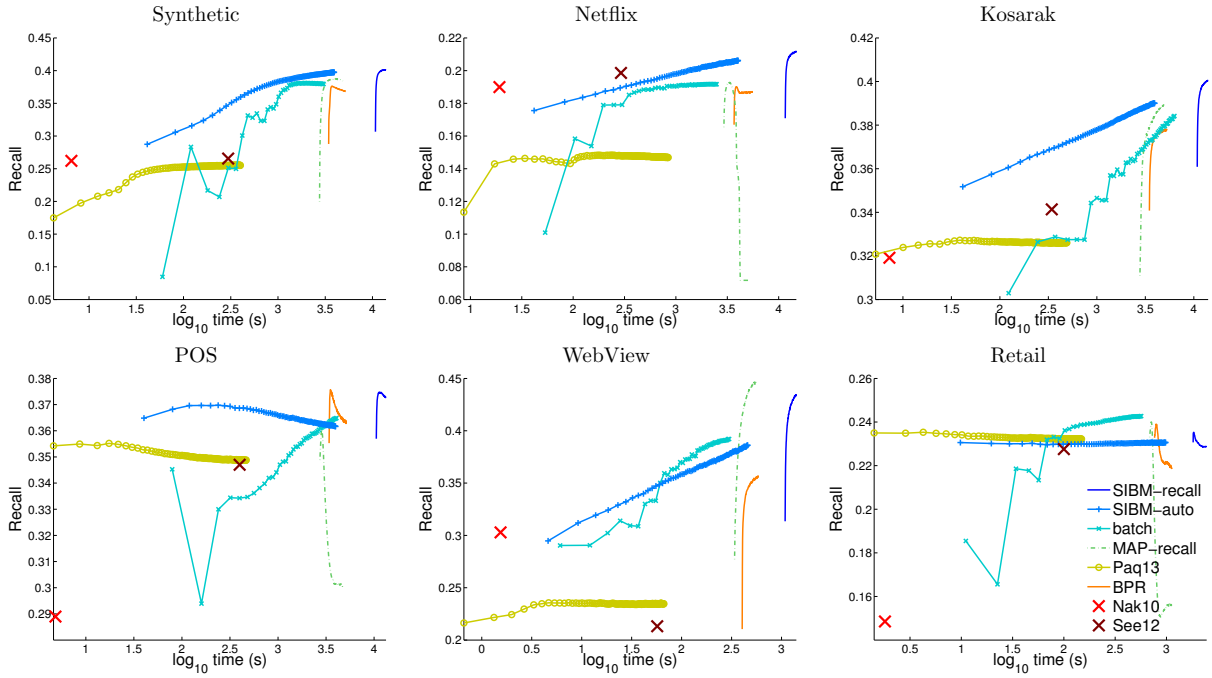

Figure 4: Average recall versus time for each method on each large dataset.

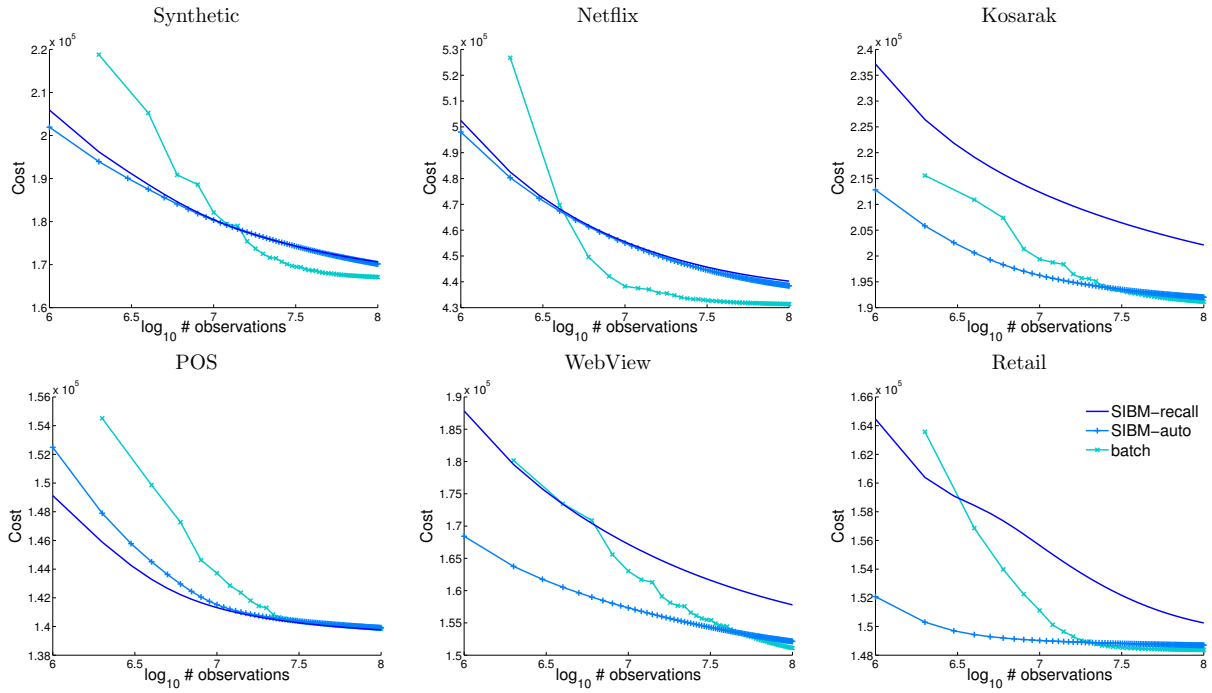

Figure 5: Average cost (negative ELBO) versus number of samples for each method on each small dataset.

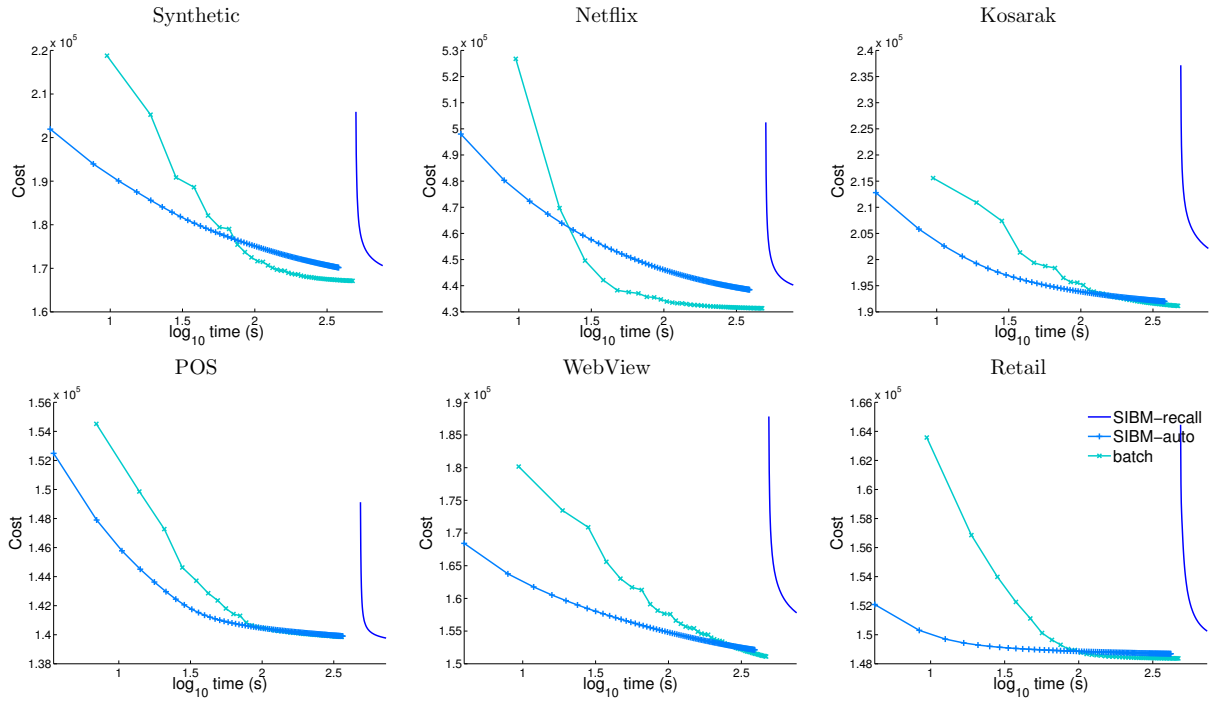

Figure 6: Average cost (negative ELBO) versus time for each method on each small dataset.

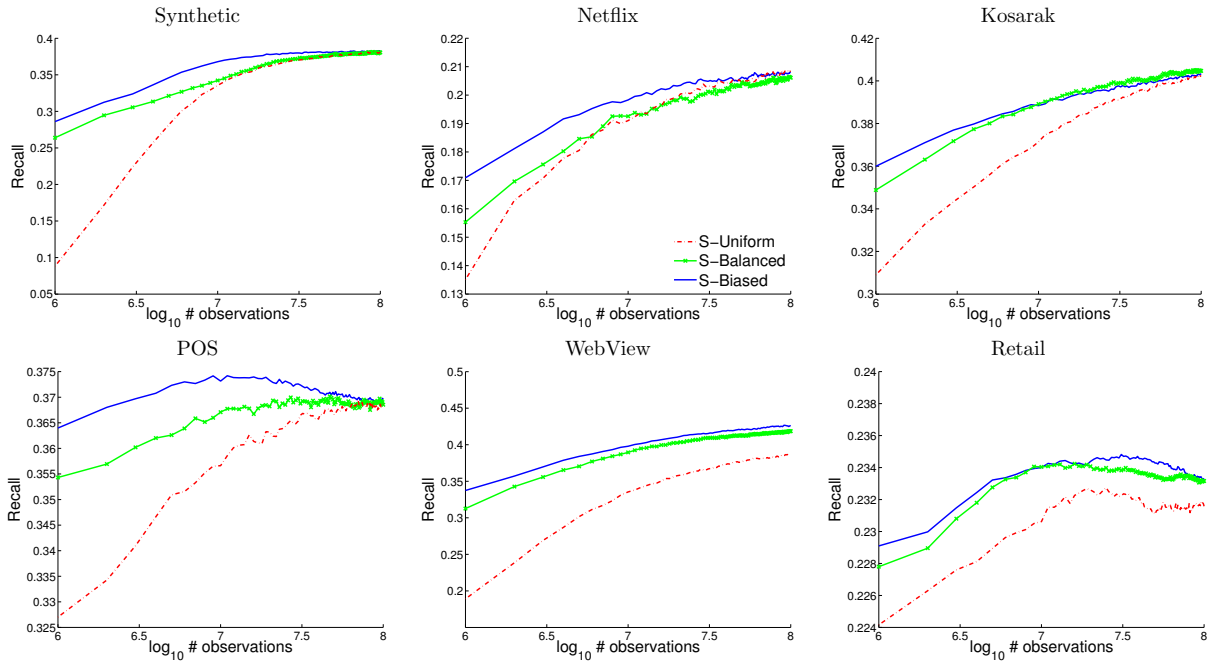

Figure 7: Average recall versus number of samples for each sampling strategy with the SIBM-recall algorithm.

References

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.