Supplementary Material
Marginalized Stochastic Natural Gradients for Black-Box Variational Inference

Geng Ji Debora Sujono Erik B. Sudderth

A. MSNG Update for Categorical Variables

We generalize our MSNG method to discrete variables with $K > 2$ states. We define $q(z_i = k) \triangleq \mu_{ik}$, and parameterize the variational distribution using natural parameters $\tau_i \triangleq \left[\log \frac{\mu_{i1}}{\mu_{iK}}, \ldots, \log \frac{\mu_{ik}}{\mu_{iK}}, \ldots, \log \frac{\mu_{iK-1}}{\mu_{iK}}\right]^T$. The Fisher information $F(\tau_i)$ is a $(K-1) \times (K-1)$ matrix whose $(k,\ell)^{\text{th}}$ entry is

$$F(\tau_i)_{k\ell} = \frac{\partial}{\partial \tau_{i\ell}} \left(\frac{e^{\tau_{ik}}}{1 + \sum_{k'=1}^{K-1} e^{\tau_{ik'}}} \right). \tag{A.1}$$

Via the chain rule, the gradient of the ELBO $\mathcal{L}$ with respect to τ_i equals $\frac{\partial \mathcal{L}}{\partial \tau_i} = \nabla_{\tau_i}\mu_i \cdot \frac{\partial \mathcal{L}}{\partial \mu_i}$. The Jacobian matrix $\nabla_{\tau_i}\mu_i$ of the soft-max transform from natural to moment parameters is also a $(K-1) \times (K-1)$ matrix, with entries

$$(\nabla_{\tau_i}\mu_i)_{k\ell} = \frac{\partial \mu_{ik}}{\partial \tau_{i\ell}} = \frac{\partial}{\partial \tau_{i\ell}} \left(\frac{e^{\tau_{ik}}}{1 + \sum_{k'=1}^{K-1} e^{\tau_{ik'}}} \right). \tag{A.2}$$

Like the binary case detailed in the main paper, $\nabla_{\tau_i}\mu_i$ cancels the inverse Fisher information when we compute the natural gradient, so $F^{-1}(\tau_i)\frac{\partial \mathcal{L}}{\partial \tau_i} = \frac{\partial \mathcal{L}}{\partial \mu_i}$. Explicitly marginalizing the K states of z_i, this gradient becomes

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mu_i} &= \mathbb{E}_{q(z)}\left[\frac{\partial \log q(z_i)}{\partial \mu_i}\left(\log p(z_i|z_{-i},x) - \log q(z_i)\right)\right] \\
&= \sum_{k=1}^{K-1} \mu_{ik}\frac{\partial \log \mu_{ik}}{\partial \mu_i}\mathbb{E}_{q(z_{-i})}\left[\log p(z_i = k|z_{-i},x) - \log \mu_{ik}\right] \\
&\quad + \mu_{iK}\frac{\partial \log \mu_{iK}}{\partial \mu_i}\mathbb{E}_{q(z_{-i})}\left[\log p(z_i = K|z_{-i},x) - \log \mu_{iK}\right]
\end{aligned}$$

We can then use Monte Carlo samples from $q(z_{-i})$ to approximate the k^{th} entry of the gradient vector as follows:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mu_{ik}} &= \frac{\mu_{ik}}{\mu_{ik}} \cdot \mathbb{E}_{q(z_{-i})}[\log p(z_i = k|z_{-i},x) - \log \mu_{ik}] \\
&\quad - \frac{\mu_{iK}}{\mu_{iK}} \cdot \mathbb{E}_{q(z_{-i})}[\log p(z_i = K|z_{-i},x) - \log \mu_{iK}] \\
&= \mathbb{E}_{q(z_{-i})}\left[\log \frac{p(z_i = k \mid z_{-i},x)}{p(z_i = K \mid z_{-i},x)}\right] - \tau_{ik} \\
&\approx \frac{1}{M}\sum_{m=1}^{M} \log \frac{p(z_i = k \mid z_{-i}^{(m)},x)}{p(z_i = K \mid z_{-i}^{(m)},x)} - \tau_{ik}.
\end{aligned}$$

Therefore, the MSNG update of τ_{ik} for $k = 1, \ldots, K-1$ with learning rate α simplifies to:

$$\begin{aligned}
\tau_{ik}^{\text{new}} &= \tau_{ik} + \alpha\left(F^{-1}(\tau_i)\frac{\partial \mathcal{L}}{\partial \tau_i}\right)_k \\
&= \tau_{ik} + \alpha\left(\mathbb{E}_{q(z_{-i})}\left[\log \frac{p(z_i = k \mid z_{-i},x)}{p(z_i = K \mid z_{-i},x)}\right] - \tau_{ik}\right) \\
&= \alpha\mathbb{E}_{q(z_{-i})}\left[\log \frac{p(z_i = k \mid z_{-i},x)}{p(z_i = K \mid z_{-i},x)}\right] + (1-\alpha)\tau_{ik} \\
&\approx \alpha\frac{1}{M}\sum_{m=1}^{M} \log \frac{p(z_i = k \mid z_{-i}^{(m)},x)}{p(z_i = K \mid z_{-i}^{(m)},x)} + (1-\alpha)\tau_{ik}.
\end{aligned}$$

B. Auxiliary-variable Variational Algorithms

Here we provide more details about the auxiliary-variable methods (AUX) used in the three binary-variable models in the experiment section. With simpler, closed-form coordinate updates than the regular CAVI method, these algorithms optimize looser, hand-designed variational bounds than the ELBO. Because they are model dependent, we go through each of them one by one.

B.1. AUX Algorithm for Noisy-OR Belief Networks

The AUX algorithm we use for the deep noisy-OR belief networks is by Ji et al. (2019). It leverages the log-concavity of the noisy-OR function f in Eq. (1) of the main paper. Following Jaakkola & Jordan (1999), it applies Jensen's inequality to f and builds a lower bound to the ELBO by introducing the auxiliary categorical distribution r defined for the non-leak parents $k \in \mathcal{P}(i)$ of each node i:

$$\begin{aligned}
&f\big(w_{0i} + \textstyle\sum_{k\in\mathcal{P}(i)} w_{ki}z_k\big) \qquad\qquad (B.3)\\
&\geq f(w_{0i}) + \textstyle\sum_{k\in\mathcal{P}(i)} r_{ki}z_k\Big[f(u_{ki}) - f(w_{0i})\Big],
\end{aligned}$$

where $f(x) \triangleq \log\big(1-\exp(-x)\big)$, $u_{ki} \triangleq w_{0i} + \frac{w_{ki}}{r_{ki}}$, and r should satisfy $r_{ki} \geq 0, \sum_{k\in\mathcal{P}(i)} r_{ki} = 1$. Because the lower bound in Eq. (B.3) is a linear function of parent states z_k, the update equation for the variational distribution $q(z)$ has a simple closed form. The auxiliary variables r are updated using the following fixed-point iterations till convergence:

$$r_{ki} \propto q_k r_{ki}\Big[f(u_{ki}) - f(w_{0i}) - \frac{w_{ki}}{r_{ki}}f'(u_{ki})\Big],$$

where $f'(x) = \frac{\exp(-x)}{1-\exp(-x)}$ is the derivative of $f(x)$. The variational algorithm interleaves between the updates of $q(z)$ and r, and is guaranteed to converge to a local maximum of the auxiliary variational bound.

B.2. AUX Algorithm for Sigmoid Belief Networks

The AUX algorithm we use for the deep sigmoid belief networks is developed by Gan et al. (2015). It derives a lower bound of the ELBO by making use of the Pólya-Gamma data augmentation trick (Polson et al., 2013), which leads to a tractable lower bound for the logistic log likelihood

$$\begin{aligned}
&\log p(z_{ij} \mid z_{i+1}) = \log \sigma(w_{ij}^T z_{i+1} + c_j) \\
\geq &-\log 2 + (z_{i,j} - 0.5)(w_{ij}^T z_{i+1} + c_j) \\
&- 0.5 \cdot \gamma_{ij} \cdot (w_{ij}^T z_{i+1} + c_j)^2 \\
&+ \mathbb{E}_{q(\gamma_{ij})}[\log \mathrm{PG}(\gamma_{ij} \mid b, 0) - \log q(\gamma_{ij})],
\end{aligned}$$

because it is only a quadratic function of z. The optimal variational distribution for each augmented variable $\gamma_{ij} \sim \mathrm{PG}(b, 0)$ can be approximately updated via

$$\begin{aligned}
q(\gamma_{i,j}) &= \mathrm{PG}\left(1, \sqrt{\mathbb{E}_{q(z_{i+1})}[(w_{ij}^T z_{i+1} + c_j)^2]}\right) \\
&\approx \mathrm{PG}\left(1, w_{ij}^T \mathbb{E}_{q(z_{i+1})}[z_{i+1}] + c_j\right).
\end{aligned}$$

B.3. AUX Algorithm for Probit Latent-feature Models

The AUX algorithm we build for the probit latent-feature relational model is based on the data augmentation trick by Albert & Chib (1993). The naive mean-field variational distribution for the latent features is $q(z) = \prod_{i=1}^N \prod_{d=1}^D q(z_{id})$, in which N is the number of entities and D is the feature dimension. Each dimension z_{id} is a Bernoulli distribution $q(z_{id}) \sim \mathrm{Bernoulli}(q_{id})$, with the activation probability $\mu_{id} \triangleq q(z_{id} = 1)$ as the free parameter.

The standard ELBO $\mathcal{L}$ of the model is

$$\begin{aligned}
\mathcal{L}(q(z)) = &\sum_{i=1}^N \mathbb{E}_{q(z)}\Bigg[\sum_{d=1}^D z_{id} \log \rho + (1 - z_{id}) \log(1-\rho) \\
&+ \sum_{j>i}^N x_{ij} \log \Phi\Big(w_0 + \sum_{d=1}^D w_d z_{id} z_{jd}\Big) \\
&+ \sum_{j>i}^N (1 - x_{ij}) \log\left(1 - \Phi\Big(w_0 + \sum_{d=1}^D w_d z_{id} z_{jd}\Big)\right) \\
&- \sum_{d=1}^D z_{id} \log \mu_{id} + (1 - z_{id}) \log(1 - \mu_{id})\Bigg]. \qquad \text{(B.4)}
\end{aligned}$$

By using the thresholded Gaussian data augmentation trick (Albert & Chib, 1993), we introduce an auxiliary variable y_{ij} for each pair of entities. Then rows two and three of Eq. (B.4) would be equivalent to

$$\begin{aligned}
&\sum_{i=1}^N \sum_{j>i}^N \log \int 1\{y_{ij} \geq 0\}^{x_{ij}} \cdot 1\{y_{ij} < 0\}^{1-x_{ij}} \cdot \\
&\qquad \mathcal{N}(y_{ij} \mid w_0 + \sum_{d=1}^D w_d z_{id} z_{jd}, 1)\, \mathrm{d}y_{ij} \\
\geq &\sum_{i=1}^N \sum_{j>i}^N \mathbb{E}_{q(y_{ij})}\left[\log \frac{p(x_{ij} \mid y_{ij}) p(y_{ij} \mid z)}{q(y_{ij})}\right], \qquad \text{(B.5)}
\end{aligned}$$

in which $p(x_{ij} \mid y_{ij}) \triangleq 1\{y_{ij} \geq 0\}^{x_{ij}} 1\{y_{ij} < 0\}^{1-x_{ij}}$, and $p(y_{ij} \mid z) \triangleq \mathcal{N}(y_{ij} \mid w_0 + \sum_{d=1}^D w_d z_{id} z_{jd}, 1)$. The greater-than-or-equal-to sign comes from applying Jensen's inequality to the log function.

Bringing Eq. (B.5) back to Eq. (B.4), we get a lower bound of the original ELBO. It's also mathematically equivalent to the ELBO of a data-augmented model, in which a latent variable y_{ij} with a unit normal prior is added to each pair of entities. We interleave the updates of $q(y)$ and $q(z)$. Following Eq. (5) of the main paper, the optimal coordinate-ascent variational factor of y_{ij} is a truncated normal distribution:

$$\begin{aligned}
&q(y_{ij}) \qquad \text{(B.6)} \\
&= \begin{cases} \mathcal{TN}_+\big(y_{ij} \mid w_0 + \sum_d w_d q_{id} q_{jd}, 1\big), & \text{if } x_{ij} = 1; \\ \mathcal{TN}_-\big(y_{ij} \mid w_0 + \sum_d w_d q_{id} q_{jd}, 1\big), & \text{if } x_{ij} = 0. \end{cases}
\end{aligned}$$

Eq. (B.5) also shows that the data-augmented ELBO is a quadratic function of z, so the coordinate update for the latent feature distribution $q(z)$ can be efficiently computed:

$$\begin{aligned}
\mu_{id} = \Phi\Big(&\log \rho - \log(1-\rho) + \sum_{j \neq i} \mu_{jd} w_d \big(\mathbb{E}_{q(y_{ij})}[y_{ij}] \\
&- w_0 - \frac{1}{2} w_d - \sum_{e \neq d} w_e \mu_{ie} \mu_{je}\big)\Big),
\end{aligned}$$

where $q(y_{ij})$ is the truncated normal of Eq. (B.6), and its mean $\mathbb{E}_q[y_{ij}]$ can be computed via the unit normal CDF.

C. Code Explanation

Here we provide a brief explanation of the code in Figs. 1 and 3 from the main paper, focusing on the Pyro syntax. For inference, Pyro requires users to specify two stochastic functions: the generative model (simply called *model*) and the variational distribution or inference model (commonly referred to as *guide* in Pyro). The model and the guide have to take the same arguments. Lines 13-28 of Fig. 1 illustrate the model specification of a three-layer Bayesian network. We use `pyro.plate` to mark the conditional independence structure (lines 14-17), and `pyro.sample` to specify the random variables in the model (lines 19, 23, 27). Each sample or plate site is named. For example, in lines 18-20, we define $N \times D_{H_2}$ independent Bernoulli random variables called `z_top`. Fig. C.1 implements the

```
class BN:
    ...
    def guide(self, data):
        dat_axis = pyro.plate('dat_axis', data.shape[0], dim=0)
        top_axis = pyro.plate('top_axis', self.D_H2, dim=1)
        mid_axis = pyro.plate('mid_axis', self.D_H1, dim=1)
        with dat_axis, top_axis:
            logit_top = self.variational_params['z_top']
            pyro.sample('z_top', dist.Bernoulli(logits=logit_top))
        with dat_axis, mid_axis:
            logit_bottom = self.variational_params['z_bot']
            pyro.sample('z_bot', dist.Bernoulli(logits=logit_bottom))
```

Figure C.1. Mean-field guide for three-layer Bayesian networks.

guide for our Bayesian network example using mean-field variational distribution where each z_i is independent.

Fig. 3 shows the implementation of one optimization step of MSNG in Pyro. We store the variational parameters to optimize in a dictionary called `variational_params` with sample site names as keys. `poutine.trace` executes the model, and records sampled values from `pyro.sample` statements and the corresponding log probabilities in a dictionary data structure. `poutine.replay` conditions the model on the previously sampled trace. Concretely, in line 13 we sample $z^{(m)}$ from our variational distribution. These sampled values can be accessed through the `value` key of `guide_trace.nodes[group]`. Since each sample site is a tensor, we use `idx` to index into each element in the tensor to access variable z_i (lines 20, 24). For each possible value k of each variable z_i, we fix $z_i = k$ and condition the model on the rest of the sampled $z_{-i}^{(m)}$ by calling `poutine.replay` in line 25. Finally, we run `poutine.trace` on the conditioned model to obtain the new log probabilities. `_compute_log_p` in line 29 is implemented to compute these log joint probabilities more efficiently by leveraging the conditional independence structure of the model specified using Pyro's plate notation.

D. Experimental Details

D.1. Deep Noisy-OR and Sigmoid Belief Networks

In the main paper, we uniformly initialize variational distributions as $\mu_i = 0.5$ for both models. We additionally try initializing each latent node in the sigmoid belief network to its marginal prior distribution, approximated using Monte Carlo sampling (Gan et al., 2015). In the results shown in Fig. D.2, the final ELBO of AUX improves to -139, better than the -192 under the uniform initialization. It is still far worse than MSNG, which is much more robust to different initializations (-109 under both initial value settings). Detailed ELBO values of different methods are recorded in Tables D.1 and D.2. They show that MSNG outperforms other approaches on almost all datasets and models.

D.2. Categorical and Binary Relational Models

For the categorical stochastic block model of relational data, we set $K = 5$ for both the countries and NeurIPS datasets.

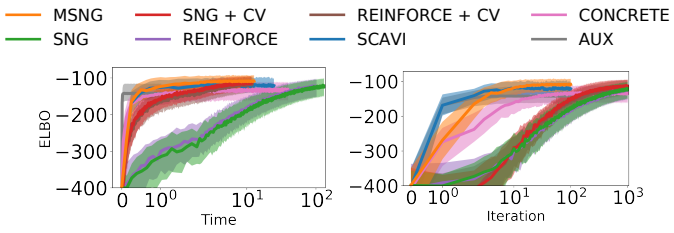


Figure D.2. Improvement of ELBO over runtime (left) and iteration (right) of the sigmoid belief network on MNIST dataset, using marginal prior initializations. AUX converges to higher local optimum than Fig. 4(b) of the main the paper with uniform initializations. MSNG proves to be more robust to initial values.

```
class SBM:
    def __init__(self, hyperparams):
        self.K, self.W, self.theta = hyperparams

    def model(self, links):
        row_idx, col_idx = tuple(torch.triu_indices(len(links), len(links), 1))
        obs = links[row_idx, col_idx]

        entity_axis = pyro.plate('entity_axis', len(links))
        link_axis = pyro.plate('link_axis', len(obs))
        with entity_axis:
            z = pyro.sample('z', dist.Categorical(self.theta))
            p = self.W[z[row_idx], z[col_idx]]
        with link_axis:
            pyro.sample('x', dist.Bernoulli(p), obs=obs)
```

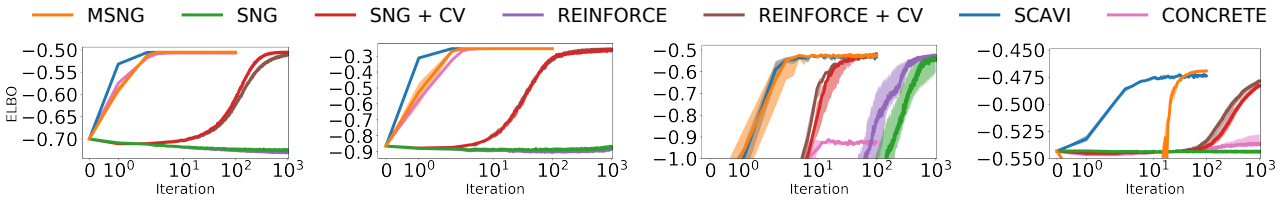
```
class LFRM:
    def __init__(self, hyperparams):
        self.D, self.W, self.W0, self.z_prior = hyperparams

    def squash_fun(self, x):
        return dist.Normal(loc=0.0, scale=1.0).cdf(x)

    def model(self, links):
        triu_idx = tuple(torch.triu_indices(len(links), len(links), 1))
        obs = links[triu_idx]

        entity_axis = pyro.plate('entity_axis', len(links), dim=0)
        D_axis = pyro.plate('D_axis', self.D, dim=1)
        link_axis = pyro.plate('link_axis', len(obs))
        with entity_axis, D_axis:
            features = pyro.sample('features',
                                   dist.Bernoulli(self.z_prior))
        wz = self.W0 + torch.einsum('id,jd->ij',
                                    self.W * features, features)[triu_idx]
        with link_axis:
            pyro.sample('links', dist.Bernoulli(self.squash_fun(wz)),
                        obs=obs)
```

Figure D.3. Pyro code for stochastic block models (top) and probit latent-feature relational models (bottom).

We assign a high interaction probability within the same community, and a low probability between different communities: $w_{kk} = 0.9$, and $w_{k\ell} = 0.05$ if $k \neq l$. We assign uniform probabilities for prior community memberships, $\pi_k = \frac{1}{K}$, and initialize the variational distributions uniformly as $\mu_{ik} = \frac{1}{K}$.

For the binary probit latent-feature relational model, we set $D = 4$ for the countries dataset. Model parameters $w_d = 2, w_0 = -2, \rho = 0.5$ are selected through grid search. For the NeurIPS dataset, we set $D = 10, w_d = 2, w_0 = -2, \rho = 0.1$. We initialize $\mu_{id} = \rho$ for all features.

Fig. D.3 gives Pyro specifications for both relational models. To supplement Fig. 5 in the main paper, Fig. D.4 shows ELBO improvement versus iteration for categorical models of relational data and crowd-sourced annotations.



(a) RTE (annotation) (b) WSD (annotation) (c) countries (SBM) (d) NeurIPS (SBM)

Figure D.4. Improvement of ELBO values (y axis) over iteration (x axis) on various datasets using general categorical models. Lines show the median, while shaded regions show the 25th - 75th percentiles of ELBO values across 10 runs.

Table D.1. ELBO values at 100-th iteration for all dataset(model, initialization) combinations. Top methods with the same ELBO values up to the first 2 decimal places are highlighted in bold. The number of samples used for each method is described in the main paper. The temperature hyperparameters for CONCRETE runs are tuned from possible values $\{0.01, 0.1, 1, 10, 100\}$ for best performance, which end up with 0.1 for NeurIPS (SBM) and 10 for all else.

Dataset	MSNG	SNG	SNG+CV	REIN	REIN+CV	SCAVI	CONC	AUX
Newsgroups (noisy-OR)	-16.520	-26.960	-21.974	-29.876	-31.697	-16.566	-79.216	**-16.120**
MNIST (sigmoid, 0.5)	**-109.280**	-214.950	-182.24	-222.612	-196.087	-118.094	-135.33	-191.612
MNIST (sigmoid, prior)	**-109.216**	-182.424	-152.254	-182.355	-157.370	-118.502	-135.715	-138.596
Countries (probit)	**-0.498**	-0.581	-0.578	-0.590	-0.589	-0.574	-0.537	-0.626
NeurIPS (probit)	**-0.086**	-0.179	-0.123	-0.179	-0.125	**-0.086**	-0.114	-0.113
Countries (SBM)	**-0.525**	-1.070	**-0.522**	-0.866	**-0.526**	-0.535	-0.926	-
NeurIPS (SBM)	**-0.469**	-0.544	-0.539	-0.544	-0.534	-0.474	-0.540	-
RTE (annotation)	**-0.505**	-0.723	-0.608	-0.726	-0.623	**-0.505**	**-0.507**	-
WSD (annotation)	**-0.254**	-0.884	-0.315	-0.899	-0.325	**-0.254**	**-0.254**	-

Table D.2. Final ELBO values for all dataset(model, initialization) combinations. Fast converging methods MSNG, SCAVI and AUX run 100 iterations. SNG(+CV), REINF(+CV) and CONCRETE run 1,000 iterations if they do not converge within the first 100 iterations. Top methods with the same ELBO values up to the first 2 decimal places are highlighted in bold. With more iterations, the performance of SNG+CV and REINF+CV starts to approach MSNG, but is still inferior for many models. The biased CONCRETE method converges to inferior ELBO values than SNG+CV and REIN+CV, despite faster ELBO improvement at the beginning iterations shown in Table D.1.

Dataset	MSNG	SNG	SNG+CV	REIN	REIN+CV	SCAVI	CONC	AUX
Newsgroups (noisy-OR)	-16.520	-17.538	-16.214	-17.386	-18.504	-16.566	-53.388	**-16.120**
MNIST (sigmoid, 0.5)	**-109.280**	-130.672	-116.990	-132.844	-122.096	-118.094	-133.879	-191.612
MNIST (sigmoid, prior)	**-109.216**	-123.054	-113.988	-124.574	-115.539	-118.502	-133.591	-138.596
Countries (probit)	**-0.498**	-0.503	-0.501	-0.504	-0.501	-0.574	-0.517	-0.626
NeurIPS (probit)	**-0.086**	-0.173	-0.112	-0.173	-0.113	**-0.086**	-0.114	-0.113
Countries (SBM)	**-0.525**	-0.586	**-0.522**	-0.555	**-0.526**	-0.535	-0.926	-
NeurIPS (SBM)	**-0.469**	-0.544	-0.483	-0.544	-0.480	-0.474	-0.533	-
RTE (annotation)	**-0.505**	-0.725	**-0.505**	-0.730	-0.511	**-0.505**	**-0.507**	-
WSD (annotation)	**-0.254**	-0.868	-0.267	-0.882	**-0.257**	**-0.254**	**-0.254**	-

D.3. Annotation Models

Because the RTE and WSD datasets contain ground-truth labels, we set K to match the true number of categories ($K = 2$ for the RTE data, and $K = 3$ for the WSD data). We omit these labels for our experiments, and treat them as the latent variables in the model. We fix the label category distribution to be uniform, $\pi_k = \frac{1}{K}$, and assign conjugate priors for the annotator reliability distributions $\theta_{jk} \sim \text{Dirichlet}(\beta_k)$. We set the Dirichlet hyperparameters to be biased towards prediction of correct labels: $\beta_{kk} = 5$, and $\beta_{k\ell} = 1$ if $k \neq \ell$. The variational distribution is initialized uniformly as $\mu_{ik} = \frac{1}{K}$.

We analytically marginalize the unknown rater accuracy distributions θ_{jk} to create a model with only discrete latent variables. Let A be the total number of annotators, n_k the number of items whose true category is k, $n_{jk\ell}$ the number of times annotator j assigns label ℓ to an item whose true category is k, and $n_{jk\cdot} \triangleq \sum_\ell n_{jk\ell}$. The marginal joint distribution $p(z, x \mid \pi, \beta)$ can then be written as

$$\prod_k \pi_k^{n_k} \left(\frac{\Gamma(\sum_\ell \beta_{k\ell})}{\prod_\ell \Gamma(\beta_{k\ell})} \right)^A \prod_{j=1}^A \frac{\prod_\ell \Gamma(n_{jk\ell} + \beta_{k\ell})}{\Gamma(n_{jk\cdot} + \sum_\ell \beta_{k\ell})}, \quad \text{(D.7)}$$

where the gamma functions $\Gamma(\cdot)$ arise from the Dirichlet normalization constants.

References

Albert, J. H. and Chib, S. Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88(422):669–679, 1993.

Gan, Z., Henao, R., Carlson, D., and Carin, L. Learning deep sigmoid belief networks with data augmentation. In *Artificial Intelligence and Statistics*, 2015.

Jaakkola, T. S. and Jordan, M. I. Variational probabilistic inference and the QMR-DT network. *Journal of Artificial Intelligence Research*, 10:291–322, 1999.

Ji, G., Cheng, D., Ning, H., Yuan, C., Zhou, H., Xiong, L., and Sudderth, E. B. Variational training for large-scale noisy-OR Bayesian networks. In *Conference on Uncertainty in Artificial Intelligence*, 2019.

Polson, N. G., Scott, J. G., and Windle, J. Bayesian inference for logistic models using Pólya–Gamma latent variables. *Journal of the American statistical Association*, 108(504):1339–1349, 2013.