Compass Mini-Project: Massively Parallel MCMC

Sam Bowyer
School of Mathematics
University of Bristol
sam.bowyer@bristol.ac.uk

Abstract

Recent work Aitchison [2019], Heap and Aitchison [2023], Bowyer et al. [2023] has shown that a "massively parallel" approach to the Bayesian inference techniques of importance weighting and sampling can achieve superior results in less computation time than standard 'global' importance weighting and sampling methods. This massively parallel approach works by considering all K^n possible combinations of K samples drawn for each of the n latent variables involved, thereby effectively reasoning about an exponential number of samples. This is particularly helpful because results from Chatterjee and Diaconis [2018] indicate that the number of samples required for accurate approximation of the true posterior via importance weighting grows exponentially with n. The computational issues arising in dealing with all K^n combinations can be dealt with firstly by carefully considering the conditional dependencies present in the generative model, and secondly by applying a trick involving differentiating through a marginal likelihood estimator. In this report we provide an introduction to the massively parallel framework by examining importance sampling and weighting, before considering a massively parallel Markov chain Monte Carlo method and implementing a small toy experiment.

1 Introduction

Importance weighting has found great success in improving probabilistic inference techniques by reweighting several samples at a time drawn from a proposal relevant to the problem at hand. Two well known examples of this are reweighted wake-sleep (RWS) Bornschein and Bengio [2015] and importance weighted autoencoders (IWAEs) Burda et al. [2016], which improve upon wake-sleep Hinton et al. [1995] and variational autoencoders Kingma and Welling [2014] respectively by calculating objective functions with $K > 1$ weighted samples per latent variable.

However, recent work by Chatterjee and Diaconis Chatterjee and Diaconis [2018] suggests that as the number of latent variables n increases, the number of samples needed for effective importance weighting grows with $\mathcal{O}(e^n)$, which can easily render even slightly large importance weighted inference problems intractable. In this report, we consider an approach to overcoming this issue by drawing K samples per latent variable and then considering all possible K^n combinations of samples. The difficulties involved with working on such a large number of combinations can be ameliorated by considering the dependency graph of the model and using tensor operations (e.g. with PyTorch Paszke et al. [2019] and opt-einsum Daniel et al. [2018]) parallelised on GPUs.

This approach was taken in the development of Tensor Monte Carlo (TMC) Aitchison [2019], which showed superior performance to IWAE with only a slight increase in computation time. Subsequent work shows that similar, but more general methods can be used to improve upon IWAE further and can also be used to improve RWS Heap and Aitchison [2023]. This improvement upon the TMC approach is referred to as the "massively parallel" framework and forms the basis of this mini-project.

In particular, in Section 2 we first examine the use of this framework in importance sampling and weighting (as presented in Bowyer et al. [2023][1]), in order to introduce its key elements and results, whilst in Section 3 we use these ideas (with a slightly different setup) to discuss a massively parallel Markov chain Monte Carlo method for which we implement a small toy experiment.

2 Massively Parallel Importance Weighting and Sampling

2.1 Background

Bayesian inference relies on computing a posterior distribution,

$$P(z'|x) = \frac{P(x|z')P(z')}{\sum_{z''} P(x, z'')} \tag{1}$$

given a prior $P(z')$ over latent variables z' and a likelihood $P(x|z')$ for data x. From this, we are often interested in calculating posterior expectations,

$$m_{\text{post}}(x) = \sum_{z'} P(z'|x)m(z'). \tag{2}$$

However, computation of $P(z'|x)$ is often intractable and so instead we make use of a proposal distribution $Q(z)$ along with an estimate of the marginal likelihood $P(x) = \sum_{z''} P(x, z'')$ within a method such as importance sampling.

2.1.1 The Global Marginal Likelihood Estimator

Regular importance samples work by drawing K samples from the full joint state space denoted

$$z = (z^1, ..., z^k) \in \mathcal{Z}^K, \tag{3}$$

with a single sample denoted $z^k = (z^k_1, z^k_2, ..., z^k_n) \in \mathcal{Z}$ (i.e. each being made from n latent variables). This is obtained by sampling K times from the proposal where $Q(z) = \prod_{k=1}^K Q(z^k)$.

We define a 'global' (following terminology from Geffner and Domke [2022]) estimator $\mathcal{P}_{\text{global}}(z)$ of the marginal likelihood as follows:

$$\mathcal{P}_{\text{global}}(z) = \tfrac{1}{K} \sum_{k\in\mathcal{K}} r_k(z) \tag{4}$$

where $\mathcal{K} = \{1, ..., K\}$ and

$$r_k(z) = \frac{P(x, z^k)}{Q(z^k)}. \tag{5}$$

Since $\mathcal{P}_{\text{global}}(z)$ is an average of K i.i.d. terms, we can easily see that it is indeed an unbiased estimator of the marginal likelihood by first converting the expression to an expectation over $Q(z^k)$ rather than $Q(z)$, at which point the rest follows from the definition of the expectation,

$$\mathbb{E}_{Q(z)}[\mathcal{P}_{\text{global}}(z)] = \mathbb{E}_{Q(z^k)}\left[\frac{P(x, z^k)}{Q(z^k)}\right] = \sum_{z^k}\left[\frac{P(x, z^k)}{Q(z^k)}Q(z^k)\right] = \sum_{z^k}\left[P(x, z^k)\right] = P(x). \tag{6}$$

2.1.2 The Massively Parallel Marginal Likelihood Estimator

In the massively parallel approach we use a slightly different marginal likelihood estimator by considering all possible K^n combinations of the n latent variables that make up each of our K samples inside z. In particular, writing the kth sample of the ith latent variable as z^k_i, the collection of all K samples of the ith latent variable can be written as

$$z_i = (z^1_i, ..., z^k_i). \tag{7}$$

[1]A large part of this mini-project involved working on this paper, particularly the results presented here in Section 2.3. A draft of the paper is included in the appendix of this report.

Then the full collection of all K samples of all n latent variables (as we had before in Eq. 3) can now be written as

$$z = (z_1, ..., z_n). \tag{8}$$

Next we define a vector of indices $\mathbf{k} = (k_1, ..., k_n) \in \mathcal{K}^n$, which allows us to write a combination of the samples with these indices as

$$z^{\mathbf{k}} = \left(z_1^{k_1}, z_2^{k_2}, \ldots, z_n^{k_n}\right) \in \mathcal{Z} \tag{9}$$

where $z_i^{k_j}$ represents the k_jth sample of the ith latent variable.

We must also now clarify how our generative probabilities $P(x, z^{\mathbf{k}})$ and massively parallel proposals $Q_{\text{MP}}(z)$ will work. To do this, we note that since we have multiple latent variables we can utilise a graphical model structure, namely

$$P(x, z^{\mathbf{k}}) = P\left(x | z_j^{k_j} \ \forall j \in \text{pa}(x)\right) \prod_{i=1}^{n} P\left(z_i^{k_i} | z_j^{k_j} \ \forall j \in \text{pa}(i)\right) \tag{10}$$

where $pa(i)$ is the set of indices of parents of the ith latent variable in the generative model (and similarly $pa(x)$ is the set of indices of parents for x).

For the proposal, we don't have to explicitly work with all K^n combinations of K samples over n latent variables in the same way, meaning that we can just define $Q_{\text{MP}}(z)$, and don't necessarily need to write $Q_{\text{MP}}(z^{\mathbf{k}})$ explicitly. However, we still do utilise the graphical model structure of Q_{MP} in its definition as follows

$$Q_{\text{MP}}(z) = \prod_{i=1}^{n} Q_{\text{MP}}\left(z_i | z_j \ \forall j \in \text{qa}(i)\right), \tag{11}$$

where qa(i) is the set of indices of parents of z_i under the proposal's graphical model. In order to incorporate the K different samples, when calculating $Q_{\text{MP}}(z_i | z_j \ \forall j \in \text{qa}(i))$ we use a permutation over samples of parent latent variables. Other proposal structures are possible, however, this has the benefit that each of the $Q_{\text{MP}}(z_i^k | z_j \ \forall j \in \text{qa}(i))$ values are easy to compute.

The massively parallel estimator of the marginal likelihood can then be written as:

$$\mathcal{P}_{\text{MP}}(z) = \tfrac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} r_{\mathbf{k}}(z) \tag{12}$$

where

$$r_{\mathbf{k}}(z) = \frac{P(x, z^{\mathbf{k}})}{\prod_{i=1}^{n} Q_{\text{MP}}(z_i^k | z_j^k \text{ for } j \in \text{qa}(i))} \tag{13}$$

The proof that this is indeed a marginal likelihood estimator is presented in Heap and Aitchison [2023] along with an explanation on how to compute it efficiently (following arguments similar to those provided in Aitchison [2019]). This efficient computation exploits the graphical model's structure of independencies and relies on the fact that we can actually write $r_{\mathbf{k}}(z)$ as a product of low-rank tensors (low-rank due to each variable having typically small number of parents, $|\text{pa}(i)|$). Therefore $\mathcal{P}_{\text{MP}}(z)$ can be computed as a large tensor product efficiently with careful orderings of the sums and products (e.g. using the Python package opt-einsum Daniel et al. [2018]). This is how the massively-parallel approach achieves its efficiency over an exponential number of samples, however, the exact details are not required for the content within this report.

2.2 Methods

2.2.1 Posterior Expectations

With the two marginal likelihood estimators given above, $\mathcal{P}_{\text{global}}$ and $\mathcal{P}_{\text{MP}}$, we can generate the necessary expectations (recalling that whilst both $z^k \in \mathcal{Z}$ and $z^{\mathbf{k}} \in \mathcal{Z}$, we're indexing with $k \in \mathcal{K}$ in the global case and with $\mathbf{k} \in \mathcal{K}^n$ in the massively parallel case[2]):

$$m_{\text{global}}(z) = \tfrac{1}{K} \sum_{k \in \mathcal{K}} \frac{r_k(z)}{\mathcal{P}_{\text{global}}(z)} m(z^k) \tag{14}$$

$$m_{\text{MP}}(z) = \tfrac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} \frac{r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)} m(z^{\mathbf{k}}). \tag{15}$$

Note that it is not obvious that these are both valid importance-sampled moment estimators since, unlike Eq. 2, they do not explicitly use the true posterior $P(z|x)$. However, their validity is proved in the appendix of Bowyer et al. [2023], which makes use of the fact that our two marginal likelihood estimators are unbiased.

Computing posterior moments, and as such, computing Eq. 15, through techniques for graphical models is often involves complex operations (e.g. Obermeyer et al. [2019]) that can be computationally expensive. Instead, we compute Eq. 15 using the 'source term trick', in which differentiating through the log of a modified version of our marginal likelihood estimator $\mathcal{P}_{\text{MP}}(z)$ leads us to exactly the posterior moments we're looking for.

To this end, we introduce a 'source term' $e^{Jm(z^{\mathbf{k}})}$ for $J \in \mathbb{R}$ to create our modified marginal likelihood estimator:

$$\mathcal{P}^{\text{exp}}_{\text{MP}}(z, J) = \tfrac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} r_{\mathbf{k}}(z) e^{Jm(z^{\mathbf{k}})}. \tag{16}$$

Note that $\mathcal{P}^{\text{exp}}_{\text{MP}}(z, J = 0) = \mathcal{P}_{\text{MP}}(z)$, and also that the source term, like the factors that make up $r_{\mathbf{k}}(z)$, can be thought of as a tensor indexed by some subset of $\mathbf{k}$ (depending on m), hence efficient computation of this sum is still possible with the right implementation e.g. using opt-einsum Daniel et al. [2018].

The trick comes into play by differentiating $\log \mathcal{P}^{\text{exp}}_{\text{MP}}(z, J)$ at $J = 0$,

$$\begin{aligned}
\frac{\partial}{\partial J}\bigg|_{J=0} \log \mathcal{P}^{\text{exp}}_{\text{MP}}(z, J) &= \frac{\frac{\partial}{\partial J}\big|_{J=0} \mathcal{P}^{\text{exp}}_{\text{MP}}(z, J)}{\mathcal{P}^{\text{exp}}_{\text{MP}}(z, 0)} && (17) \\
&= \frac{\frac{\partial}{\partial J}\big|_{J=0} \mathcal{P}^{\text{exp}}_{\text{MP}}(z, J)}{\mathcal{P}_{\text{MP}}(z)} && (18) \\
&= \frac{\frac{\partial}{\partial J}\big|_{J=0} \frac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} r_{\mathbf{k}}(z) e^{Jm(z^{\mathbf{k}})}}{\mathcal{P}_{\text{MP}}(z)} && (19) \\
&= \frac{\frac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} r_{\mathbf{k}}(z) \frac{\partial}{\partial J}\big|_{J=0} e^{Jm(z^{\mathbf{k}})}}{\mathcal{P}_{\text{MP}}(z)}. && (20)
\end{aligned}$$

And since $\frac{\partial}{\partial J}\big|_{J=0} e^{Jm(z^{\mathbf{k}})} = m(z^{\mathbf{k}})$, by the definition of $m_{\text{MP}}(z)$ in Eq. 15 we see that we arrive at the posterior moments as desired

$$\begin{aligned}
\frac{\partial}{\partial J}\bigg|_{J=0} \log \mathcal{P}^{\text{exp}}_{\text{MP}}(z, J) &= \frac{\frac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} r_{\mathbf{k}}(z) m(z^{\mathbf{k}})}{\mathcal{P}_{\text{MP}}(z)} && (21) \\
&= m_{\text{MP}}(z). && (22)
\end{aligned}$$

2.2.2 Importance Weights

We reach a more flexible result if we consider how to compute marginal importance weights for each latent variable (meaning that we can calculate arbitrary expectations per variable). To do this, we define our marginal importance weights for the ith latent variable as

$$w^i_{k_i} = \frac{\frac{1}{K^n} \sum_{\mathbf{k}/k_i \in \mathcal{K}^{n-1}} r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)}, \tag{23}$$

[2]Indeed, note that for a given z we have that z^k and $z^{\mathbf{k}}$ are equal when $\mathbf{k} = (k, ..., k) \in \mathcal{K}^n$.

where $\mathbf{k}/k_i = (k_1, ..., k_{i-1}, k_{i+1}, ..., k_n) \in \mathcal{K}^{n-1}$. This allows us to write the moment for the ith latent variable as a sum over k_i

$$m_{\text{MP}}(z) = \sum_{\mathbf{k}\in\mathcal{K}^n} \frac{r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)} m(z_i^{k_i}) = \sum_{k_i} w_{k_i}^i m(z_i^{k_i}). \tag{24}$$

To calculate the weights in Eq. 23 we use a slightly modified version of the source trick wherein we introduce a vector $\mathbf{J} \in \mathbb{R}^K$ to another marginal likelihood estimator,

$$\mathcal{P}_{\text{MP}}^{\text{marg}}(z, \mathbf{J}) = \tfrac{1}{K^n} \sum_{\mathbf{k}\in\mathcal{K}^n} r_{\mathbf{k}}(z) e^{J_{k_i}} \tag{25}$$

and differentiate $\log \mathcal{P}_{\text{MP}}^{\text{marg}}(z, \mathbf{J})$ at $\mathbf{J} = \mathbf{0}$ with respect to $J_{k_i'}$,

$$\left.\frac{\partial}{\partial J_{k_i'}}\right|_{\mathbf{J}=\mathbf{0}} \log \mathcal{P}_{\text{MP}}^{\text{marg}}(z, \mathbf{J}) = \frac{\left.\frac{\partial}{\partial J_{k_i'}}\right|_{\mathbf{J}=\mathbf{0}} \mathcal{P}_{\text{MP}}^{\text{marg}}(z, \mathbf{J})}{\mathcal{P}_{\text{MP}}^{\text{marg}}(z, \mathbf{0})} \tag{26}$$

$$= \frac{\frac{1}{K^n} \sum_{\mathbf{k}\in\mathcal{K}^n} r_{\mathbf{k}}(z) \left.\frac{\partial}{\partial J_{k_i'}}\right|_{\mathbf{J}=\mathbf{0}} e^{J_{k_i}}}{\mathcal{P}_{\text{MP}}(z)}, \tag{27}$$

Where in the last equality we've used the fact that $\mathcal{P}_{\text{MP}}^{\text{marg}}(z, \mathbf{J} = \mathbf{0}) = \mathcal{P}_{\text{MP}}(z)$. Note that

$$\left.\frac{\partial}{\partial J_{k_i'}}\right|_{\mathbf{J}=\mathbf{0}} e^{J_{k_i}} = \begin{cases} 1 & \text{if } k_i' = k_i \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

This means that we can rewrite the numerator in Eq. 27 by only summing over $\mathbf{k}$ that don't include k_i, i.e. summing over $\mathbf{k}/k_i$, hence

$$\left.\frac{\partial}{\partial J_{k_i}}\right|_{\mathbf{J}=\mathbf{0}} \log \mathcal{P}_{\text{MP}}^{\text{marg}}(z, \mathbf{J}) = \frac{\frac{1}{K^n} \sum_{\mathbf{k}/k_i\in\mathcal{K}^{n-1}} r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)} \tag{29}$$

$$= w_{k_i}^i. \tag{30}$$

Therefore using the modified marginal likelihood estimator $\mathcal{P}_{\text{MP}}^{\text{marg}}(z, \mathbf{J})$ (which can be computed via Eq. 25) we are able to obtain the weights $w_{k_i}^i$ required to compute the wide variety of moments possible from Eq. 24.

2.2.3 Importance Samples

The final contribution in Bowyer et al. [2023] presents a massively parallel importance sampling method. In this context, we write the expectation estimates via a distribution over indices $P(\mathbf{k})$,

$$m_{\text{MP}}(z) = \sum_{\mathbf{k}\in\mathcal{K}} P(\mathbf{k}) m(z^{\mathbf{k}}) \tag{31}$$

$$P(\mathbf{k}) = \frac{1}{K_n} \frac{1}{\mathcal{P}_{\text{MP}}(z)} r_{\mathbf{k}}(z). \tag{32}$$

Sampling $\mathbf{k}$ from $P(\mathbf{k})$ will give us $z^{\mathbf{k}}$ that are approximate samples from the true posterior $P(z|x)$, however, we have to factorise the distribution $P(\mathbf{k})$ in some way to make computation feasible (since $\mathbf{k}$ can take K^n possible values). If we choose to factorise following the generative graphical model, i.e. via

$$P(\mathbf{k}) = \prod_{i=1}^{n} P(k_i|\mathbf{k}_{\text{pa}(i)}), \tag{33}$$

where $\mathbf{k}_{\text{pa}(i)} = (k_j \; \forall j \in \text{pa}(i))$, then we can continue using the low-rank tensor product approach for computation. Moreover, we can again use a source term trick-type argument using the modified estimate of the marginal likelihood given by

$$\mathcal{P}_{\text{MP}}^{\text{samp}}(z, \mathbf{J}) = \frac{1}{K^n} \sum_{\mathbf{k}\in\mathcal{K}^n} \frac{P(x, z^{\mathbf{k}})}{Q_{\text{MP}}(z^{\mathbf{k}})} e^{J_{k_i, \mathbf{k}_{\text{pa}i}}} \tag{34}$$

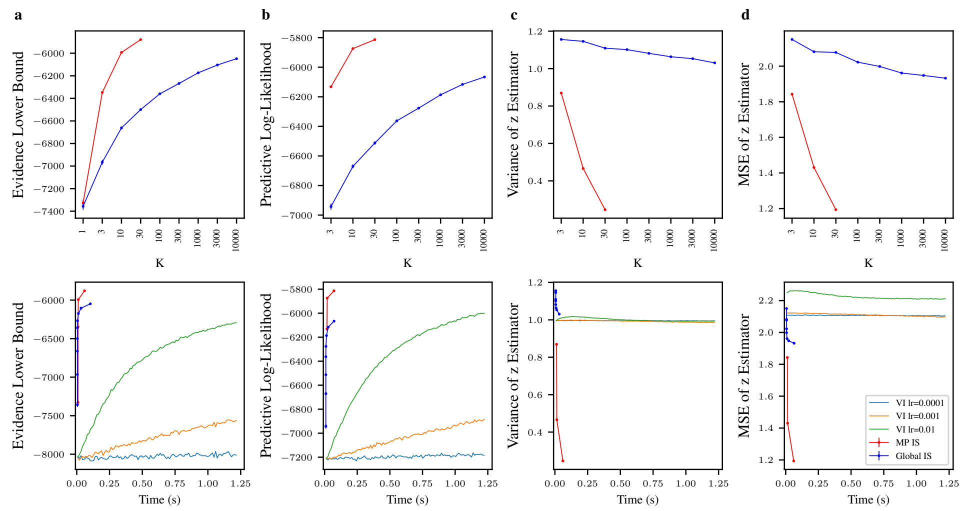

Figure 1: Results obtained in the MovieLens model. Columns **a–c** show the evidence lower bound, predictive log-likelihood and variance in the estimator of the variable $\mathbf{z}_m$ using the true MovieLens100K data. Column **d** shows the mean squared error in the estimator of $\mathbf{z}_m$ when the data is sampled from the model and thus the true value of $\mathbf{z}_m$ is known.

where $\mathbf{J} \in \mathbb{R}^{K^{1+|\text{pa}(i)|}}$ is now a tensor indexed by $k_i \in \mathbb{R}$ and $\mathbf{k}_{\text{pa}(i)} \in \mathbb{R}^{|\text{pa}(i)|}$. Then since

$$\frac{\partial}{\partial J_{k_i}}\bigg|_{\mathbf{J}=\mathbf{0}} e^{J_{k_i,\mathbf{k}_{\text{pa}(i)}}} = \begin{cases} 1 & \text{if } k_i' = k_i \text{ and } \mathbf{k}_{\text{pa}(i)} = \mathbf{k}'_{\text{pa}(i)} \\ 0 & \text{otherwise,} \end{cases} \tag{35}$$

we may follow a similar argument to that taken in Eq. 26–29 and will end up summing in the numerator over all $\mathbf{k}$ except for k_i and $\mathbf{k}_{\text{pa}(i)}$, which we write as $\mathbf{k}/(k_i, \mathbf{k}_{\text{pa}(i)})$, arriving at the result

$$\frac{\partial}{\partial J_{k_i,\mathbf{k}_{\text{pa}i}}}\bigg|_{\mathbf{J}=\mathbf{0}} \log \mathcal{P}_{\text{MP}}^{\text{samp}}(z, \mathbf{J}) = \frac{\frac{1}{K^n} \sum_{\mathbf{k}/(k_i,\mathbf{k}_{\text{pa}i})} r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)} \tag{36}$$

$$= \sum_{\mathbf{k}/(k_i,\mathbf{k}_{\text{pa}i})} P(\mathbf{k}) \tag{37}$$

$$= P(k_i, \mathbf{k}_{\text{pa}(i)}). \tag{38}$$

Finally, from these marginals $P(k_i, \mathbf{k}_{\text{pa}(i)})$ we can compute the conditionals required in Eq. 33 using Bayes' theorem:

$$P(k_i|\mathbf{k}_{\text{pa}(i)}) = \frac{P(k_i, \mathbf{k}_{\text{pa}(i)})}{\sum_{k_i'} P(k_i', \mathbf{k}_{\text{pa}(i)})} \tag{39}$$

2.3 Experiments

The methods described above were evaluated on graphical models for two datasets: Movielens Harper and Konstan [2015], containing 100,000 ratings for a variety of films from various users; and NYC Bus Breakdown, which contains information on school bus journey delay times in New York based on year, borough and bus ID. The full specifications of the models and further experiment details can be found in the paper[3] Bowyer et al. [2023].

We tested samples generated from both the global and massively parallel importance sampling methods where for simplicity we used the prior as our proposal distribution. We also tested against variational inference, where we repeatedly updated our approximate posterior based on single samples (i.e. with $K = 1$, for which global and massively parallel methods are equivalent) by optimising the ELBO. For each method we tracked four metrics: the evidence lower bound of our samples; the

[3]Attached in the appendix.

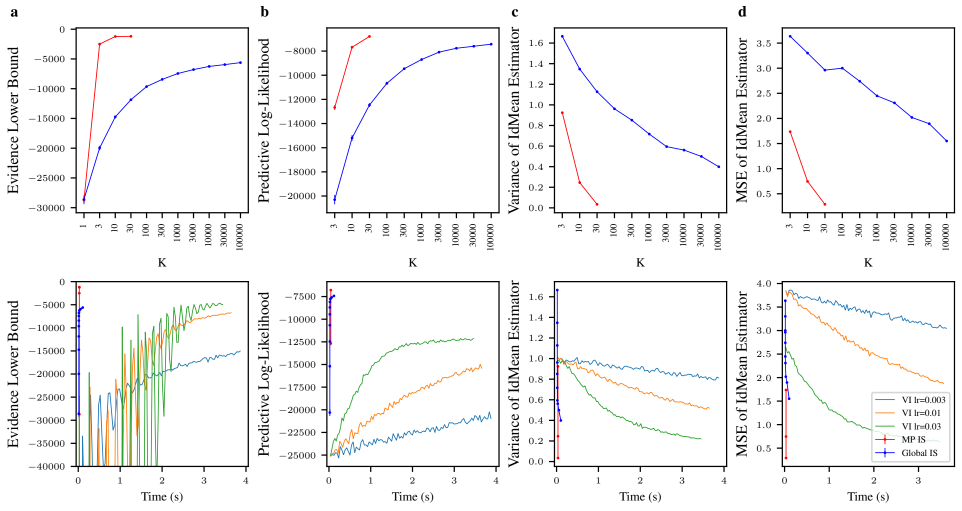

Figure 2: Results obtained in the NYC Bus Breakdown model. Columns **a**–**c** show the evidence lower bound, predictive log-likelihood and variance in the estimator of the variable IdMean_{mj} using the true data. Column **d** shows the mean squared error in the estimator of IdMean_{mj} when the data is sampled from the model.

predictive log-likelihood (evaluated on a test set that was disjoint from the training set); the variance of the posterior mean estimates for one variable in the model; and the mean squared error between our posterior mean estimates and the true variable value when these were known (which was possible by generating a fake dataset using our graphical models).

The results from the Movielens experiment are shown in Fig. 1 and those from the NYC Bus Breakdown experiment are shown in Fig. 2. In both datasets we obtained higher predictive log-likelihoods with our new method in a shorter amount of time when compared to the other methods, suggesting that the massively parallel importance samples were of higher quality. We also found that the new method obtained lower-variance posterior mean estimates than the other methods in a shorter amount of time when using the real data, and saw the same pattern when looking at the MSE of the posterior mean estimates generated from fake data. This indicates that tighter and better posterior mean estimates can be obtained from our massively parallel approach.

3 Massively Parallel Markov Chain Monte Carlo

3.1 Background

Markov chain Monte Carlo methods are an extremely popular and important set of tools in modern statistics Diaconis [2008], however, they can often be very computationally expensive, particularly due to the inherent sequential nature of Markov chains. Because of this, there exist a wide range of algorithms designed to improve the efficiency of MCMC, including some that make use of parallel computing. A simple approach, such as that discussed in Rosenthal [2000], involves running several Markov chains independently in parallel and combining results after computation, whilst slightly more complex methods involve sharing information across these parallel chains in order to improve the (adaptive) proposal distribution being used Craiu et al. [2009], Solonen et al. [2012]. However, we can also run a single chain on which we sequentially make multiple proposals in parallel, say K proposals, either only accepting one of these proposals at each step Liu et al. [2000], Neal [2011] or using all K proposals at each step to generate the proposals at the next step Calderhead [2014]. This has the benefit that in high-dimension state-spaces we're more likely to obtain some proposals for which the latent variables have high joint probability under the posterior if we can use a large enough K. However, often K will be far too small for this to be of much use. That is, unless we consider all K^n combinations of K samples of the n latent variables involved, in which case we might indeed find some combinations that have high joint probability under the posterior—suggesting we might be able to use our massively parallel framework in the context of MCMC.

3.2 Methods

We employ a slightly different setup to the previous section in establishing our MCMC method. First, let us denote a tuple containing K samples of our ith latent variable by

$$z_i = (z_i^1, ..., z_i^K) \in \mathcal{Z}_i^K, \tag{40}$$

where we may say the jth sample of the ith latent variable is $z_i^j \in \mathcal{Z}_i$. Then the full collection of K samples of each of our n latent variables is given by

$$z = (z_1, ..., z_n) \in \mathcal{Z}_1^K \times \cdots \times \mathcal{Z}_n^K = Z^K. \tag{41}$$

Now as in Eq. 9, for some indices $\mathbf{k} = (k_1, ..., k_n) \in \mathcal{K}^n$, we write the combination of samples using these indices as

$$z^{\mathbf{k}} = \left(z_1^{k_1}, z_2^{k_2}, \ldots, z_n^{k_n}\right) \in \mathcal{Z}, \tag{42}$$

which we now refer to as our *indexed samples*. Likewise, we refer to the collection of samples *not* corresponding to the indices $\mathbf{k}$ as *unindexed samples*, $z^{/\mathbf{k}}$ where

$$z^{/\mathbf{k}} = (z_1^{/k_1}, ..., z_n^{/k_n}) \in \mathcal{Z}^{K-1}, \tag{43}$$

$$z_i^{/k_i} = (z_i^1, ..., z_i^{k_i-1}, z_i^{k_i+1}, ..., z_i^K) \in \mathcal{Z}_i^{K-1}. \tag{44}$$

Now for our data, x, and a single set of samples for each latent variable, i.e. some $z' = (z_1', ..., z_n') \in \mathcal{Z}$, we can use the graphical model from our generative distribution to see that

$$P(x, z') = P(x|z'_{\text{pa}(x)}) \prod_{i=1}^{n} P(z_i'|z_{\text{pa}(i)}) \tag{45}$$

where $z'_{\text{pa}(i)}$ is the collection of all parent latent variables of variable i. Our indexed samples $z^{\mathbf{k}}$ could take the place of this generic $z' \in \mathcal{Z}$, however, in doing so we must extend our state space to capture the distribution of $\mathbf{k}$, where the full joint distribution of data x, samples $z \in \mathcal{Z}^K$ and indices $\mathbf{k} \in \mathcal{K}^n$ factorises as

$$P(x, z, \mathbf{k}) = P(x, z|\mathbf{k})P(\mathbf{k}). \tag{46}$$

This is the main step in which we are taking inspiration from Calderhead [2014], where the state space is also extended using a discrete auxiliary variable. However, their auxiliary variable is only integer valued, unlike our $\mathbf{k} \in \mathcal{K}^n$, since they do not consider different combinations of samples of the latent variables between each of their K proposals.

We put a uniform prior on each $k_i \in \mathcal{K}$ so that we can sample easily and obtain an initial $\mathbf{k}$, from which point we can generate indexed latent variable samples z^k and data x from Eq. 45 using $z^{\mathbf{k}}$ as our z' and following the dependency structure given through the generative graphical model by pa(x) and pa(i). Once we've generated the indexed samples $z^{\mathbf{k}} \in \mathcal{Z}$ within $z \in \mathcal{Z}^K$, to generate the remaining unindexed samples $z^{/\mathbf{k}} \in \mathcal{Z}^{K-1}$ we use the probability density

$$P(x, z|\mathbf{k}) = P(x, z^{\mathbf{k}}) \prod_{i=1}^{n} q(z_i^{/k_i}; x, z^{\mathbf{k}}, z_{\text{qa}(i)}^{/\mathbf{k}}) \tag{47}$$

based on the proposal $q(z_i^{/k_i}; x, z^{\mathbf{k}}, z_{\text{qa}(i)}^{/\mathbf{k}})$, where $z_{\text{qa}(i)}^{/\mathbf{k}}$ denotes the unindexed samples of all latent variables that are direct parents of $z_i^{k_i}$ under q. It is important to note that the conditional dependencies in q need not necessarily be the same as those in the generative graphical model, i.e. it is possible, and may even be desirable, to have $qa(i) \neq pa(i)$. This becomes clearer if we now explicitly state how our MCMC method will work once we've got past our initial generation of $\mathbf{k}$, $z^{\mathbf{k}}$ and $z^{/\mathbf{k}}$ as just described.

Specifically, in each iteration of our method we want to generate new values for $\mathbf{k}$, $z^{\mathbf{k}}$ and $z^{/\mathbf{k}}$ via Gibbs sampling. Then by combining each iteration's indexed and unindexed samples, $z^{\mathbf{k}} \in \mathcal{Z}$ and $z^{/\mathbf{k}} \in \mathcal{Z}^{K-1}$, we can obtain a full collection of samples $z \in \mathcal{Z}^K$ from every iteration.

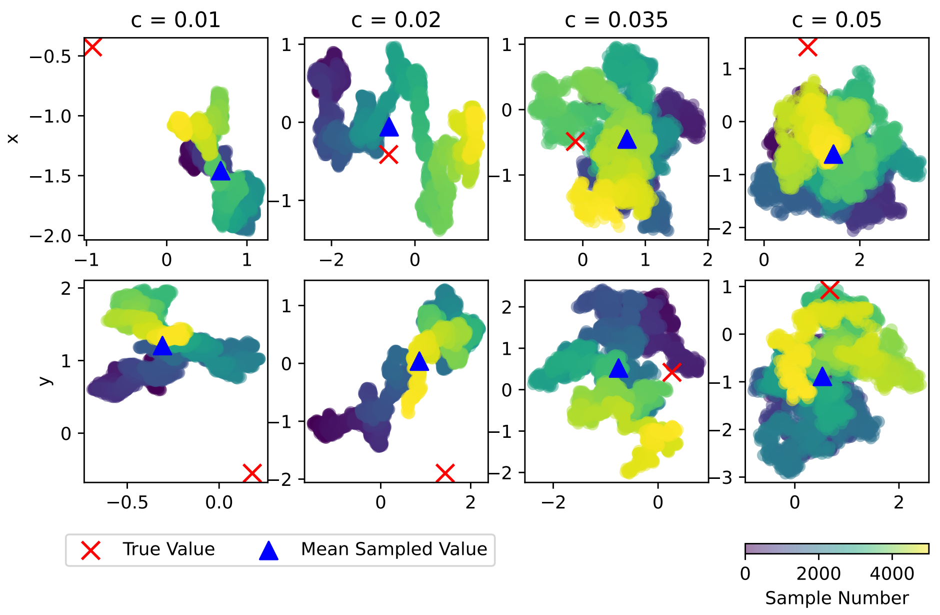

Figure 3: Trace plots of the indexed samples $z^{\mathbf{k}}$ for both x (top row) and y (bottom row) over 5000 iterations with iteration number shown via colour and different values of c used in each of the columns.

Generating new unindexed samples $z^{/\mathbf{k}}$ given $z^{\mathbf{k}}$ can be done straightforwardly by sampling from $\prod_{i=1}^{n} q(z_i^{/k_i}; x, z^{\mathbf{k}}, z_{\text{qa}(i)}^{/\mathbf{k}})$ (as long as we do it in accordance with the dependency structure given by each qa(i)). Similarly, once we have a new $\mathbf{k}$ we can use Eq. 45 for $z' = z^{\mathbf{k}}$ to obtain new indexed samples $z^{\mathbf{k}}$. However, it is the generation of new indices $\mathbf{k}$ given values for $z^{\mathbf{k}}$ and $z^{/\mathbf{k}}$ that requires some more thought.

We can sample $\mathbf{k}$ using

$$P(\mathbf{k}|z, x) \propto P(x, z^{\mathbf{k}}) \prod_{i=1}^{n} q(z_i^{/k_i}; x, z^{\mathbf{k}}, z_{\text{qa}(i)}^{/\mathbf{k}}), \tag{48}$$

but in order to simplify this we choose proposals which are independent of all other variables, i.e. for all $i \in \{1, ..., n\}$ we have

$$q(z_i^{/k_i}; x, z^{\mathbf{k}}, z_{\text{qa}(i)}^{/\mathbf{k}}) = q(z_i^{/k_i}; z_i^{k_i}). \tag{49}$$

We further restrict our proposals to have some symmetry with respect to k_i, in that for any $k_i' \neq k_i$ we have

$$q(z_i^{/k_i}; z_i^{k_i}) = q(z_i^{/k_i'}; z_i^{k_i'}). \tag{50}$$

We do this to ultimately arrive at the conditional distribution for Gibbs sampling a new $\mathbf{k}$ being proportional the underlying model,

$$P(\mathbf{k}|z, x) \propto P(x, z^{\mathbf{k}}). \tag{51}$$

Finally since $P(x, z^{\mathbf{k}})$ can be computed via Eq. 45, which is equivalent to Eq. 10 (but written in a more MCMC-friendly way), this final step can be done using the efficient tensor operations mentioned in Section 2.1.

3.3 Toy Experiment

We perform one toy experiment using the massively parallel MCMC method on the following model:

$$\mathbf{x}, \mathbf{y} \sim \mathcal{N}(\mathbf{0}_2, \mathbf{I}_2) \tag{52}$$

$$\mathbf{d} \sim \mathcal{N}(\mathbf{x}, \exp(\mathbf{y})\mathbf{I}_2). \tag{53}$$

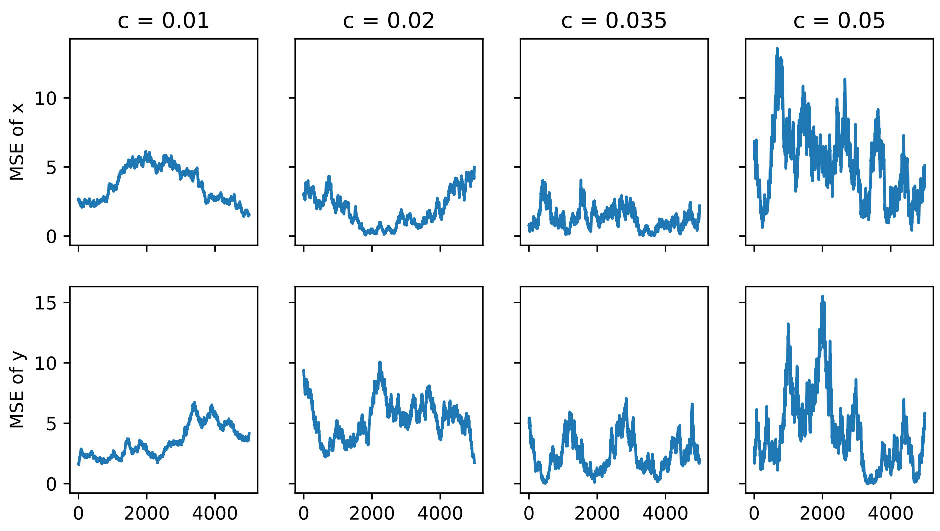

Figure 4: Mean squared error of the indexed samples $z^{\mathbf{k}}$ for both x (top row) and y (bottom row) over 5000 iterations compared to the true underlying values, with different values of c used in each of the columns.

We do this for $K = 2$ with proposal distributions given by

$$q(z_i^{/k_i}; z_i^{k_i}) = \mathcal{N}(z_i^{k_i}, c\mathbf{I}) \tag{54}$$

for $c \in \{0.01, 0.02, 0.035, 0.05\}$. We note here that these values of c have been chosen in order to present a variety of behaviour from our method, and in particular we found that—as is common in MCMC methods—the proposal distribution was very sensitive to changes and in general difficult to tune. As will be justified below, we found that the value $c = 0.035$ seemed to produce the best results out of all values for c that were tried.

To obtain our results we first sampled from our model to obtain data $\mathbf{d}$, using which we ran the massively parallel MCMC method for 5000 iterations. For simplicity, we report results obtained from the indexed samples $z^{\mathbf{k}}$ in each iteration, which can be see in the 2D trace plot presented in Fig. 3. Looking at the leftmost column we see that proposal distribution with $c = 0.01$ is not wide enough for the samples to effectively move towards the true value of x or y in just 5000 iterations. With enough time we would expect see the sample means of our variables eventually move toward the true values, however, for this experiment the mixing time of this chain is simply too high. It is important to note that focusing on the closeness of the sample means to the true values is not enough when interpreting these trace plots, otherwise we may be led to believe that using $c = 0.02$ gives us a proposal distribution that works well for generating samples of $\mathbf{x}$, when in actual fact the samples that were produced don't give us the Gaussian shape we'd expect from the posterior, and furthermore the corresponding trace plot for $\mathbf{y}$ under $c = 0.02$ not only lacks this Gaussian look but does still have a sample mean that is far from the true value of $\mathbf{y}$. This also goes to show a weakness in this small experiment: we aren't averaging over many runs and so our results are vulnerable to high-variance behaviour.

We see better results in the $c = 0.035$ column, with a general shape that looks slightly more Gaussian than in the previous two columns and comparatively smaller distances between the sample means and true values of $\mathbf{x}$ and $\mathbf{y}$. Increasing c further to 0.05, however, leads to new problems: the proposal distribution is now too wide to effectively capture the posterior, meaning that whilst we do get Gaussian-looking trace plots, the samples haven't really moved toward the true values of $\mathbf{x}$ and $\mathbf{y}$. The over-wideness of this proposal distribution can also be seen in Fig. 4, in which we have plotted the mean squared error between each of the 5000 samples of $\mathbf{x}$ and $\mathbf{y}$ and their true underlying values. The $c = 0.05$ plot shows by far the highest MSE values as the samples move across the proposal distribution without effectively modelling the posterior. Meanwhile, for the other values of c we see much smaller MSEs, with arguably the lowest consistent values occuring in the $c = 0.035$

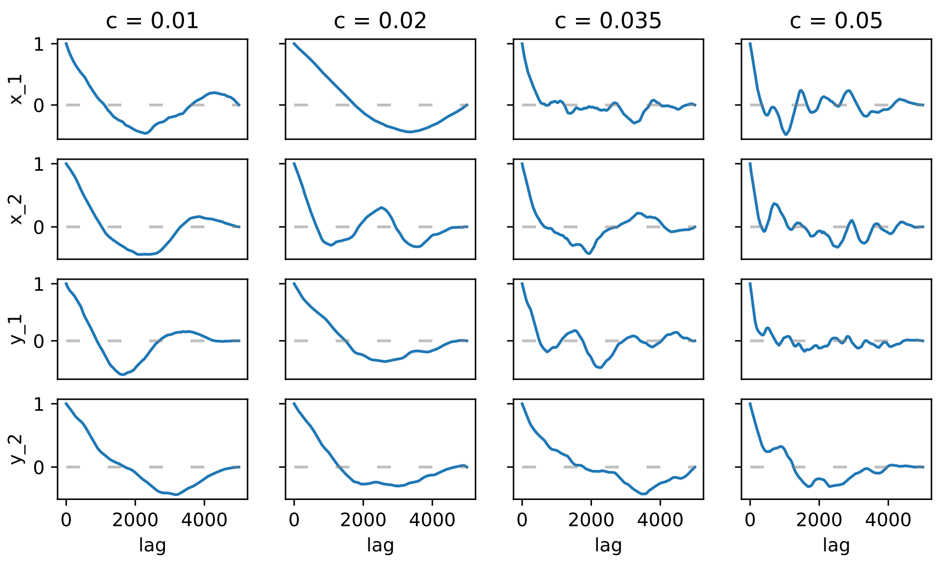

Figure 5: Autocorrelation of the indexed samples $z^{\mathbf{k}}$ for the two dimensions of x (top two rows) and the two dimensions of y (bottom two rows) over 5000 iterations with different values of c used in each of the columns.

column—although, again we should avoid reading into these particular results too much since the scale of the experiment is so small.

The final plot we present shows the autocorrelation in each dimension of both x and y. Whilst the $c = 0.05$ plots in the rightmost column seem to stay closest to the zero-line overall, we know that this is because the samples involved are effectively just coming from a wide Gaussian distribution, whereas the $c = 0.01$ and $c = 0.02$ plots don't fluctuate around the zero-line very much because their posteriors not wide enough. Although the autocorrelation plots in the third column are far from ideal they do help confirm the idea from Fig. 3 and Fig. 4 that $c = 0.035$ gives us the best proposal distribution out of the four we've looked at.

This toy experiment is by no means a comprehensive evaluation of the massively parallel MCMC method, which would require a much more thorough consideration of the algorithm's performance with a wider range of proposal distributions and ideally in a wide variety of more complex models. However, we have successfully shown that we can indeed run the very basic $K = 2$ method and obtain sensible results, suggesting that it is worthwhile to continue researching implementation of the method.

4 Future Work

The above introduction to massively parallel MCMC relies on a symmetry in generating proposals that comes about when we have only one unindexed proposal to generate at each iteration, i.e. with $K = 2$. In order to fully examine the potential for this method, we need to extend it to the $K > 2$ case, which will require looking further into how our proposal distributions must work in order to maintain this symmetry. (And in general, it would be good to gain some idea of what sort of proposals will behave well in this setting—perhaps adaptive proposals Haario et al. [2001] may be of use here).

On a larger scale, the massively parallel framework has the potential to be very wide-reaching, as it should be applicable to most probabilistic inference tasks, hopefully resulting in more efficient methods that are able to effectively use parallel GPU computing. Because of this, a long term goal is to develop a probabilistic programming language, similar to STAN Carpenter et al. [2017] or Turing Ge et al. [2018], with a massively parallel-based implementation.

5 Conclusion

In this mini-project we have introduced the massively parallel approach to Bayesian inference through importance weighting and sampling, and briefly discussed the results from Bowyer et al. [2023] which show the approach's success. Furthermore, we have introduced a massively parallel MCMC method that makes use of similar ideas, implemented a small toy example, and discussed possible avenues for future work under this framework.

References

Laurence Aitchison. Tensor Monte Carlo: particle methods for the GPU era, January 2019. URL `http://arxiv.org/abs/1806.08593`. arXiv:1806.08593 [cs, stat].

Jörg Bornschein and Yoshua Bengio. Reweighted Wake-Sleep, April 2015. URL `http://arxiv.org/abs/1406.2751`. arXiv:1406.2751 [cs].

Sam Bowyer, Thomas Heap, and Laurence Aitchison. Bayesian inference with massively parallel importance weighting. *Submitted to UAI 2023, and included in the Appendix*, 2023.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance Weighted Autoencoders, November 2016. URL `http://arxiv.org/abs/1509.00519`. arXiv:1509.00519 [cs, stat].

Ben Calderhead. A general construction for parallelizing MetropolisHastings algorithms. *Proceedings of the National Academy of Sciences*, 111(49):17408–17413, December 2014. doi: 10.1073/pnas.1408184111. URL `https://www.pnas.org/doi/10.1073/pnas.1408184111`. Publisher: Proceedings of the National Academy of Sciences.

Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1), 2017. Publisher: Columbia Univ., New York, NY (United States); Harvard Univ., Cambridge, MA (United States).

Sourav Chatterjee and Persi Diaconis. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2), April 2018. ISSN 1050-5164. doi: 10.1214/17-AAP1326. URL `https://projecteuclid.org/journals/annals-of-applied-probability/volume-28/issue-2/The-sample-size-required-in-importance-sampling/10.1214/17-AAP1326.full`.

Radu V. Craiu, Jeffrey Rosenthal, and Chao Yang. Learn From Thy Neighbor: Parallel-Chain and Regional Adaptive MCMC. *Journal of the American Statistical Association*, 104(488): 1454–1466, December 2009. ISSN 0162-1459. doi: 10.1198/jasa.2009.tm08393. URL `https://doi.org/10.1198/jasa.2009.tm08393`. Publisher: Taylor & Francis _eprint: https://doi.org/10.1198/jasa.2009.tm08393.

G Daniel, Johnnie Gray, et al. Opt_einsum-a python package for optimizing contraction order for einsum-like expressions. *Journal of Open Source Software*, 3(26):753, 2018.

Persi Diaconis. The Markov chain Monte Carlo revolution. *Bulletin of the American Mathematical Society*, 46(2):179–205, November 2008. ISSN 0273-0979. doi: 10.1090/S0273-0979-08-01238-X. URL `http://www.ams.org/journal-getitem?pii=S0273-0979-08-01238-X`.

Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: A Language for Flexible Probabilistic Inference. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 1682–1690. PMLR, March 2018. URL `https://proceedings.mlr.press/v84/ge18b.html`. ISSN: 2640-3498.

Tomas Geffner and Justin Domke. Variational Inference with Locally Enhanced Bounds for Hierarchical Models, July 2022. URL `http://arxiv.org/abs/2203.04432`. arXiv:2203.04432 [cs, stat].

Heikki Haario, Eero Saksman, and Johanna Tamminen. An Adaptive Metropolis Algorithm. *Bernoulli*, 7(2):223–242, 2001. ISSN 1350-7265. doi: 10.2307/3318737. URL `https://www.jstor.org/stable/3318737`. Publisher: International Statistical Institute (ISI) and Bernoulli Society for Mathematical Statistics and Probability.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Thomas Heap and Laurence Aitchison. Massively parallel reweighted wake-sleep. *Submitted to UAI 2023*, 2023.

Geoffrey E. Hinton, Peter Dayan, Brendan J. Frey, and Radford M. Neal. The "Wake-Sleep" Algorithm for Unsupervised Neural Networks. *Science*, 268(5214):1158–1161, May 1995. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.7761831. URL `https://www.science.org/doi/10.1126/science.7761831`.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, 2014. URL `http://arxiv.org/abs/1312.6114`. arXiv:1312.6114 [cs, stat].

Jun S. Liu, Faming Liang, and Wing Hung Wong. The Multiple-Try Method and Local Optimization in Metropolis Sampling. *Journal of the American Statistical Association*, 95(449):121–134, 2000. ISSN 0162-1459. doi: 10.2307/2669532. URL `https://www.jstor.org/stable/2669532`. Publisher: [American Statistical Association, Taylor & Francis, Ltd.].

Radford M. Neal. MCMC Using Ensembles of States for Problems with Fast and Slow Variables such as Gaussian Process Regression, January 2011. URL `http://arxiv.org/abs/1101.0387`. arXiv:1101.0387 [stat].

Fritz Obermeyer, Eli Bingham, Martin Jankowiak, Neeraj Pradhan, Justin Chiu, Alexander Rush, and Noah Goodman. Tensor variable elimination for plated factor graphs. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4871–4880. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/obermeyer19a.html`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Jeffrey S Rosenthal. Parallel computing and monte carlo algorithms. *Far east journal of theoretical statistics*, 4(2):207–236, 2000.

Antti Solonen, Pirkka Ollinaho, Marko Laine, Heikki Haario, Johanna Tamminen, and Heikki Järvinen. Efficient MCMC for Climate Model Parameter Estimation: Parallel Adaptive Chains and Early Rejection. *Bayesian Analysis*, 7(3):715–736, September 2012. ISSN 1936-0975, 1931-6690. doi: 10.1214/12-BA724. URL `https://projecteuclid.org/journals/bayesian-analysis/volume-7/issue-3/Efficient-MCMC-for-Climate-Model-Parameter-Estimation--Parallel-Adaptive/10.1214/12-BA724.full`. Publisher: International Society for Bayesian Analysis.

Massively Parallel Importance Weighting and Sampling

Sam Bowyer
School of Mathematics
University of Bristol
Bristol

Thomas Heap
Department of Computer Science
University of Bristol
Bristol

Laurence Aitchison
Department of Computer Science
University of Bristol
Bristol
`laurence.aitchison@brisol.ac.uk`

Abstract

Importance sampling is a popular technique in Bayesian inference: by reweighting samples drawn from a proposal distribution we are able to obtain samples and moment estimates from a Bayesian posterior over some n latent variables. Recent work, however, indicates that importance sampling scales poorly — in order to accurately approximate the true posterior, the required number of importance samples grows is exponential in the number of latent variables [Chatterjee and Diaconis, 2018]. We work around this issue by drawing K samples for each of the n latent variables and reasoning about all K^n combinations of latent samples. In principle, we reason efficiently over K^n combinations of samples by exploiting conditional independencies in the generative model. However, in practice this requires complex algorithms. Instead, we exploit a trick from physics to compute all the required quantities — posterior expectations, marginals and samples — by differentiating through a marginal likelihood estimator.

1 Introduction

Importance weighting allows us to reweight samples drawn from a proposal in order to compute expectations of a different distribution, such as a Bayesian posterior. However, importance weighting breaks down in larger models. Chatterjee and Diaconis [2018] showed that the number of samples required to accurately approximate the true posterior scales as $\exp\left(D_{\text{KL}}\left(\text{P}\left(z|x\right)||\text{Q}\left(z\right)\right)\right)$, where $\text{P}\left(z|x\right)$ is the true posterior over latent variables, z, given data x, and $\text{Q}\left(z\right)$ is the proposal. Problematically, we expect the KL divergence to scale with n, the number of latent variables. Indeed, if $\text{P}\left(z|x\right)$ and $\text{Q}\left(z\right)$ are IID with n variables, then the KL-divergence is exactly proportional to n. Thus, we expect the required number of importance samples to be exponential in the number of latent variables, and hence we expect accurate importance sampling to be intractable in larger models.

To resolve this issue, we introduce a massively parallel importance sampling scheme that in effect uses an exponential number of samples to compute posterior expectations, marginals and samples. We do this by drawing K samples of each of the n latent variables from the proposal, then individually reweighting all K^n combinations of all samples of all latent variables. While reasoning about all K^n combinations of samples might seem intractable, we should in principle be able to perform efficient computations by exploiting conditional independencies in the underlying graphical model.

Preprint. Under review.

Of course, many computations that are possible in principle are extremely complex in practice, and that turns out to be the case here.

We then noticed that we could perform this reasoning over K^n latent variables using methods from the discrete graphical model literature. However, this turned out to be less helpful than expected because these algorithms are highly complex, and different for computing posterior expectations, marginals and samples. We therefore develop a much simpler approach to computing posterior expectations, marginals and samples, which entirely avoids the need to explicitly write backwards computations. In particular, we show that posterior expectations, marginals and samples can be obtained simply by differentiating through (a slightly modified) forward computation that produces an estimate of the marginal likelihood. The required gradients can be computed straightforwardly using modern autodiff, and the resulting implicit backward computations automatically inherit potentially complex optimizations from the forward pass.

2 Related work

There is a considerable body of work in the discrete graphical model setting that computes posterior expectations, marginals and samples [Dawid, 1992, Pfeffer, 2005, Bidyuk and Dechter, 2007, Geldenhuys et al., 2012, Claret et al., 2013, Sankaranarayanan et al., 2013, Goodman and Stuhlmüller, 2014, Gehr et al., 2016, Narayanan et al., 2016, Albarghouthi et al., 2017, Wang et al., 2018, Obermeyer et al., 2019, Holtzen et al., 2020]. Our work differs in two respects. First, our massively parallel methods are not restricted to discrete graphical models, but can operate with arbitrary continuous latent variables. Second, this work involves complex implementations that — in one sense or another — "proceed by recording an adjoint compute graph alongside the forward computation and then traversing the adjoint graph backwards starting from the final result of the forward computation" [Obermeyer et al., 2019]. The forward computation is reasonably straightforward: it is just a big tensor product that can be computed efficiently using pre-existing libraries such as opt-einsum, and results in (an estimate of) the marginal likelihood. However, the backward traversal is much more complex, if for no other reason than the need to implement separate traversals for each operation of interest (computing posterior expectations, marginals and samples). Additionally, these traversals need to correctly handle all special cases (including e.g. optimized implementations of plates and timeseries). Importantly, the optimized forward is often relatively straightforward to implement, while the backward traversal is far more complex. For instance, the forward computation for a timeseries involves a product of T matrices arranged in a chain. Naively computing this product on GPUs is very slow, as it requires T separate matrix multiplications. However, it is possible to massively optimize this forward computation, converting $\mathcal{O}(T)$ to $\mathcal{O}(\log(T))$ tensor operations by multiplying adjacent pairs of matrices in a single batched matrix multiplication operation. This optimization is straightforward in the forward computation. However, applying this optimization as part of the backward computation is far more complex (see Corenflos et al., 2022 for details). This complexity (along with similar complexity for other important optimizations such as plates) was prohibitive for our small team (which ultimately aims to implement a new massively parallel probabilistic programming language). In contrast, we provide a much simpler approach, where we only explicitly perform the forward computation; we compute the posterior expectations, marginals and samples by differentiating through that forward computation.

There is work on fitting importance weighted autoencoders [Burda et al., 2015, IWAE;] and reweighted wake-sleep [RWS Bornschein and Bengio, 2014, Le et al., 2020] in the massively parallel setting [Aitchison, 2019, Geffner and Domke, 2022, Anonymous, 2023] for general probabilistic models (i.e. with continuous latent variables). However, this work only provides methods for performing massively parallel updates to approximate posteriors (e.g. by optimizing a massively parallel ELBO). This work does not provide a method to individually reweight the samples to provide accurate posterior expectations, marginals and samples. Instead, this previous work simply takes the learned approximate posterior as an estimate of the true posterior, and does not attempt to correct for inevitable biases.

Massively parallel importance weighting for timeseries have some similarities to particle filtering/SMC methods [Gordon et al., 1993, Doucet et al., 2009, Andrieu et al., 2010, Maddison et al., 2017, Le et al., 2017, Lindsten et al., 2017, Naesseth et al., 2018, Lai et al., 2022]. However, our massively parallel methods allow a very general class of proposal distributions in a very general class of probabilistic models. Additionally, we develop a new approach to computing posterior expectations, marginals and samples by differentiating through a slightly modified marginal likelihood estimator.

3 Background

Bayesian inference. In Bayesian inference, we have a prior, $\mathrm{P}\left(z'\right)$ over latent variables, z', and a likelihood, $\mathrm{P}\left(x|z'\right)$ connecting the latents to the data, x. Here, we use z' rather than z because we reserve z for future use as a collection of K samples (Eq. 3). Our goal is to compute the posterior distribution over latent variables conditioned on observed data,

$$\mathrm{P}\left(z'|x\right) = \frac{\mathrm{P}\left(x|z'\right)\mathrm{P}\left(z'\right)}{\sum_{z''}\mathrm{P}\left(x, z''\right)}, \tag{1}$$

We often seek to obtain samples from the posterior or to compute posterior expectations,

$$m_{\text{post}} = \sum_{z'} \mathrm{P}\left(z'|x\right) m(z') \tag{2}$$

However, the true posterior moment is usually intractable, so instead we are forced to use an alternative method such as importance weighting.

Importance weighting. In importance weighting, we draw a collection of K samples from the full joint state space. This collection of K samples is denoted $z \in \mathcal{Z}$, with a single sample denoted z^k,

$$z = (z^1, z^2, \dots, z^K) \in \mathcal{Z}^K. \tag{3}$$

The collection of K samples, z, is drawn by sampling K times from the proposal,

$$\mathrm{Q}\left(z\right) = \prod_{k\in\mathcal{K}} \mathrm{Q}\left(z^k\right), \tag{4}$$

where $\mathcal{K}$ is the set of possible indices, $\mathcal{K} = \{1, \dots, K\}$. As the true posterior moment is usually intractable, one approach is to use a self-normalized importance sampling estimate, $m_{\text{global}}(z)$. We call this a "global" importance weighted estimate following terminology in [Geffner and Domke, 2022] and in contrast with the massively parallel methods that we define later,

$$m_{\text{global}}(z) = \tfrac{1}{K} \sum_{k\in\mathcal{K}} \frac{r_k(z)}{\mathcal{P}_{\text{global}}(z)} m(z^k) \tag{5}$$

where,

$$r_k(z) = \frac{\mathrm{P}\left(x, z^k\right)}{\mathrm{Q}\left(z^k\right)} \tag{6}$$

$$\mathcal{P}_{\text{global}}(z) = \tfrac{1}{K} \sum_{k\in\mathcal{K}} r_k(z) \tag{7}$$

with samples z drawn from the proposal (Eq. 4). Here, $r_k(z)$ is the ratio of the generative and proposal probabilities, and $\mathcal{P}_{\text{global}}(z)$ is an unbiased estimator of the marginal likelihood,

$$\begin{aligned} \mathrm{E}_{\mathrm{Q}(z)}\left[\mathcal{P}_{\text{global}}(z)\right] &= \mathrm{E}_{\mathrm{Q}(z^k)}\left[\frac{\mathrm{P}\left(x, z^k\right)}{\mathrm{Q}\left(z^k\right)}\right] \\ &= \sum_{z^k} \mathrm{P}\left(x, z^k\right) = \mathrm{P}\left(x\right) \end{aligned} \tag{8}$$

The first equality arises because $\mathcal{P}_{\text{global}}(z)$ is the average of K IID terms, $\mathrm{P}\left(x, z^k\right)/\mathrm{Q}\left(z^k\right)$, so is equal to the expectation of a single term, and the second equality arises if we write the expectation as a sum.

Source term trick. Here, we outline a standard trick from physics that can be used to compute expectations of arbitrary probability distribution by differentiating a modified log-normalizing constant. This trick is used frequently in Quantum Field Theory, for instance [Weinberg, 1995] (Chapter

16), and also turns up in the theory of neural networks [Zavatone-Veth et al., 2021]. But the trick is simple enough that we can give a self-contained introduction here.

In our context, Bayes theorem (Eq. 1) defines an unnormalized density, $\mathrm{P}\left(z|x\right) \propto \mathrm{P}\left(x, z\right)$, with normalizing constant, $\sum_{z'} \mathrm{P}\left(x, z'\right)$. Of course, the normalizing constant is usually intractable, but in our massively parallel context, we can often obtain a reasonable estimate of the normalizing constant, and that will be sufficient. It turns out that we can compute posterior expectations using a slightly modified normalizing constant,

$$Z_m(J) = \sum_{z'} \mathrm{P}\left(x, z'\right) e^{Jm(z')}. \tag{9}$$

where $e^{Jm(z')}$ is known as a source term. Note that setting J to zero recovers the usual normalizing constant,

$$Z_m(J = 0) = \sum_{z'} \mathrm{P}\left(x, z'\right). \tag{10}$$

Now, we can extract the posterior moment by evaluating the gradient of $\log Z_m(J)$ at $J = 0$,

$$\left.\frac{\partial}{\partial J}\right|_{J=0} \log Z_m(J) = \left.\frac{\partial}{\partial J}\right|_{J=0} \log \sum_{z'} \mathrm{P}\left(x, z'\right) e^{Jm(z')}. \tag{11}$$

Differentiating the logarithm at $J = 0$,

$$\left.\frac{\partial}{\partial J}\right|_{J=0} \log Z_m(J) = \frac{\sum_{z'} \mathrm{P}\left(x, z'\right) \left.\frac{\partial}{\partial J}\right|_{J=0} e^{Jm(z')}}{\sum_{z''} \mathrm{P}\left(x, z''\right)}. \tag{12}$$

Differentiating the exponential at $J = 0$,

$$\left.\frac{\partial}{\partial J}\right|_{J=0} \log Z_m(J) = \sum_{z'} \frac{\mathrm{P}\left(x, z'\right)}{\sum_{z''} \mathrm{P}\left(x, z''\right)} m(z'). \tag{13}$$

and identifying the posterior using Bayes theorem (Eq. 1).

$$\left.\frac{\partial}{\partial J}\right|_{J=0} \log Z_m(J) = \sum_{z'} \mathrm{P}\left(z'|x\right) m(z) = m_{\text{post}} \tag{14}$$

This is exactly the form for the posterior moment in Eq. (2).

Massively parallel marginal likelihood estimators To get an accurate marginal likelihood estimator, we introduce a massively parallel estimator which individually weights all K^n combinations of K samples on n latent variables. We write the kth sample of the ith latent variable as z_i^k, and we can then write the collection of all K samples of the ith latent variable as,

$$z_i = (z_i^1, z_i^2, \dots, z_i^K). \tag{15}$$

And z is the collection of all K samples of all n latent variables, (as in Eq. 3),

$$z = (z_1, z_2, \dots, z_n). \tag{16}$$

As we have multiple latent variables, our massively-parallel proposals have a graphical model structure,

$$\mathrm{Q}_{\text{MP}}\left(z\right) = \prod_{i=1}^{n} \mathrm{Q}_{\text{MP}}\left(z_i | z_j \text{ for } j \in \text{qa}\left(i\right)\right), \tag{17}$$

where qa (i) is the set of indices of parents of z_i under that graphical model. Note that this form implies that z_i may depends on all samples of the parent latent variables, and that the K samples of z_i, namely $z_i^1, \dots, z_i^K$, may have some dependencies. Usually, we derive the massively-parallel proposal from an underlying, user-specified, single-sample proposal. The simplest approach is just to draw K samples independently from the full joint latent space. Other alternatives are available, including to use a uniform mixture / permutation over samples of parent latent variables (see [Anonymous, 2023] for further details). The key property of these proposals is that the single-sample marginals, $\text{Q}_{\text{MP}}\left(z_i^k \middle| z_j \text{ for } j \in \text{qa}\left(i\right)\right)$ are easily computable. In all our experiments, we will use a permutation over samples of the parent latent variables.

For the generative model, we need to explicitly consider all K^n combinations of K samples on n latent variables. To help us write down these combinations, we define a vector of indices,

$$\mathbf{k} = (k_1, k_2, \dots, k_n) \in \mathcal{K}^n, \tag{18}$$

with one index, k_i for each latent variable, z_i. Thus, we can write the "indexed" latent variables as,

$$z^{\mathbf{k}} = \left(z_1^{k_1}, z_2^{k_2}, \dots, z_n^{k_n}\right) \in \mathcal{Z}, \tag{19}$$

which represents a single sample from the full joint latent space. The generative model also has graphical model structure, with the set of indices of parents of the ith latent variable under the generative model begin denoted pa (i) (contrast this with qa (i) which is the parents of the ith latent variable under the proposal).

Thus, the generative probability for a single combination of samples, denoted $z^{\mathbf{k}}$, can be written as,

$$\begin{aligned} \text{P}\left(x, z^{\mathbf{k}}\right) = &\text{P}\left(x \middle| z_j^{k_j} \text{ for all } j \in \text{pa}\left(x\right)\right) \\ &\prod_{i=1}^{n} \text{P}\left(z_i^{k_i} \middle| z_j^{k_j} \text{ for all } j \in \text{pa}\left(i\right)\right). \end{aligned} \tag{20}$$

Thus, we can write a massively parallel marginal likelihood estimator as,

$$r_{\mathbf{k}}(z) = \frac{\text{P}\left(x, z^{\mathbf{k}}\right)}{\prod_i \text{Q}_{\text{MP}}\left(z_i^{k_i} \middle| z_j \text{ for } j \in \text{qa}\left(i\right)\right)} \tag{21}$$

$$\mathcal{P}_{\text{MP}}(z) = \tfrac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} r_{\mathbf{k}}(z). \tag{22}$$

While this looks intuitively reasonable, proving that Eq. (22) is a valid marginal likelihood estimator: the full proofs are given in [Anonymous, 2023] (their Appendix C.1.3).

The next challenge is to compute the sum in Eq. (22). The sum looks intractable as we have to sum over K^n settings of $\mathbf{k}$. However, it turns out that these sums usually are tractable. The reason is that that if we fix the samples, z, then $r_{\mathbf{k}}(z)$ can be understood as a product of low-rank tensors,

$$r_{\mathbf{k}}(z) = f^x_{\mathbf{k}_{\text{pa}(x)}}(z) \prod_i f^i_{k_i, \mathbf{k}_{\text{pa}(i)}}(z) \tag{23}$$

$$f^x_{\mathbf{k}_{\text{pa}(x)}}(z) = \text{P}\left(x \middle| z_j^{k_j} \text{ for all } j \in \text{pa}\left(x\right)\right), \tag{24}$$

$$f^i_{k_i, \mathbf{k}_{\text{pa}(i)}}(z) = \frac{\text{P}\left(z_i^{k_i} \middle| z_j^{k_j} \text{ for all } j \in \text{pa}\left(i\right)\right)}{\text{Q}_{\text{MP}}\left(z_i^{k_i} \middle| z_j \text{ for all } j \in \text{qa}\left(i\right)\right)}. \tag{25}$$

Here, $f^x_{\mathbf{k}_{\text{pa}(x)}}(z)$ is a rank $|\text{pa}\left(x\right)|$ tensor, and $f^i_{k_i, \mathbf{k}_{\text{pa}(i)}}(z)$ are rank $1 + |\text{pa}\left(i\right)|$ tensors, where $|\text{pa}\left(i\right)|$ denotes the number of parents of the ith latent variable. Thus, the sum in Eq. (22) is really a large tensor product,

$$\mathcal{P}_{\text{MP}}(z) = \tfrac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} f^x_{\mathbf{k}_{\text{pa}(x)}}(z) \prod_i f^i_{k_i, \mathbf{k}_{\text{pa}(i)}}(z) \tag{26}$$

which can be directly and efficiently computed using an opt-einsum implementation. This is discussed in more depth in [Anonymous, 2023].

Of course, the contributions of this paper are not in computing the unbiased marginal likelihood estimator. They are in using the marginal likelihood estimator to compute the other key quantities of interest in Bayesian computations, namely posterior expectations, marginals and samples, and we consider each in turn in the following sections.

4 Methods

Now, we can define an importance sampling scheme that operates on all K^n combinations of samples,

$$m_{\text{MP}}(z) = \tfrac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} \frac{r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)} m(z^{\mathbf{k}}). \tag{27}$$

This looks very similar to the standard global importance sampling scheme in Eq. (5), except that Eq. (5) averages only over K samples, whereas this massively parallel moment estimator averages over all K^n combinations of samples. Proving that this is a valid importance-sampled moment estimator is not trivial (see Appendix A for details).

4.1 Interpreting massively parallel importance weighting as inference in a discrete graphical model

Now, $\tfrac{1}{K^n} r_{\mathbf{k}}(z)/\mathcal{P}_{\text{MP}}(z)$ in (Eq. 27) can be understood as a normalized probability distribution over $\mathbf{k}$. In particular, this quantity is always positive, and we can show that it normalizes to 1 by substituting the definition of $\mathcal{P}_{\text{MP}}(z)$ from Eq. (22),

$$\tfrac{1}{K^n} \sum_{\mathbf{k}} \frac{r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)} = \frac{\tfrac{1}{K^n} \sum_{\mathbf{k}} r_{\mathbf{k}}(z)}{\tfrac{1}{K^n} \sum_{\mathbf{k}'} r_{\mathbf{k}'}(z)} = 1 \tag{28}$$

As such, we can in principle use methods for discrete graphical models, treating $\mathbf{k}$ as a random variable. However, as discussed in the Related work, this treating the problem as inference in a discrete graphical model is still prohibitively difficult.

4.2 Computing expectations by differentiating an estimate of the normalizing constant

Instead, inspired by Sec. 3, we modify our marginal likelihood estimator by introducing a source term,

$$\mathcal{P}^{\text{exp}}_{\text{MP}}(z, J) = \frac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} r_{\mathbf{k}}(z) e^{J m(z^{\mathbf{k}})}. \tag{29}$$

Remember that $r_{\mathbf{k}}(z)$ is a product of low-rank tensors, indexed by subsets of $\mathbf{k}$ (Eq. 23), so the sum can be computed efficiently using opt-einsum. Critically, the source term is just another factor with indices given by a subset of $\mathbf{k}$. For instance, most often m (the function whose expectation we want to compute) will depend on only a single latent variable $m(z^{\mathbf{k}}) = m(z_i^{k_i})$, in which case the source term can be understood as just another tensor in the tensor product, with one index, k_i,

$$f^m_{k_i} = e^{J m(z^{\mathbf{k}})}. \tag{30}$$

Again, we have that $\mathcal{P}^{\text{exp}}_{\text{MP}}(z, J = 0) = \mathcal{P}_{\text{MP}}(z)$. Inspired by Sec. 3, we now differentiate $\log \mathcal{P}^{\text{exp}}_{\text{MP}}(z, J)$ at $J = 0$. Specifically,

$$\frac{\partial}{\partial J}\bigg|_{J=0} \log \mathcal{P}^{\text{exp}}_{\text{MP}}(z, J) = \frac{\frac{\partial}{\partial J}\big|_{J=0} \mathcal{P}^{\text{exp}}_{\text{MP}}(z, J)}{\mathcal{P}^{\text{exp}}_{\text{MP}}(z, 0)} \tag{31}$$

Substituting for $\mathcal{P}^{\text{exp}}_{\text{MP}}(z, J)$ (Eq. 29), and remembering that $\mathcal{P}^{\text{exp}}_{\text{MP}}(z, 0) = \mathcal{P}_{\text{MP}}(z)$,

$$\frac{\partial}{\partial J}\bigg|_{J=0} \log \mathcal{P}^{\text{exp}}_{\text{MP}}(z, J) = \frac{\frac{1}{K^n} \sum_{\mathbf{k}} r_{\mathbf{k}}(z) \, \frac{\partial}{\partial J}\big|_{J=0} e^{J m(z^{\mathbf{k}})}}{\mathcal{P}_{\text{MP}}(z)} \tag{32}$$

Computing the gradient of $e^{J m(z^{\mathbf{k}})}$ at $J = 0$,

$$\begin{aligned} \frac{\partial}{\partial J}\bigg|_{J=0} \log \mathcal{P}^{\text{exp}}_{\text{MP}}(z, J) &= \frac{\frac{1}{K^n} \sum_{\mathbf{k}} r_{\mathbf{k}}(z) m(z^{\mathbf{k}})}{\mathcal{P}_{\text{MP}}(z)} \\ &= m_{\text{MP}}(z) \end{aligned} \tag{33}$$

where the final equality comes from the definition of $m_{\text{MP}}(z)$ in Eq. (27). Note that this derivation is quite different from the standard "source-term trick" from Physics described in Sec. 3, which works with either the true normalizing constant, or with a low-order perturbation to that normalizing constant. In contrast, here we use a quite different massively parallel sample-based estimate of the marginal likelihood. Importantly, the subsequent two derivations are even more different from uses of the "source-term trick" in Physics, as these uses are typically around computing a moment/expectation, while the subsequent two derivations use the same trick to compute quite different quantities (namely, probability distributions over samples).

4.3 Computing marginal importance weights

Computing expectations directly is very powerful and almost certainly necessary for computing complex quantities that depend on multiple latent variables. However, if we are primarily interested in expectations of individual variables, then it is considerably more flexible to compute "marginal" importance weights. Once we have these marginal importance weights, we can easily compute arbitrary expectations for individual variables, and other quantities such as effective sample sizes. To define the marginal weights for the ith latent, note that a moment for the ith latent variable can be written as a sum over k_i,

$$m_{\text{MP}}(z) = \sum_{\mathbf{k}\in\mathcal{K}^n} \frac{r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)} m(z_i^{k_i}) = \sum_{k_i} w^i_{k_i} m(z_i^{k_i}), \tag{34}$$

where $w^i_{k_i}$ are the marginal importance weights for the ith latent variable, which are defined by,

$$w^i_{k_i} = \frac{\frac{1}{K^n}\sum_{\mathbf{k}/k_i\in\mathcal{K}^{n-1}} r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)}, \tag{35}$$

where the sum is over all $\mathbf{k}$ except k_i. Formally,

$$\mathbf{k}/k_i = (k_1, \dots, k_{i-1}, k_{i+1}, \dots, k_n) \in \mathcal{K}^{n-1}. \tag{36}$$

Again we can compute the marginal importance weights using gradients of a slightly different modified marginal likelihood estimator. Specifically, we now use a vector-valued $\mathbf{J} \in \mathbb{R}^K$ in a slightly different modified marginal likelihood estimator,

$$\mathcal{P}^{\text{marg}}_{\text{MP}}(z, \mathbf{J}) = \tfrac{1}{K^n}\sum_{\mathbf{k}} r_{\mathbf{k}}(z) e^{J_{k_i}}. \tag{37}$$

Again, $\mathcal{P}^{\text{marg}}_{\text{MP}}(z, \mathbf{0}) = \mathcal{P}_{\text{MP}}(z)$. As before, we differentiate $\log \mathcal{P}^{\text{marg}}_{\text{MP}}(z, \mathbf{J})$ at $\mathbf{J} = \mathbf{0}$,

$$\frac{\partial}{\partial J_{k_i'}}\bigg|_{\mathbf{J}=\mathbf{0}} \log \mathcal{P}^{\text{marg}}_{\text{MP}}(z, \mathbf{J}) = \frac{\frac{\partial}{\partial J_{k_i'}}\Big|_{\mathbf{J}=\mathbf{0}} \mathcal{P}^{\text{marg}}_{\text{MP}}(z, \mathbf{J})}{\mathcal{P}^{\text{marg}}_{\text{MP}}(z, \mathbf{0})}. \tag{38}$$

Substituting for $\mathcal{P}^{\text{marg}}_{\text{MP}}(z, \mathbf{J})$ in the numerator,

$$\frac{\partial}{\partial J_{k_i'}}\bigg|_{\mathbf{J}=\mathbf{0}} \log \mathcal{P}^{\text{marg}}_{\text{MP}}(z, \mathbf{J}) = \frac{\frac{1}{K^n}\sum_{\mathbf{k}} r_{\mathbf{k}}(z) \; \frac{\partial}{\partial J_{k_i'}}\Big|_{\mathbf{J}=\mathbf{0}} e^{J_{k_i}}}{\mathcal{P}_{\text{MP}}(z)}. \tag{39}$$

The gradient is 1 when $k_i' = k_i$ and zero otherwise which can be represented using a Kronecker delta,

$$\frac{\partial}{\partial J_{k_i'}}\bigg|_{\mathbf{J}=\mathbf{0}} \log \mathcal{P}^{\text{marg}}_{\text{MP}}(z, \mathbf{J}) = \frac{\frac{1}{K^n}\sum_{\mathbf{k}} r_{\mathbf{k}}(z)\delta_{k_i', k_i}}{\mathcal{P}_{\text{MP}}(z)}. \tag{40}$$

We can rewrite this as a sum over all $\mathbf{k}$ except k_i,

$$\begin{aligned}\frac{\partial}{\partial J_{k_i}}\bigg|_{\mathbf{J}=\mathbf{0}} \log \mathcal{P}^{\text{marg}}_{\text{MP}}(z, \mathbf{J}) &= \frac{\frac{1}{K^n}\sum_{\mathbf{k}/k_i\in\mathcal{K}^{n-1}} r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)} \\ &= w^i_{k_i},\end{aligned} \tag{41}$$

which is exactly the definition of the marginal importance weights in Eq. (35).

4.4 Computing conditional distributions for importance sampling

A common alternative to importance weighting is importance sampling. In importance sampling, we rewrite the usual estimates of the expectations in terms of a distribution over indices, $\mathrm{P}(\mathbf{k})$,

$$m_{\text{MP}}(z) = \sum_{\mathbf{k} \in \mathcal{K}^n} \mathrm{P}(\mathbf{k})\, m(z^{\mathbf{k}}) \tag{42}$$

$$\mathrm{P}(\mathbf{k}) = \frac{1}{K^n} \frac{1}{\mathcal{P}_{\text{MP}}(z)} r_{\mathbf{k}}(z) \tag{43}$$

Using $\mathbf{k}$ samples from this distribution, $z^{\mathbf{k}}$ are approximate samples from the true posterior. However, sampling from $\mathrm{P}(\mathbf{k})$ is difficult in our context, as there are K^n possible settings of $\mathbf{k}$, so we cannot explicitly compute the full distribution. Instead, we need to factorise the distribution in some way, and iteratively sample (e.g. we sample k_1 from $\mathrm{P}(k_1)$ then sample k_2 from $\mathrm{P}(k_2|k_1)$ etc.) However, this raises a question: how should we factorise the distribution over $\mathbf{k}$? This is a difficult problem in the probabilistic programming setting, because we need a factorisation that is always valid, and at the same time, has as few indices in each term as possible, to ensure that the computations remain efficient. By substituting Eq. (23) into Eq. (43),

$$\mathrm{P}(\mathbf{k}) = \frac{1}{K^n} \frac{1}{\mathcal{P}_{\text{MP}}(z)} f^x_{\mathbf{k}_{\text{pa}(x)}}(z) \prod_i f^i_{k_i, \mathbf{k}_{\text{pa}(i)}}(z) \tag{44}$$

we can see that one valid factorisation follows the factorisation of the generative model. This is particularly useful, as it is guaranteed to be a valid factorisation, likely to be small (if not minimal) and it is easy for us to extract. Formally, we use,

$$\mathrm{P}(\mathbf{k}) = \prod_i \mathrm{P}\left(k_i \middle| \mathbf{k}_{\text{pa}(i)}\right) \tag{45}$$

where, remember pa(i) is the set of indices of parents of the ith latent variable under the generative model, so,

$$\mathbf{k}_{\text{pa}(i)} = (k_j \text{ for all } j \in \text{pa}(i)) \tag{46}$$

Now, we have the problem of computing the conditionals, $\mathrm{P}\left(k_i \middle| \mathbf{k}_{\text{pa}(i)}\right)$. We can compute the conditionals from the marginals using Bayes theorem,

$$\mathrm{P}\left(k_i \middle| \mathbf{k}_{\text{pa}(i)}\right) = \frac{\mathrm{P}\left(k_i, \mathbf{k}_{\text{pa}(i)}\right)}{\sum_{k_i'} \mathrm{P}\left(k_i', \mathbf{k}_{\text{pa}(i)}\right)} \tag{47}$$

where the marginals are given by,

$$\mathrm{P}\left(k_i, \mathbf{k}_{\text{pa}(i)}\right) = \sum_{\mathbf{k}/(k_i, \mathbf{k}_{\text{pa}(i)})} \mathrm{P}(\mathbf{k}) \tag{48}$$

Again, we can compute these marginals efficiently by differentiating a modified estimate of the marginal likelihood. This time, we take a tensor-valued $\mathbf{J} \in \mathbb{R}^{K^{1+|\text{pa}(i)|}}$, where remember $|\text{pa}(i)|$ is the number of parents of the ith latent variable under the generative model.

$$\mathcal{P}^{\text{samp}}_{\text{MP}}(z, \mathbf{J}) = \frac{1}{K^n} \sum_{\mathbf{k}} \frac{\mathrm{P}\left(x, z^{\mathbf{k}}\right)}{\mathrm{Q}\left(z^{\mathbf{k}}\right)} e^{J_{k_i, \mathbf{k}_{\text{pa}(i)}}} \tag{49}$$

As usual, we differentiate with respect to $\mathbf{J}$ at $\mathbf{J} = \mathbf{0}$,

$$\left.\frac{\partial}{\partial J_{k_i', \mathbf{k}'_{\text{pa}(i)}}}\right|_{\mathbf{J}=\mathbf{0}} \log \mathcal{P}^{\text{samp}}_{\text{MP}}(z, \mathbf{J}) = \frac{\frac{1}{K^n} \sum_{\mathbf{k}} r_{\mathbf{k}}(z) \delta_{(k_i, \mathbf{k}_{\text{pa}(i)}), (k_i', \mathbf{k}'_{\text{pa}(i)})}}{\mathcal{P}_{\text{MP}}(z)} \tag{50}$$

Here, $\delta_{(k_i, \mathbf{k}_{\text{pa}(i)}),(k'_i, \mathbf{k}'_{\text{pa}(i)})}$ is a generalisation of the Kronecker delta. It is 1 when all the indices match (i.e. $k_i = k'_i$, and $\mathbf{k}_{\text{pa}(i)} = \mathbf{k}'_{\text{pa}(i)}$) and zero otherwise.

$$\begin{aligned}
\left.\frac{\partial}{\partial J_{k_i,\mathbf{k}_{\text{pa}(i)}}}\right|_{\mathbf{J}=\mathbf{0}} \log \mathcal{P}_{\text{MP}}^{\text{samp}}(z, \mathbf{J}) &= \frac{\frac{1}{K^n}\sum_{\mathbf{k}/(k_i,\mathbf{k}_{\text{pa}(i)})} r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)} \\
&= \sum_{\mathbf{k}/(k_i,\mathbf{k}_{\text{pa}(i)})} \text{P}\left(\mathbf{k}\right) \\
&= \text{P}\left(k_i, \mathbf{k}_{\text{pa}(i)}\right)
\end{aligned} \tag{51}$$

These are precisely the marginals in Eq. (48).

5 Experiments

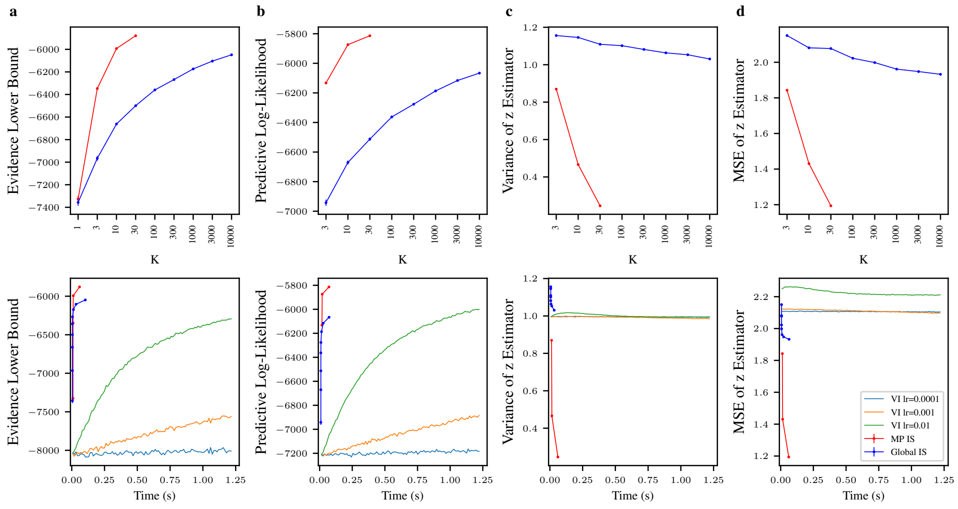

Figure 1: Results obtained in the MovieLens model. Columns **a–c** show the evidence lower bound, predictive log-likelihood and variance in the estimator of $\mathbf{z}_m$ using the true MovieLens100K data. Column **d** shows the mean squared error in the estimator of $\mathbf{z}_m$ when the data is sampled from the model and thus the true value of $\mathbf{z}_m$ is known.

We provide empirical results comparing the global and massively parallel importance weighting/sampling methods. We consider four quantities in Fig. 1 and Fig. 2. First, we compute the ELBO that arises just from the forward pass. While the ELBO can be computed using methods in [Anonymous, 2023], and does not require the contributions in this paper, it is a useful sanity check. Second, we consider the predictive log-likelihood, computed using posterior samples of the latent variables. To obtain these posterior samples, we use the methods in Sec. 4.4. Finally, we consider the quality of the moment estimates (in particular an estimate of the posterior mean of one of the latent variables), where those moments are computed using the methods in Sec. 4.3. In particular, we use two measures of the quality of the estimates, based on real data and data sampled from the generative model. On real data, all we can do is to look at the variance of our estimator of the posterior mean (Fig. 1c, Fig. 2c). In contrast, if we sample from the generative model, we know the true value of the latent variable used to sample the observations. We can therefore compute the MSE between our estimate of the posterior mean and the true value of the latent variable used to generate the data. However, we must be careful when interpreting this quantity, as the true posterior mean is not necessarily equal to the value of the latent variable used to generate the data (they only become equal in the infinite data limit).

In order to compare the two methods in the simplest possible case, we use the prior as the proposal in all models. The values of interest (evidence lower bound, predictive log-likelihood and posterior expectation estimates both with real and generated data) were calculated 1000 times using different importance weights, generated with both the global and the massively parallel scheme, and averaged to obtain the results presented in figures 1 and 2.

5.1 MovieLens Dataset

For our first experiment we use the MovieLens100K Harper and Konstan [2015] dataset, containing 100K ratings of $\mathrm{N} = 1682$ films from among $\mathrm{M} = 943$ users. In our experiments these ratings are binarised from $(0, 1, 2, 3)$ to 0 and from $(4, 5)$ to 1, as in Geffner and Domke [2022]. We use n to index films, and m to index users. Each film has a feature vector $\mathbf{x}_n$ and we use the following hierarchical model, similar to that in [Anonymous, 2023]:

$$\begin{aligned}
\boldsymbol{\mu} &\sim \mathcal{N}(\mathbf{0}_{18}, 0.25\mathbf{I}) \\
\boldsymbol{\psi} &\sim \mathcal{N}(\mathbf{0}_{18}, 0.25\mathbf{I}) \\
\mathbf{z}_m &\sim \mathcal{N}(\boldsymbol{\mu}, \exp(\boldsymbol{\psi})\mathbf{I}),\ m = 1, \dots, \mathrm{M} \\
\mathrm{Rating}_{mn} &\sim \mathrm{Bernoulli}(\sigma(\mathbf{z}_m^{\intercal}\mathbf{x}_n)),\ n = 1, \dots, \mathrm{N}
\end{aligned} \tag{52}$$

After first sampling a global mean, $\boldsymbol{\mu}$, and variances, $\boldsymbol{\psi}$, the model samples a latent vector, $\mathbf{z}_m$, for each user, corresponding to the features of films $\mathbf{x}_n$ they are likely to rate highly. We take the dot-product of the latent user-vector, $\mathbf{z}_m$, and the film's feature vector, $\mathbf{x}_n$, to obtain the probability of a positive rating for film n from user m. A corresponding graphical model is given in Appendix B.1.

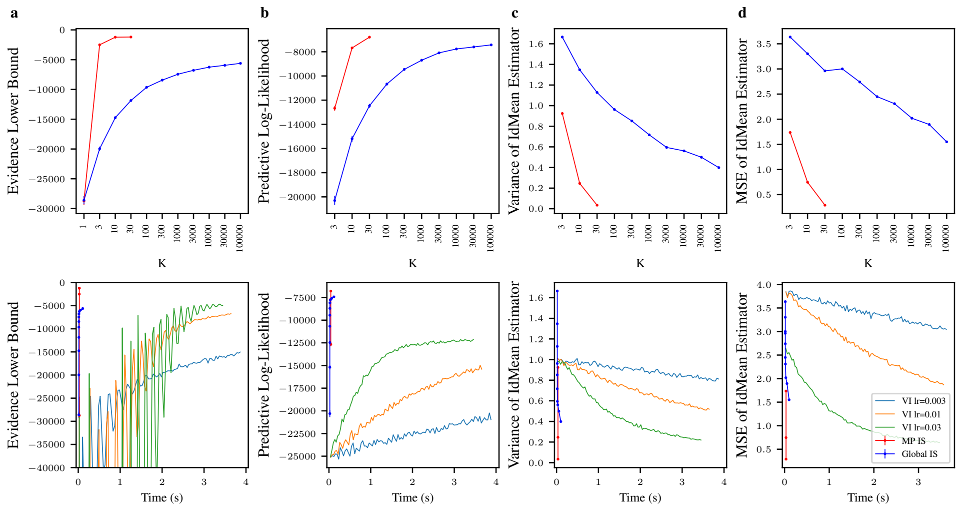

Figure 2: Results obtained in the NYC Bus Breakdown model. Columns **a**–**c** show the evidence lower bound, predictive log-likelihood and variance in the estimator of IdMean_{mj} using the true data. Column **d** shows the mean squared error in the estimator of IdMean_{mj} when the data is sampled from the model.

We use a random subset of $\mathrm{N} = 20$ films and $\mathrm{M} = 450$ users for our experiment, with an equally sized but disjoint subset held aside for calculation of the predictive log-likelihood. We ensure high levels of uncertainty in the per user mean $\mathbf{z}_m$ by using far fewer films than users, and assume ratings of 0 for films which users have not previously rated. In order to obtain results with comparable computation times we test the massively parallel importance samples for $K \in \{3, 10, 30\}$ and the global importance samples for $K \in \{3, 10, 30, 100, 300, 1000, 3000, 10000\}$.

Our results (Fig. 1) show tighter evidence lower bounds and higher predictive log-likelihoods using massively parallel importance samples than using global importance samples for a given $K > 1$, with superior results obtained in less time once K is large enough in the massively parallel approach (in this case, for $K \geq 10$). We also see much lower variance in our importance weighted estimates for the expectation of per user means, $\mathbf{z}_m$, for every $K > 1$ and again see this reduced variance achieved in less time than the global method for large enough K (here, $K \geq 3$). Using data sampled from the model in order to obtain ground truth values of $\mathbf{z}_m$, we found the mean squared error of the

$\mathbf{z}_m$ expectation estimates was far lower for all $K > 1$ when those estimates were calculated using massively parallel importance weights rather than global importance weights, with similar behaviour in the time required to reduce the MSE as with reducing the variance.

5.2 NYC Bus Breakdown Dataset

In our second experiment, we model the length of delay time of New York school bus journeys, working with a dataset supplied by the City of New York DOE [2023], based on the school year, borough, and the ID of the bus in question. These three levels motivate a hierarchical model similar to that in Anonymous [2023] — a mean and variance is sampled for each year, which are used to sample a borough mean which, along with a sampled borough variance, is used to sample an ID mean for each year and borough. It is these ID means, denoted by IdMean_{mj} for year m and borough j, over IDs $i = 1, ..., \text{I}$, for which we estimate the posterior expectation. After this, the model samples a final variance that is used to sample two weight vectors which are multiplied with two vectors of covariates indicating the bus company and journey type. Finally, the predicted delay is sampled from a negative binomial distribution with logits given by the sum of these two multiplied terms and the ID mean. The full specification of the model can be found in Appendix B.2, with a corresponding graphical model provided in Appendix B.3.

The experiments were run using a subset of the data with $\text{I} = 30$ IDs from $\text{J} = 3$ boroughs in $\text{M} = 3$ years. As with the previous experiment, an equally sized but disjoint subset of the data is used to calculate predictive log-likelihoods. We test the massively parallel importance samples for $K \in \{3, 10, 30\}$ and the global importance samples for $K \in \{3, 10, 30, 100, 300, 1000, 3000, 10000, 30000, 100000\}$.

We run the same experiments as on the MovieLens model, but using posterior expectation estimates for IdMean_{mj}, as displayed in Fig. 2. We again observe improved performance from massively parallel importance sampling and weighting compared to global importance sampling and weighting.

6 Conclusion

We have shown how posterior moments, marginals and samples may be computed using massively parallel importance sampling/weighting methods by drawing K samples for n latent variables, and individually reasoning about all K^n combinations. We gave a new and far simpler method for computing these quantities based on differentiating a slightly modified marginal likelihood estimator. These approaches led to better importance samples (as measured by predictive log-likelihoods) and better importance weighted moment estimates.

References

Laurence Aitchison. Tensor Monte Carlo: particle methods for the GPU era. *Advances in Neural Information Processing Systems*, 32, 2019.

Aws Albarghouthi, Loris D'Antoni, Samuel Drews, and Aditya V Nori. Fairsquare: probabilistic verification of program fairness. *Proceedings of the ACM on Programming Languages*, 1 (OOPSLA):1–30, 2017.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3): 269–342, 2010.

Anonymous. Massively parallel reweighted wake-sleep. *Submitted to UAI 2023, and included in the Appendix*, 2023.

Bozhena Bidyuk and Rina Dechter. Cutset sampling for bayesian networks. *Journal of Artificial Intelligence Research*, 28:1–48, 2007.

Jörg Bornschein and Yoshua Bengio. Reweighted wake-sleep. *arXiv preprint arXiv:1406.2751*, 2014.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Sourav Chatterjee and Persi Diaconis. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135, 2018.

Guillaume Claret, Sriram K Rajamani, Aditya V Nori, Andrew D Gordon, and Johannes Borgström. Bayesian inference using data flow analysis. In *Proceedings of the 2013 9th joint meeting on foundations of software engineering*, pages 92–102, 2013.

Adrien Corenflos, Nicolas Chopin, and Simo Särkkä. De-sequentialized monte carlo: a parallel-in-time particle smoother. *arXiv preprint arXiv:2202.02264*, 2022.

A Philip Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and computing*, 2(1):25–36, 1992.

DOE. Bus breakdown and delays, 2023. url https://data.cityofnewyork.us/Transportation/Bus-Breakdown-and-Delays/ez4e-fazm Accessed on: 05.05.2023.

Arnaud Doucet, Adam M Johansen, et al. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

Tomas Geffner and Justin Domke. Variational inference with locally enhanced bounds for hierarchical models. *arXiv preprint arXiv:2203.04432*, 2022.

Timon Gehr, Sasa Misailovic, and Martin Vechev. Psi: Exact symbolic inference for probabilistic programs. In *International Conference on Computer Aided Verification*, pages 62–83. Springer, 2016.

Jaco Geldenhuys, Matthew B Dwyer, and Willem Visser. Probabilistic symbolic execution. In *Proceedings of the 2012 International Symposium on Software Testing and Analysis*, pages 166–176, 2012.

Noah D Goodman and Andreas Stuhlmüller. The design and implementation of probabilistic programming languages, 2014.

Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET, 1993.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Steven Holtzen, Guy Van den Broeck, and Todd Millstein. Scaling exact inference for discrete probabilistic programs. *Proceedings of the ACM on Programming Languages*, 4(OOPSLA):1–31, 2020.

Jinlin Lai, Justin Domke, and Daniel Sheldon. Variational marginal particle filters. In *International Conference on Artificial Intelligence and Statistics*, pages 875–895. PMLR, 2022.

Tuan Anh Le, Maximilian Igl, Tom Rainforth, Tom Jin, and Frank Wood. Auto-encoding sequential monte carlo. *arXiv preprint arXiv:1705.10306*, 2017.

Tuan Anh Le, Adam R Kosiorek, N Siddharth, Yee Whye Teh, and Frank Wood. Revisiting reweighted wake-sleep for models with stochastic control flow. In *Uncertainty in Artificial Intelligence*, pages 1039–1049. PMLR, 2020.

Fredrik Lindsten, Adam M Johansen, Christian A Naesseth, Bonnie Kirkpatrick, Thomas B Schön, JAD Aston, and Alexandre Bouchard-Côté. Divide-and-conquer with sequential monte carlo. *Journal of Computational and Graphical Statistics*, 26(2):445–458, 2017.

Chris J Maddison, John Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Teh. Filtering variational objectives. *Advances in Neural Information Processing Systems*, 30, 2017.

Christian Naesseth, Scott Linderman, Rajesh Ranganath, and David Blei. Variational sequential monte carlo. In *International conference on artificial intelligence and statistics*, pages 968–977. PMLR, 2018.

Praveen Narayanan, Jacques Carette, Wren Romano, Chung-chieh Shan, and Robert Zinkov. Probabilistic inference by program transformation in hakaru (system description). In *International Symposium on Functional and Logic Programming*, pages 62–79. Springer, 2016.

Fritz Obermeyer, Eli Bingham, Martin Jankowiak, Neeraj Pradhan, Justin Chiu, Alexander Rush, and Noah Goodman. Tensor variable elimination for plated factor graphs. In *International Conference on Machine Learning*, pages 4871–4880. PMLR, 2019.

Avi Pfeffer. The design and implementation of ibal: A generalpurpose probabilistic programming language. In *Harvard Univesity*. Citeseer, 2005.

Sriram Sankaranarayanan, Aleksandar Chakarov, and Sumit Gulwani. Static analysis for probabilistic programs: inferring whole program properties from finitely many paths. In *Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation*, pages 447–458, 2013.

Di Wang, Jan Hoffmann, and Thomas Reps. Pmaf: an algebraic framework for static analysis of probabilistic programs. *ACM SIGPLAN Notices*, 53(4):513–528, 2018.

Steven Weinberg. *The quantum theory of fields*, volume 2. Cambridge university press, 1995.

Jacob Zavatone-Veth, Abdulkadir Canatar, Ben Ruben, and Cengiz Pehlevan. Asymptotics of representation learning in finite bayesian neural networks. *Advances in neural information processing systems*, 34:24765–24777, 2021.

A Derivations

A.1 Global Importance Sampling

Here, we give the derivation for standard global importance sampling. Ideally we would compute moments using the true posterior, $\mathrm{P}\left(z|x\right)$,

$$m_{\text{post}} = \mathrm{E}_{\mathrm{P}\left(z^k|x\right)}\left[m(z^k)\right]. \tag{53}$$

However, the true posterior is not known. Instead, we write down the moment under the true posterior as an integral,

$$m_{\text{post}} = \int dz^k\, \mathrm{P}\left(z^k|x\right) m(z^k). \tag{54}$$

Next, we multiply the integrand by $1 = \mathrm{Q}\left(z^k\right)/\mathrm{Q}\left(z^k\right)$,

$$m_{\text{post}} = \int dz^k \mathrm{Q}\left(z^k\right) \frac{\mathrm{P}\left(z^k|x\right)}{\mathrm{Q}\left(z^k\right)} m(z^k). \tag{55}$$

Next, the integral can be written as an expectation,

$$m_{\text{post}} = \mathrm{E}_{\mathrm{Q}\left(z^k\right)}\left[\frac{\mathrm{P}\left(z^k|x\right)}{\mathrm{Q}\left(z^k\right)} m(z^k)\right]. \tag{56}$$

It looks like we should be able to estimate m_{post} by sampling from our approximate posterior, $\mathrm{Q}\left(z^k\right)$. However, this is not yet possible, as we are not able to compute the true posterior, $\mathrm{P}\left(z^k|x\right)$. We might consider using Bayes theorem,

$$\mathrm{P}\left(z^k|x\right) = \frac{\mathrm{P}\left(z^k, x\right)}{\mathrm{P}\left(x\right)} \tag{57}$$

But this requires computing an intractable normalizing constant,

$$\mathrm{P}\left(x\right) = \int dz^k\, \mathrm{P}\left(z^k, x\right). \tag{58}$$

Instead, we use an unbiased, importance-sampled estimate of the normalizing constant, $\mathcal{P}_{\text{global}}(z)$ (Eq. 7). Additionally, Burda et al. [2015] showed that in the limit as $K \to \infty$, $\mathcal{P}_{\text{global}}(z)$ approaches $\mathrm{P}\left(x\right)$. Using this estimate of the marginal likelihood our moment estimate becomes,

$$m_{\text{global}} = \mathrm{E}_{\mathrm{Q}\left(z^k\right)}\left[\frac{\frac{\mathrm{P}\left(z^k,x\right)}{\mathrm{Q}\left(z^k\right)}}{\mathcal{P}_{\text{global}}(z)} m(z^k)\right]. \tag{59}$$

Using $r_k(z)$ (Eq. 6), we can write this expression as,

$$m_{\text{global}} = \mathrm{E}_{\mathrm{Q}_\phi(z)}\left[\frac{r_k(z)}{\mathcal{P}_{\text{global}}(z)} m(z^k)\right]. \tag{60}$$

The approximate posterior and generative probabilities are the same for different values of k, so we can average over k, which gives Eq. (5) in the main text.

A.2 Massively Parallel Importance Sampling

Inspired by the global importance sampling derivation, we consider massively parallel importance sampling. In the global importance sampling derivation, the key idea was to show that the estimator was unbiased for each of the K samples, z^k, in which case the average over all K samples is also unbiased. In massively parallel importance sampling, we use the same idea, except that we now have K^n samples, denoted $z^{\mathbf{k}}$. As before,

$$m_{\text{post}} = \mathrm{E}_{\mathrm{P}\left(z^{\mathbf{k}}|x\right)}\left[m(z^{\mathbf{k}})\right] = \int dz^{\mathbf{k}}\, \mathrm{P}\left(z^{\mathbf{k}}|x\right) m(z^{\mathbf{k}}). \tag{61}$$

Again, we multiply by $1 = \prod_i \mathrm{Q}\left(z_i^{k_i} \middle| z_{\mathrm{qa}(i)}\right) / \prod_i \mathrm{Q}\left(z_i^{k_i} \middle| z_{\mathrm{qa}(i)}\right)$,

$$m_{\text{post}} = \int dz^{\mathbf{k}} \left(\prod_i \mathrm{Q}\left(z_i^{k_i} \middle| z_{\mathrm{qa}(i)}\right)\right) \frac{\mathrm{P}\left(z^{\mathbf{k}}|x\right)}{\prod_i \mathrm{Q}\left(z_i^{k_i} \middle| z_{\mathrm{qa}(i)}\right)} m(z^{\mathbf{k}}). \tag{62}$$

Overall, our goal is to convert the integral over the indexed latent variables in Eq. (62) into an integral over the full latent space, z, so that it can be written as an expectation over the proposal, $\mathrm{Q}(z)$. To do that, we need to introduce the concept of non-indexed latent variables. These are all samples of the latent variables, except for the "indexed", or kth sample. For the ith latent variable, the non-indexed samples are,

$$z_i^{/k_i} = \left(z_i^1, \dots, z_i^{k_i-1}, z_i^{k_i+1}, \dots, z_i^K\right) \in \mathcal{Z}_i^{K-1}. \tag{63}$$

We can also succinctly write the non-indexed samples of all latent variables as,

$$z^{/\mathbf{k}} = \left(z_1^{/k_1}, z_2^{/k_2}, \dots, z_n^{/k_n}\right) \in \mathcal{Z}^{K-1}. \tag{64}$$

The joint distribution over the non-indexed latent variables, conditioned on the indexed latent variables integrates to 1,

$$1 = \int dz^{/\mathbf{k}} \mathrm{Q}\left(z^{/\mathbf{k}} \middle| z^{\mathbf{k}}\right) = \int dz^{/\mathbf{k}} \prod_i \mathrm{Q}\left(z_i^{/k_i} \middle| z_i^{k_i}, z_{\mathrm{qa}(i)}\right), \tag{65}$$

We use this to multiply the integrand in Eq. (62),

$$m_{\text{post}} = \int dz^{\mathbf{k}} \left(\prod_i \mathrm{Q}\left(z_i^{k_i} \middle| z_{\mathrm{qa}(i)}\right)\right) \frac{\mathrm{P}\left(z^{\mathbf{k}}|x\right)}{\prod_i \mathrm{Q}\left(z_i^{k_i} \middle| z_{\mathrm{qa}(i)}\right)} m(z^{\mathbf{k}}) \int dz^{/\mathbf{k}} \prod_i \mathrm{Q}\left(z_i^{/k_i} \middle| z_i^{k_i}, z_{\mathrm{qa}(i)}\right). \tag{66}$$

Next, we merge the integrals over over $z^{\mathbf{k}}$ and $z^{/\mathbf{k}}$ to form one integral over z,

$$m_{\text{post}} = \int dz\, \mathrm{Q}(z) \frac{\mathrm{P}\left(z^{\mathbf{k}}|x\right)}{\prod_i \mathrm{Q}\left(z_i^{k_i} \middle| z_{\mathrm{qa}(i)}\right)} m(z^{\mathbf{k}}). \tag{67}$$

This integral can be written as an expectation,

$$m_{\text{post}} = \mathrm{E}_{\mathrm{Q}(z)}\left[\frac{\mathrm{P}\left(z^{\mathbf{k}}|x\right)}{\prod_i \mathrm{Q}\left(z_i^{k_i}|z_{\mathrm{pa}(i)}\right)} m(z^{\mathbf{k}})\right]. \tag{68}$$

As in the derivation for global importance sampling, it looks like we might be able to estimate this by sampling from $\mathrm{Q}(z|x)$, but this does not yet work as we do not yet have a form for the posterior. Again, we could compute the posterior using Bayes theorem,

$$\mathrm{P}\left(z^{\mathbf{k}}|x\right) = \tfrac{\mathrm{P}\left(z^{\mathbf{k}},x\right)}{\mathrm{P}(x)}, \tag{69}$$

but we cannot compute the model evidence,

$$\mathrm{P}_\theta(x) = \int dz^{\mathbf{k}}\, \mathrm{P}_\theta\left(z^{\mathbf{k}}, x\right). \tag{70}$$

As in the global importance sampling section, we instead use an estimate of the marginal likelihood. Here, we use a massively parallel estimate, $\mathcal{P}_{\text{MP}}(z)$,

$$m_{\text{MP}} = \mathrm{E}_{\mathrm{Q}(z|x)}\left[\frac{\frac{\mathrm{P}\left(z^{\mathbf{k}},x\right)}{\prod_i \mathrm{Q}\left(z_i^{k_i}|z_{\mathrm{pa}(i)}\right)}}{\mathcal{P}_{\text{MP}}(z)} m(z^{\mathbf{k}})\right]. \tag{71}$$

Again, we use $r_{\mathbf{k}}(z)$ (Eq. 21),

$$m_{\text{MP}} = \mathrm{E}_{\mathrm{Q}(z)}\left[\frac{r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)} m(z^{\mathbf{k}})\right]. \tag{72}$$

So the value for a single set of latent variables, $z^{\mathbf{k}}$, has the right expectation. Thus, averaging over all K^n settings of $\mathbf{k}$, we get the unbiased estimator in the main text, (Eq. 27).

B Experiment Models

B.1 MovieLens Graphical Model

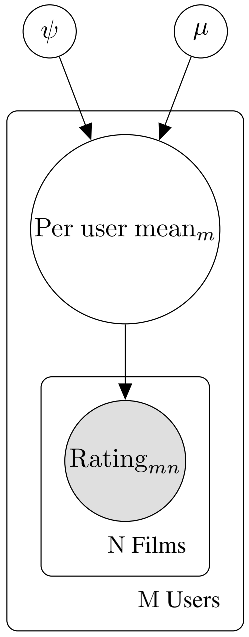

Figure 3: Graphical model for the MovieLens dataset

B.2 Bus Delay Model Specification

$$\begin{aligned}
\text{YearVariance} &\sim \mathcal{N}(0, 10^{-4}) \\
\text{YearMean} &\sim \mathcal{N}(0, 10^{-4}) \\
\text{BoroughMean}_m &\sim \mathcal{N}(\text{YearMean}, \exp(\text{YearVariance})),\ m = 1, ..., \text{M} \\
\text{BoroughVariance}_j &\sim \mathcal{N}(0, 0.25),\ j = 1, ..., \text{J} \\
\text{IdMean}_{mj} &\sim \mathcal{N}(\text{BoroughMean}_m, \text{BoroughVariance}_j),\ \text{j} = 1, ..., \text{J} \\
\text{WeightVariance}_i &\sim \mathcal{N}(0, 10^{-4}),\ i = 1, ..., \text{I} \\
\mathbf{C}_i &\sim \mathcal{N}(\mathbf{0}_{\#\text{BusCo.s}}, \exp(\text{WeightVariance}_i)),\ i = 1, ..., \text{I} \\
\mathbf{J}_i &\sim \mathcal{N}(\mathbf{0}_{\#\text{JourneyTypes}}, \exp(\text{WeightVariance}_i)),\ i = 1, ..., \text{I} \\
\text{logits}_{mji} &= \text{IdMean}_{mj} + \mathbf{C}_i * \text{Bus company name}_{mji} + \mathbf{J}_i * \text{Journey type}_{mji} \\
\text{Delay}_{mji} &\sim \text{NegativeBinomial}(\text{total count} = 130, \text{logits}_{mji}),\ i = 1, ..., \text{I}
\end{aligned} \tag{73}$$

Here $\text{Bus company name}_{mji}$ and $\text{Journey type}_{mji}$ are one-hot encoded indicator variables indicating the bus company and the type of bus journey respectively. The dataset's largest recorded delay is 130, hence we use $\text{total count} = 130$.

B.3 NYC Bus Breakdown Graphical Model

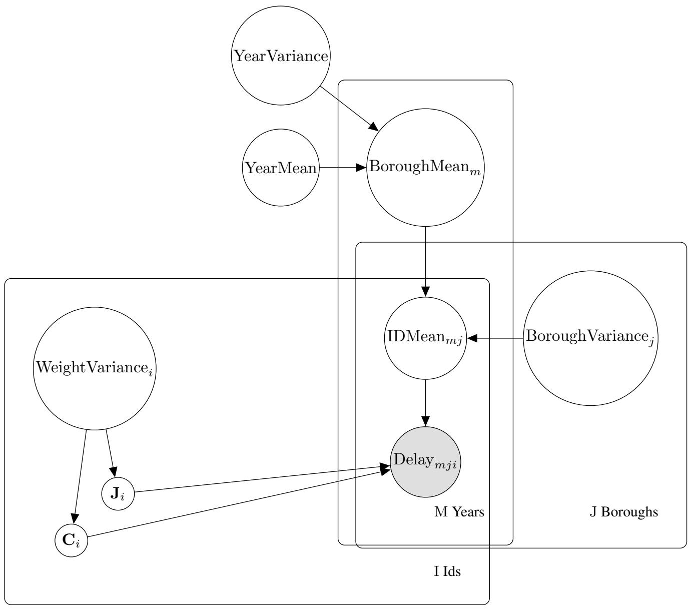

Figure 4: Graphical model for the NYC Bus Breakdown dataset