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 $z^k = (z_1^k, z_2^k, ..., z_n^k) \in \mathcal{Z}$ (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_i^k, the collection of all K samples of the ith latent variable can be written as

$$z_i = (z_i^1, ..., z_i^k). \tag{7}$$

[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 $J_{k_i'}$,

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

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

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

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

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

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

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

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

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

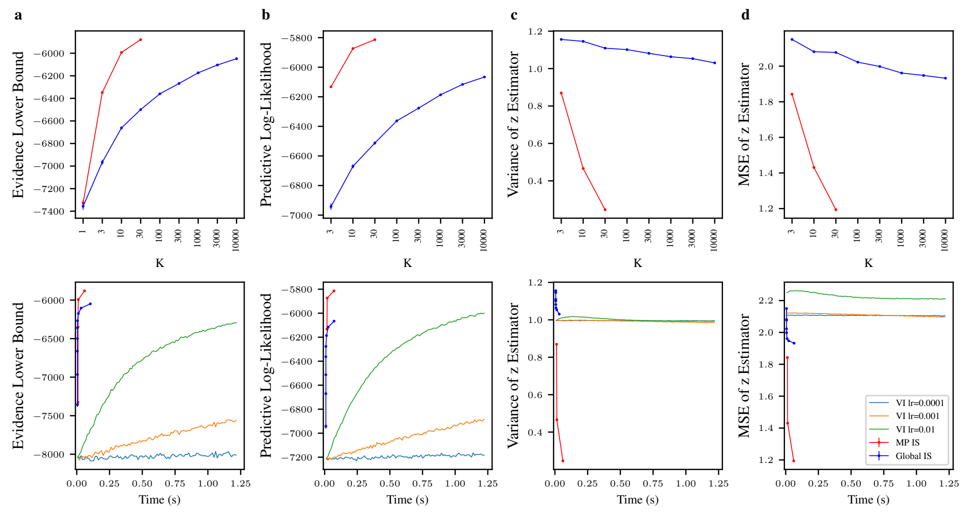

Figure 1: Results obtained in the MovieLens model. Columns **a–c** show the evidence lower bound, predictive log-likelihood and variance in the estimator of the variable $\mathbf{z}_m$ using the true MovieLens100K data. Column **d** shows the mean squared error in the estimator of $\mathbf{z}_m$ when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

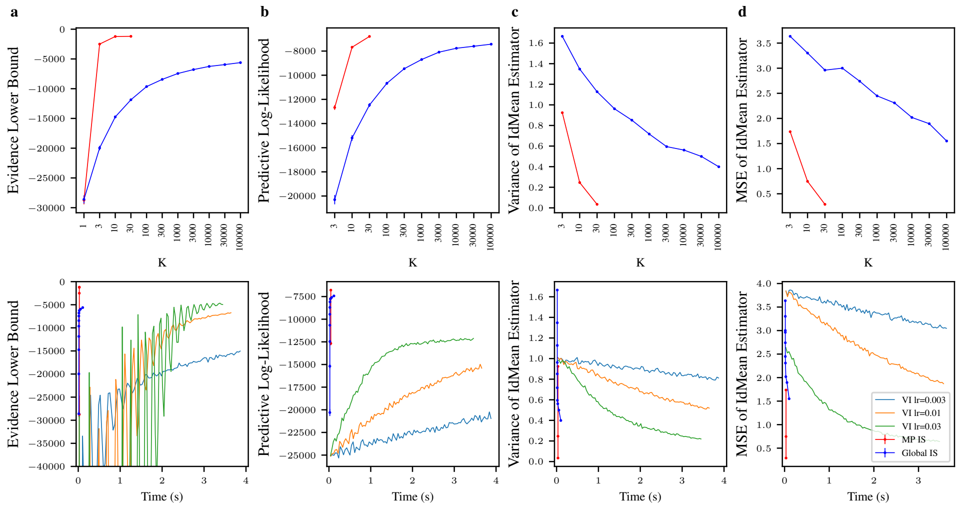

Figure 2: Results obtained in the NYC Bus Breakdown model. Columns **a**–**c** show the evidence lower bound, predictive log-likelihood and variance in the estimator of the variable IdMean_{mj} using the true data. Column **d** shows the mean squared error in the estimator of IdMean_{mj} when the data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

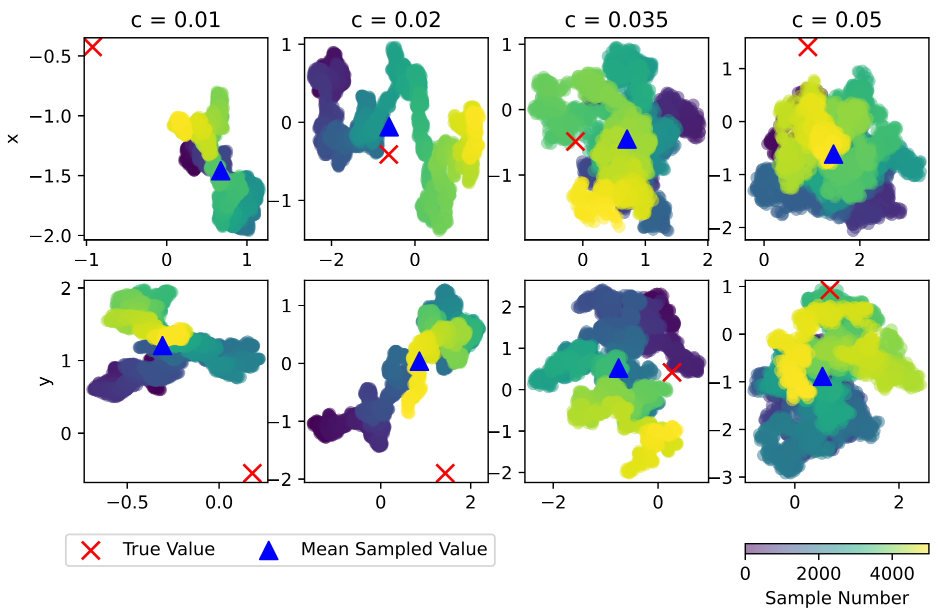

Figure 3: Trace plots of the indexed samples $z^{\mathbf{k}}$ for both x (top row) and y (bottom row) over 5000 iterations with iteration number shown via colour and different values of c used in each of the columns.

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:

$$\mathbf{x}, \mathbf{y} \sim \mathcal{N}(\mathbf{0}_2, \mathbf{I}_2) \tag{52}$$
$$\mathbf{d} \sim \mathcal{N}(\mathbf{x}, \exp(\mathbf{y})\mathbf{I}_2). \tag{53}$$

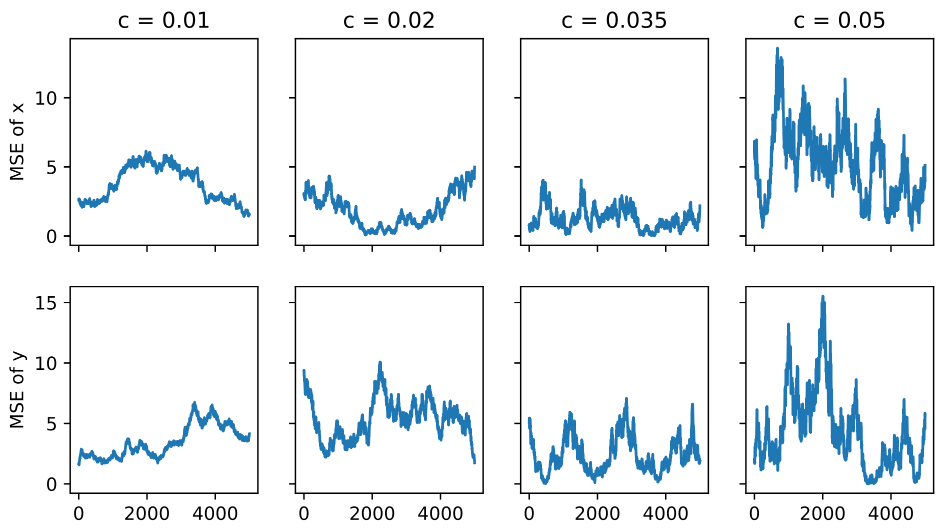

Figure 4: Mean squared error of the indexed samples $z^{\mathbf{k}}$ for both x (top row) and y (bottom row) over 5000 iterations compared to the true underlying values, with different values of c used in each of the columns.

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 $c = 0.035$

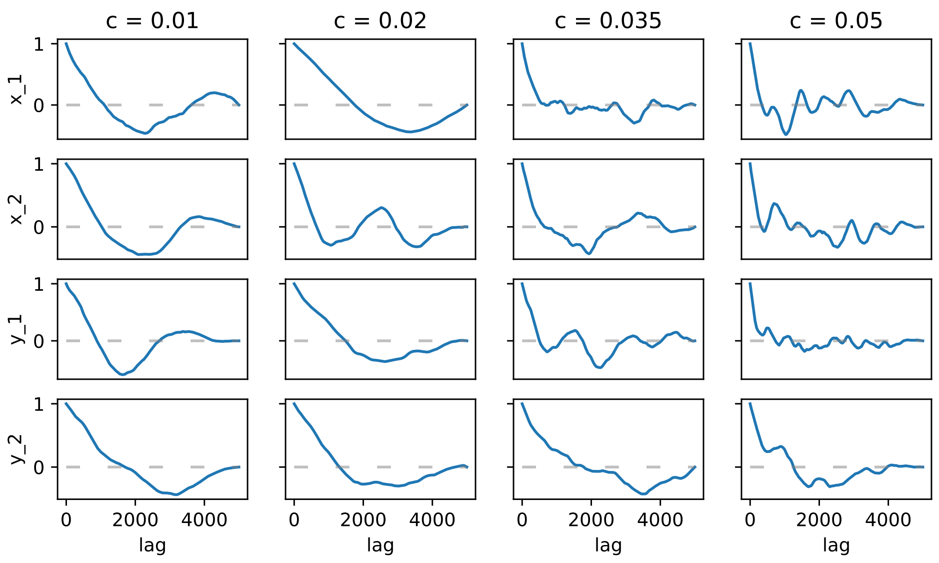

Figure 5: Autocorrelation of the indexed samples $z^{\mathbf{k}}$ for the two dimensions of x (top two rows) and the two dimensions of y (bottom two rows) over 5000 iterations with different values of c used in each of the columns.

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.