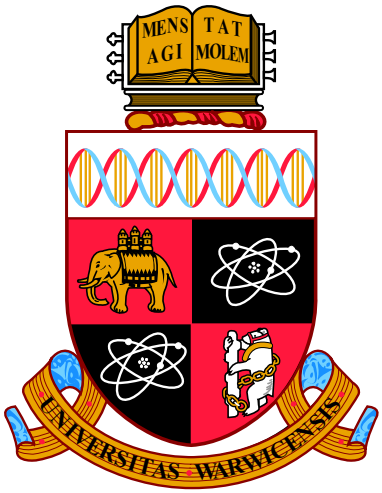

University of Warwick

Department of Statistics

Global consensus Monte Carlo

Lewis Rendell

Supervisor: Adam Johansen

24-month report

September 2017

Global consensus Monte Carlo

September 2017

Abstract

For problems involving large data sets, it is often convenient or necessary to distribute the data across multiple processors. Consider a target probability density function expressed as a product of terms, each being the contribution of one such subset of the data, with an additional 'prior' factor. We introduce an instrumental hierarchical model, associating an instrumental variable with each subset; these variables are conditionally independent when conditioned on a top-level parameter. This permits the construction of an MCMC algorithm on an extended state space, yielding an approximation of the target density as a marginal of its invariant distribution. Specifying the conditional distribution of the artificial variables allows the trade-off between computational tractability and fidelity to the original model to be controlled. In contrast to similar such algorithms, this approach requires few distributional assumptions; we illustrate its performance on a number of simulated examples, and suggest a number of directions for future investigation.

1 Introduction

The large data sets that are commonplace in modern statistical inference make many traditional Markov chain Monte Carlo (MCMC) methods computationally infeasible. In Bayesian inference for example, one typically wishes to produce samples distributed according to some posterior target distribution; such methods generally require a number of operations that is linear in the number of data. Several existing approaches to this problem aim to reduce computational complexity by using only a subsample or batch of the data in each iteration. These include use of the formation of approximate acceptance ratios in standard MCMC algorithms (Korattikara et al., 2014; Bardenet et al., 2014), the use of bounds on the likelihood contribution of each datum (Maclaurin and Adams, 2014), and the application of various stochastic optimisation procedures (Welling and Teh, 2011; Hoffman et al., 2013).

Such procedures, however, are serial in nature and cannot easily be adapted to exploit distributed architectures. When dealing with massive data sets, it is often convenient or necessary to partition the data across multiple cores or machines. We consider the problem of sampling according to a target probability density function on a space $\mathsf{E} \subseteq \mathbb{R}^d$, given by

$$\pi(z) \propto \mu(z) \prod_{j=1}^{b} f_j(z). \tag{1}$$

We assume that f_j is computable on computing node j and involves consideration of $\mathbf{y}_j$, the jth subset or 'block' of the full data set. The function μ corresponds to the prior density in the Bayesian setting, and is not dependent on the data.

Owing to the cost associated with communication between computing nodes, much recent work has focused on 'embarrassingly parallel' approaches to this problem, which run a separate MCMC chain on each node followed by some final processing step. Such algorithms require communication between the nodes only at the very beginning and end of the procedure, and therefore fall into the MapReduce framework (Dean and Ghemawat, 2008). Consensus Monte Carlo (Scott et al., 2016) proceeds in this manner, running an MCMC chain on each node that is invariant with respect to an appropriately-defined 'subposterior' distribution, dependent only on the corresponding block of data. In order to form a chain of values that resemble draws from the true target distribution (with density π), the individual chains are combined elementwise by weighted averaging.

For many examples, however, this approach performs poorly, particularly in cases where at least some of the f_j are highly non-Gaussian (as in examples of Wang et al., 2015; Srivastava et al., 2015). Noting such cases, various authors have proposed other techniques for utilising the values from each of these chains in order to form estimates of π. Scott (2017) suggests a strategy based on finite mixture models, following an earlier proposal by Neiswanger et al. (2014) employing kernel density estimates, though notes that both methods may be infeasible or impractical in high-dimensional settings. The same author proposes a model-agnostic method employing sequential importance sampling, which is also observed to perform poorly in cases of high dimension.

Various other techniques have been proposed for combining the outputs of the individual chains in embarrassingly parallel approaches. Rabinovich et al. (2015) suggest that instead of combining the chains' values by averaging (as in consensus Monte Carlo), one could use variational optimisation to choose a function with which to aggregate the chains. Wang and Dunson (2013) build on the idea of using kernel density estimates by using approximations based on Weierstrass transforms; Wang et al. (2015) suggest a method employing random partition trees. Another alternative direction is taken by Minsker et al. (2014) and Srivastava et al. (2015), who consider forming estimates of each subposterior distribution and taking a suitably-defined mean or median.

In general however, embarrassingly parallel procedures can exhibit serious shortfalls when there are significant differences between the blocks of data. One such case is that in which the data are partitioned into blocks in a non-random manner; there may then be significant differences between the functions f_j, and so the final combination procedure may result in highly inaccurate representations of the true target density. While permuting and re-partitioning the data may result in greater homogeneity across the data blocks, this may not always be feasible, and we later describe cases in which this may not necessarily resolve the problem.

Rather than limiting inter-node communication to a minimum, Xu et al. (2014) propose the use of expectation propagation to approximate each f_j by a surrogate density from an exponential family. Separate MCMC chains are run in parallel on each node, targeting densities in which those f_j that cannot be computed on that node are replaced by their surrogates. Regular moment-

sharing between the nodes allows these surrogates to be iteratively updated, in order that each chain's target density forms a close approximation of the true target density π. Again, however, the effectiveness of this method relies on approximations that may not always be appropriate.

In order to circumvent many of these issues, we propose a new method, motivated by concepts in distributed optimisation. Instead of aiming to avoid entirely communication between nodes, our proposed algorithm is intended to be robust to significant differences between the density contributions f_j, and therefore between the blocks of data.

We proceed by introducing our proposed framework and the resulting algorithmic structure in Section 2, including some discussion of its connections with the consensus Monte Carlo algorithm of Scott et al. (2016). A case study for Gaussian densities is provided in Section 3, giving some insight into the asymptotic behaviour of the algorithm. Various simulation examples are presented in Section 4, before concluding remarks and various directions for further investigation are given in Section 5.

2 The instrumental model

For simplicity, we shall occasionally abuse notation by using the same symbol for a probability measure on E, and for its density with respect to some dominating measure.

In order to facilitate the specification of the algorithm, we introduce a collection of instrumental variables on E denoted by $x_{1:b}$. On the extended state space $\mathsf{E} \times \mathsf{E}^b$, we define the probability density function π_λ by

$$\pi_\lambda(z, x_{1:b}) \propto \mu(z) \prod_{j=1}^{b} K_\lambda(z, x_j) f_j(x_j), \tag{2}$$

where $\{K_\lambda : \lambda \in \mathbb{R}_+\}$ is a family of Markov transition densities. If we define

$$f_j^\lambda(z) \coloneqq \int_{\mathsf{E}} K_\lambda(z, x) f_j(x) \mathrm{d}x, \tag{3}$$

then the z-marginal of π_λ is

$$\pi_\lambda(z) \propto \mu(z) \prod_{j=1}^{b} f_j^\lambda(z).$$

We assume that this family satisfies $f_j^\lambda \to f_j$ pointwise as $\lambda \to 0$, and that f_j is bounded, for all $j \in \{1, \ldots, b\}$. With regard to this z-marginal, this implies convergence in total variation of π_λ to the target π due to Scheffé's lemma (Williams, 1991, p. 55), and hence convergence of expectations of bounded functions.

Considering the family of probability measures induced by $\{K_\lambda : \lambda \in \mathbb{R}_+\}$ with respect to the Lebesgue measure, a sufficient condition for this to hold is that $K_\lambda(z, \cdot)$ converges weakly to the Dirac measure at z as $\lambda \to 0$. To provide an example, considering the state space $\mathsf{E} = \mathbb{R}$, a typical choice for these transition densities might be $K_\lambda(z, x) = \mathcal{N}(x; z, \lambda)$.

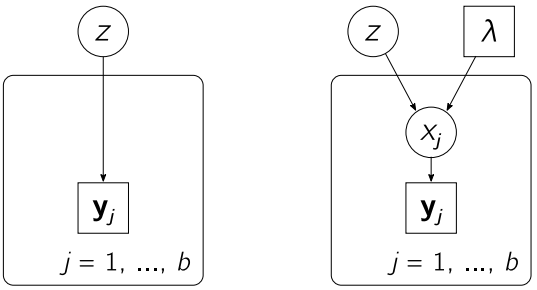

Figure 1: Directed acyclic graphs, representing the true data generating model (left) and the instrumental model we construct (right).

This approach may be considered in terms of the instrumental hierarchical model that it induces, presented diagrammatically in Figure 1. The variables $x_{1:b}$ may be seen as 'proxies' of z associated with each of the data subsets, which are conditionally independent given z and the newly-introduced parameter λ. Loosely speaking, λ represents the extent to which we allow the local variables $x_{1:b}$ to differ from the global variable z.

There are connections between this setup and various concepts in the distributed optimisation literature. The global consensus problem (see Boyd et al., 2011, Section 7 and references therein) is the problem of minimising a sum of functions on a common domain, under the constraint that their arguments are all equal to some global common value. If the Gaussian kernel density $K_\lambda(z, x) = \mathcal{N}(x; z, \lambda)$ is used in (2), then taking the negative logarithm gives

$$-\log \pi_\lambda(z, x_{1:b}) = C - \log \mu(z) - \sum_{j=1}^{b} \log f_j(x_j) + \frac{1}{2\lambda} \sum_{j=1}^{b} (z - x_j)^2 . \quad (4)$$

Maximising $\pi(z)$ is equivalent to minimising this function under the constraint that $z = x_j$ for $j \in \{1, \ldots, b\}$, which may be achieved using the alternating direction method of multipliers (Bertsekas and Tsitsiklis, 1989). Specifically, (4) corresponds to the use of $1/\lambda$ as the penalty parameter in this procedure.

2.1 An MCMC algorithm

The instrumental model described forms the basis of our proposed global consensus framework; 'global consensus Monte Carlo' is correspondingly the application of any Monte Carlo method to form an approximation of the joint density π_λ. One such implementation is the construction of a π_λ-reversible Markov chain on the extended space $\mathsf{E} \times \mathsf{E}^b$. If λ is chosen to be sufficiently small, the z-marginal of π_λ provides an approximation of the true target π. Consequently, given a chain with values denoted $(z^i, x^i_{1:b})$ for $i = 1, \ldots, N$, an estimator of $\int \varphi(z)\pi(z)\mathrm{d}z$ for some bounded function φ is given by

$$\frac{1}{N} \sum_{i=1}^{N} \varphi(z^i). \quad (5)$$

An approach that exploits the distributed setup may be obtained by consid-

ering the conditional densities

$$\pi_\lambda(z \mid x_{1:b}) \propto \mu(z) \prod_{j=1}^{b} K_\lambda(z, x_j)$$

and

$$\pi_\lambda(x_j \mid z) \propto K_\lambda(z, x_j) f_j(x_j); \tag{6}$$

in the latter case we note that the $x_{1:b}$ are mutually conditionally independent given z, by construction. We define $P_1^{(\lambda)}$ to be a π_λ-invariant Markov kernel that fixes the first component z. Denoting by δ_z the Dirac measure at z, in the simplest case one could take the Gibbs kernel

$$P_1^{(\lambda)}((z, x_{1:b}); \mathrm{d}(z', x'_{1:b})) = \delta_z(\mathrm{d}z') \prod_{j=1}^{b} \pi_\lambda(x'_j \mid z)\mathrm{d}x'_j, \tag{7}$$

which corresponds to sampling $x'_{1:b}$ directly from their respective full conditional distributions. More generally, one could allow

$$P_1^{(\lambda)}((z, x_{1:b}); \mathrm{d}(z', x'_{1:b})) = \delta_z(\mathrm{d}z') \prod_{j=1}^{b} R_{j,z}^{(\lambda)}(x_j, \mathrm{d}x'_j),$$

where $R_{j,z}(x_j, \cdot)$ is a $\pi_\lambda(x_j \mid z)$-invariant Markov kernel for each j, z. We similarly define $P_2^{(\lambda)}$ to be a π_λ-invariant Markov kernel that fixes $x_{1:b}$; the simplest choice would be

$$P_2^{(\lambda)}((z, x_{1:b}); \mathrm{d}(z', x'_{1:b})) = \left[\prod_{j=1}^{b} \delta_{x_j}(\mathrm{d}x'_j)\right] \pi_\lambda(z' \mid x_{1:b})\mathrm{d}z'. \tag{8}$$

The simple cases here correspond to a two-variable Gibbs sampler, where the two variables are z and $x_{1:b}$. The interest from a distributed perspective is that $P_1^{(\lambda)}$ can be implemented by having each node j sample from $\pi_\lambda(x_j \mid z)$, sending only $x'_{1:b}$ to a central node that implements $P_2^{(\lambda)}$. A two-variable Gibbs Markov chain has the property that each of the coordinates of the Markov chain is also a reversible Markov chain. In this setting we may therefore consider the z-chain with transition kernel

$$P_{12}^{(\lambda)}(z, \mathrm{d}z') = \int_{\mathsf{E}^b} \left[\prod_{j=1}^{b} \pi_\lambda(x_j \mid z)\right] \pi_\lambda(z' \mid x_{1:b})\mathrm{d}x_{1:b}\mathrm{d}z.$$

Note that depending on the form of μ and $(f_j)_{j=1}^b$, a careful choice of K_λ may allow either of the simplest Markov kernels here to be implemented, sampling directly from the corresponding full conditional densities. To provide a simplistic example, if μ is a Gaussian density, then choosing a Gaussian density for K_λ results in the conditional distribution of z given $x_{1:b}$ being Gaussian; this is further explored in Section 3.

An important consideration when implementing the algorithm is the choice of the parameter λ. Recall that, under certain conditions on the transition

densities K_λ, the z-marginal of π_λ will converge in total variation to π as $\lambda \to 0$. As such, if λ is too large $\pi_\lambda(z)$ may form a poor approximation of $\pi(z)$, possibly introducing large biases to estimators of the form (5).

However, the algorithm as specified may mix poorly if λ is chosen too small, which may result in such estimators having high variance. Considering the case in which $K_\lambda(z, \cdot)$ converges weakly to the Dirac measure at z as $\lambda \to 0$, it is straightforward to show that for any $z' \in \mathsf{E}$ with $f_j(z') > 0$, the conditional distribution of each x_j given $z = z'$ will converge weakly to the Dirac measure at z'. The consequence of this is that when updating each local variable x_j given the global variable z, the influence of each f_j (and therefore the influence of the corresponding block of data) becomes negligible as λ approaches 0. The Gaussian case study in Section 3 exemplifies this, being a case in which the resulting Gibbs sampler is asymptotically deterministic.

It follows that λ should ideally be chosen in order to balance these two considerations, in a bias–variance trade-off. For a given computational cost it may be desirable to choose λ to control the mean squared error of some estimator, and indeed by this measure the global consensus approach may outperform exact schemes. While the existence of this tuning parameter may appear problematic, we show examples in Section 4 in which the algorithm produces similar results for various values of λ chosen across orders of magnitude.

2.2 Comparisons with consensus Monte Carlo

The consensus Monte Carlo algorithm of Scott et al. (2016) has similar aims to our algorithm, and provides a useful benchmark for comparison. This requires b chains to be run in parallel, with the jth chain targeting a 'subposterior' density proportional to $\mu(z)^{1/b} f_j(z)$. Once chains of the required length are generated, a 'consensus chain' is formed by combining the draws from each of the chains, for which a form of weighted averaging is suggested.

There are some similarities between our proposed algorithm and this procedure. The global consensus Monte Carlo algorithm similarly results in a Markov chain for each block (the chain associated with each x_j), and these are somehow 'combined' to form a chain for the global variable z. In our case, this occurs through the application of the Markov kernel $P_2^{(\lambda)}$, invariant with respect to the conditional distribution of z given $x_{1:b}$. To make this explicit, consider the case in which we have the improper prior $\mu(z) \propto 1$, and we use the Gaussian transition densities $K_\lambda(z, x) = \mathcal{N}(x; z, \lambda)$. We have

$$\pi_\lambda(z \mid x_{1:b}) = \mathcal{N}\left(z; \bar{x}, \frac{\lambda}{b}\right),$$

so that the central z-chain is similarly based on (unweighted) averaging across the block-level chains. More general forms of μ provide additional regularisation on the local variables $x_{1:b}$.

Beyond this, there are clear differences between the consensus Monte Carlo procedure of Scott et al. and the framework we propose. In distributed settings, the former only requires communication between computing nodes at the beginning and end of the algorithm, in contrast to our proposed algorithm, which requires more frequent communication. Additionally, the consensus Monte Carlo algorithm is largely motivated by assumptions of Gaussianity – indeed in cases

where the subposterior densities may be assumed to be approximately Gaussian, one would expect it to perform well, with the final 'consensus chain' comprising values that resemble samples from π. If such assumptions can be made, then consensus Monte Carlo may be strongly advantageous compared to our approach – particularly if the time taken for each sample to be drawn is relatively small compared to the communication between computing nodes.

A useful case for analysis of the consensus Monte Carlo algorithm is that in which μ and $(f_j)_{j=1}^b$ are all Gaussian densities. The authors note that if the b chains comprise samples drawn exactly from the subposterior target distributions, then it is possible for the values of the consensus chain to be distributed exactly according to the correct target density. This requires taking weighted averages across each set of draws, using weights proportional to the precision matrices of the subposterior distributions. While this is justified in the Gaussian setting, the application of this approach in more general settings comes with no theoretical guarantees. In contrast, our global consensus procedure is intended to be more robust to cases where few assumptions can be made of the functions f_j.

Based on this Gaussian setting, the authors suggest that for the weights used in the averaging, one could use inverted sample covariance matrices of each chain. Other weighting schemes are possible; the authors use unweighted averages in one example, though note that this choice may produce poor results when the subposterior distributions differ considerably. As earlier noted, other procedures for combining the chains have been proposed in the literature, though these would also be expected to suffer when the f_j differ significantly.

Another fundamental difference in our approach is the treatment of μ. Within the consensus Monte Carlo framework, each subposterior receives an equal share of the prior information encoded by μ, in the form of the fractionated prior density $\mu(z)^{1/b}$. It is not clear, however, when this approach is justified. For example, suppose the prior distribution belongs to an exponential family; any property that is not invariant to multiplying the canonical parameters by a constant will not be preserved when fractionating the density. For several common distributions (including gamma and Wishart), this is true of the first moment. As such if $\mu(z)^{1/b}$ is proportional to a valid probability density function, the corresponding distribution may be qualitatively very different to the full prior. Although Scott et al. (2016) note that fractionated priors perform poorly on one example (for which a tailored solution is provided), there is no other obvious way of assigning prior information to each block that naturally presents itself.

The role of μ in the global consensus algorithm is to provide a form of regularisation at the 'global' level, and so this problem is avoided entirely. As seen in the conditional density function (6) of x_j given z, the Markov transition density $K_\lambda(z, x_j)$ effectively provides a pseudo-prior for x_j, so that while fractionating μ across the blocks is possible, it is unnecessary.

3 Case study: Gaussians

The setting in which μ and $(f_j)_{j=1}^b$ are all Gaussian densities provides a useful case for analysis, and may give insight into the algorithm's performance in many practical applications involving Bayesian posterior densities, due to the Bernstein–von Mises theorem (van der Vaart, 2000).

Suppose that $\mu(z) = \mathcal{N}(z; \mu_0, \sigma_0^2)$ and $f_j(z) = \mathcal{N}(z; \mu_j, \sigma_j^2)$ for $j \in \{1, \ldots, b\}$, so that the target density is

$$\pi(z) = \mathcal{N}\left(z; \frac{\frac{\mu_0}{\sigma_0^2} + \sum_{j=1}^b \frac{\mu_j}{\sigma_j^2}}{\frac{1}{\sigma_0^2} + \sum_{j=1}^b \frac{1}{\sigma_j^2}}, \frac{1}{\frac{1}{\sigma_0^2} + \sum_{j=1}^b \frac{1}{\sigma_j^2}}\right).$$

If we choose $K_\lambda(z, x) = \mathcal{N}(x; z, \lambda)$, then

$$f_j^\lambda(z) = \int_{\mathsf{E}} K_\lambda(z, x) f_j(x) \mathrm{d}x = \mathcal{N}\left(z; \mu_j, \sigma_j^2 + \lambda\right),$$

and so

$$\pi_\lambda(z) = \mathcal{N}\left(z; \frac{\frac{\mu_0}{\sigma_0^2} + \sum_{j=1}^b \frac{\mu_j}{\sigma_j^2+\lambda}}{\frac{1}{\sigma_0^2} + \sum_{j=1}^b \frac{1}{\sigma_j^2+\lambda}}, \frac{1}{\frac{1}{\sigma_0^2} + \sum_{j=1}^b \frac{1}{\sigma_j^2+\lambda}}\right). \tag{9}$$

The full conditional distributions of $\pi_\lambda(z, x_{1:b})$ are given by

$$\begin{aligned}\pi_\lambda(z \mid x_{1:b}) &= \mathcal{N}\left(z; \frac{\frac{\mu_0}{\sigma_0^2} + \sum_{j=1}^b \frac{x_j}{\lambda}}{\frac{1}{\sigma_0^2} + \sum_{j=1}^b \frac{1}{\lambda}}, \frac{1}{\frac{1}{\sigma_0^2} + \sum_{j=1}^b \frac{1}{\lambda}}\right) \\ &= \mathcal{N}\left(z; \frac{\lambda\mu_0}{\lambda + b\sigma_0^2} + \frac{\sigma_0^2}{\lambda + b\sigma_0^2} \sum_{j=1}^b x_j, \frac{\lambda\sigma_0^2}{\lambda + b\sigma_0^2}\right) \end{aligned} \tag{10}$$

and

$$\pi_\lambda(x_j \mid z) = \mathcal{N}\left(x_j; \frac{\frac{z}{\lambda} + \frac{\mu_j}{\sigma_j^2}}{\frac{1}{\lambda} + \frac{1}{\sigma_j^2}}, \frac{1}{\frac{1}{\lambda} + \frac{1}{\sigma_j^2}}\right) = \mathcal{N}\left(x_j; \frac{\sigma_j^2 z + \lambda\mu_j}{\sigma_j^2 + \lambda}, \frac{\lambda\sigma_j^2}{\sigma_j^2 + \lambda}\right).$$

We consider the form of the resulting z-chain. If $P_1^{(\lambda)}$ and $P_2^{(\lambda)}$ are Gibbs kernels as in (7) and (8), then we see that $P_2^{(\lambda)}$ depends on $x_{1:b}$ only through the sum $\sum_{j=1}^b x_j$ obtained via $P_1^{(\lambda)}$, which satisfies

$$\sum_{j=1}^b x_j \mid z \sim \mathcal{N}\left(\sum_{j=1}^b \frac{\sigma_j^2}{\sigma_j^2 + \lambda} z + \sum_{j=1}^b \frac{\lambda\mu_j}{\sigma_j^2 + \lambda}, \sum_{j=1}^b \frac{\lambda\sigma_j^2}{\sigma_j^2 + \lambda}\right).$$

Therefore, the z-chain is defined by the transition kernel

$$\begin{aligned}P_{12}^{(\lambda)}(z, z') = \mathcal{N}\Bigg(z'; &\frac{\lambda\mu_0}{\lambda + b\sigma_0^2} + \frac{\sigma_0^2}{\lambda + b\sigma_0^2}\left[\sum_{j=1}^b \frac{\sigma_j^2}{\sigma_j^2 + \lambda} z + \sum_{j=1}^b \frac{\lambda\mu_j}{\sigma_j^2 + \lambda}\right], \\ &\frac{\lambda\sigma_0^2}{\lambda + b\sigma_0^2} + \left[\frac{\sigma_0^2}{\lambda + b\sigma_0^2}\right]^2 \sum_{j=1}^b \frac{\lambda\sigma_j^2}{\sigma_j^2 + \lambda}\Bigg). \end{aligned} \tag{11}$$

This defines an AR(1) process. Letting

$$\alpha \coloneqq \frac{\sigma_0^2}{\lambda + b\sigma_0^2} \sum_{j=1}^b \frac{\sigma_j^2}{\sigma_j^2 + \lambda},$$

$$C \coloneqq \frac{\lambda\mu_0}{\lambda + b\sigma_0^2} + \frac{\sigma_0^2}{\lambda + b\sigma_0^2} \sum_{j=1}^b \frac{\lambda\mu_j}{\sigma_j^2 + \lambda},$$

we obtain

$$Z_k = C + \alpha Z_{k-1} + W_k, \qquad k > 0,$$

where $W_k \overset{\text{iid}}{\sim} \mathcal{N}\left(0, \frac{\lambda\sigma_0^2}{\lambda+b\sigma_0^2} + \left[\frac{\sigma_0^2}{\lambda+b\sigma_0^2}\right]^2 \sum_{j=1}^b \frac{\lambda\sigma_j^2}{\sigma_j^2+\lambda}\right)$. It follows that the autocorrelation of lag k is given by α^k, $k \geq 0$.

3.1 Asymptotic analysis

Consider a Bayesian posterior distribution based on i.i.d. observations $y_{1:n}$, given by

$$\pi(z) = \mu(z) \prod_{i=1}^n L_i(z);$$

the prior density is $\mu(z) = \mathcal{N}(z; \mu_0, \sigma_0^2)$, with likelihood contributions $L_i(z) = \mathcal{N}(y_i; z, \sigma^2)$. This may be written in the form (1) by grouping the observations into b blocks. For simplicity, assume that b divides n, that each block contains n/b observations, and that the observations are allocated to the blocks sequentially, so that the jth block comprises those y_i for $i \in B_j := \{(j-1)n/b + 1, \ldots, jn/b\}$. Then one may take

$$f_j(z) = \prod_{i \in B_j} L_i(z) \propto \mathcal{N}\left(z; \frac{b}{n}\sum_{i \in B_j} y_i, \frac{b}{n}\sigma^2\right).$$

Suppose the global consensus algorithm is used, with $K_\lambda(z, x) = \mathcal{N}(x; z, \lambda)$, in order to estimate the posterior mean $\int_{\mathsf{E}} z\pi(z)\mathrm{d}z$; we consider the mean and asymptotic variance of the resulting estimator. We shall assume the use of a Gibbs sampler, so that the z-chain behaves according to (11).

A bias is introduced by instead approximating $\int_{\mathsf{E}} z\pi_\lambda(z)\mathrm{d}z$; by comparison with (9) we see that

$$\pi_\lambda(z) = \mathcal{N}\left(z; \frac{\frac{\mu_0}{\sigma_0^2} + \frac{n\bar{y}}{\sigma^2+n\lambda/b}}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2+n\lambda/b}}, \frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2+n\lambda/b}}\right),$$

and so the resulting bias is approximately

$$\frac{\frac{\mu_0}{\sigma_0^2} + \frac{n\bar{y}}{\sigma^2+n\lambda/b}}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2+n\lambda/b}} - \frac{\frac{\mu_0}{\sigma_0^2} + \frac{n\bar{y}}{\sigma^2}}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}} = \frac{n^2\,(\lambda/b)\,\sigma_0^2\,(\mu_0 - \bar{y})}{(\sigma^2 + n\sigma_0^2)\,(\sigma^2 + n\sigma_0^2 + n\lambda/b)}. \tag{12}$$

By the Markov chain central limit theorem, the asymptotic variance of the Markov chain ergodic average is

$$\mathrm{var}(Z_0)\left[1 + 2\sum_{k \geq 1} \mathrm{corr}(Z_0, Z_k)\right], \qquad Z_0 \sim \pi_\lambda.$$

Comparing with the general Gaussian case, we see that the AR(1) process formed by the resulting z-chain has parameter

$$\alpha = \frac{n\sigma^2\sigma_0^2}{(\sigma^2 + n\lambda/b)\,(n\sigma_0^2 + n\lambda/b)};$$

and so this asymptotic variance is therefore found to be

$$\frac{\sigma_0^2\left(\sigma^2+n\lambda/b\right)\left[(n\lambda/b)^2+\left(\sigma^2+n\sigma_0^2\right)(n\lambda/b)+2n\sigma^2\sigma_0^2\right]}{(n\lambda/b)\left(\sigma^2+n\sigma_0^2+n\lambda/b\right)^2}. \tag{13}$$

We shall use these expressions to investigate some of the asymptotic properties of the algorithm. As a caveat, however, estimation of the mean in Gaussian settings may not accurately reflect what happens in more complex settings. For example, if one uses an improper prior then there is no bias for any λ, as seen in (12) with $\sigma_0^2 \to \infty$; this will not be true in general. Additionally, in expressions (12) and (13) for the bias and asymptotic variance, all dependence on λ and b is through their ratio λ/b, so that splitting the data into more blocks has the same effect on the mean squared error of this estimator as reducing λ.

Optimal selection of λ

Consider the problem of choosing λ in this setting, where we take the mean squared error of the posterior mean estimator as the objective to be minimised. Defining $B(\lambda)$ to be the bias as given in (12), we see that this is $\mathcal{O}(\lambda)$; in particular, as $\lambda \to 0$, we have that

$$\frac{B(\lambda)}{\lambda} \to \frac{n^2\sigma_0^2\left(\mu_0-\bar{y}\right)}{b\left(\sigma^2+n\sigma_0^2\right)^2} =: B_\star.$$

Similarly, denoting by $V(\lambda)$ the asymptotic variance (13), we see that this is $\mathcal{O}(1/\lambda)$; as $\lambda \to 0$, we have that

$$\lambda V(\lambda) \to \frac{2b\sigma^4\sigma_0^4}{\left(\sigma^2+n\sigma_0^2\right)^2} =: V_\star.$$

Now suppose that we run the algorithm to obtain a z-chain of length N, where N is large enough that we may approximate the estimator variance by $V(\lambda)/N$. For small λ, the mean squared error of the estimate is given approximately by

$$\left(\lambda B_\star\right)^2 + \frac{1}{N}\frac{V_\star}{\lambda}.$$

This expression is minimised when

$$2\lambda B_\star^2 - \frac{1}{N}\frac{V_\star}{\lambda^2} = 0 \quad \Rightarrow \quad \lambda^3 = \frac{V_\star}{2B_\star^2 N} = \frac{b^3\sigma^4\left(\sigma^2+n\sigma_0^2\right)^2}{n^4 N\left(\mu_0-\bar{y}\right)^2}.$$

The corresponding minimal value of the mean squared error is

$$\frac{3n^{4/3}\sigma^{8/3}\sigma_0^4\left(\mu_0-\bar{y}\right)^{2/3}}{N^{2/3}\left(\sigma^2+n\sigma_0^2\right)^{8/3}},$$

in which the contribution of the variance is twice that of the squared bias.

We see that, for fixed data n, we should scale λ with the number of samples N as $\mathcal{O}(N^{-1/3})$. Alternatively, consider fixing N, and suppose that the number of blocks b increases with n as $\mathcal{O}(n^\alpha)$, for some $\alpha \in [0,1]$. We see that λ should be scaled with the total number of data n as $\mathcal{O}(n^{\alpha-2/3})$; notably, if the total number of blocks is fixed then λ should be decreased for increasing n, but if the total data per block is fixed, then λ should be increased.

Fixed λ, increasing n

We may also consider how these quantities behave for fixed λ, as the total data n increases. Again supposing that the number of blocks b is $\mathcal{O}(n^\alpha)$ for some $\alpha \in [0, 1]$, in this case both the bias (12) and asymptotic variance (13) of the posterior mean estimator are $\mathcal{O}(n^{-\alpha})$. An important consequence of this is that if the MCMC algorithm is run for a specific value of λ, it is necessary for the number of blocks b to increase with n in order for the resulting sequence of estimators to be consistent.

4 Examples

In the following simulation studies, we compare the global consensus Monte Carlo (GCMC) algorithm described in Section 2.1 with consensus Monte Carlo (CMC), as proposed by Scott et al. (2016).

4.1 Log-normal toy example

As noted in Section 2.2, consensus Monte Carlo assigns prior information to each node in the form of the fractionated prior $\mu(z)^{1/b}$, though it is not clear when this is justified. We here demonstrate with a toy example how such treatment of μ can lead to large biases in the resulting estimators.

Let $\mathcal{LN}(x; \mu, \sigma^2)$ denote the density of a log-normal distribution with parameters μ and σ^2; that is,

$$\mathcal{LN}(x; \mu, \sigma^2) = \frac{1}{x\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\log(x) - \mu)^2}{2\sigma^2}\right).$$

One may consider a toy model with prior density $\mu(z) = \mathcal{LN}(z; \mu_0, \sigma_0^2)$, and likelihood contributions $f_j(z) = \mathcal{LN}(z; \mu_j, \sigma_j^2)$ for $j = 1, \ldots, b$. Note that this is identical to the Gaussian setting introduced in Section 3, except for a reparametrisation. For the implementation of the global consensus algorithm, we choose Markov transition kernels given by $K_\lambda(z, x) = \mathcal{LN}(x; \log(z), \lambda)$, which satisfy the necessary asymptotic assumptions as $\lambda \to 0$. This toy example is convenient as it allows for the target distribution π to be expressed analytically (it is also a log-normal distribution). It also allows for exact sampling from all of the full conditional distributions in the GCMC setting, and from all of the subposteriors in the CMC case.

It is practically interesting, however, because of the behaviour of the log-normal density when raised to a power. Log-normal distributions form an exponential family, and so if $\mu(z)$ is a log-normal density, then the fractionated prior $\mu(z)^{1/b}$ is also proportional to a log-normal density. However, the distribution induced by this fractionated prior density will always have a larger mean than the original prior distribution, and this difference may be extreme. For example, suppose $\mu(z) = \mathcal{LN}(z; 0, 1)$, for which the corresponding distribution has a mean of $e^{0.5} \approx 1.65$. The fractionated prior applied to $b = 10$ blocks is then $\mu(z)^{1/10} \propto \mathcal{LN}(z; 9, 10)$, with the corresponding distribution having mean $e^{14} \approx 1.20 \times 10^6$, implying very different prior information.

For our simulations, we use $\mu(z) = \mathcal{LN}(z; 0, 25)$. We take $b = 10$ blocks, with $f_j(z) = \mathcal{LN}(z; \mu_j, 1)$ where the μ_j (representing the 'data') are generated

		Bias	Variance
GCMC	$\lambda = 10^{1}$	7.2×10^{-1}	7.4×10^{-6}
	$\lambda = 10^{0}$	6.0×10^{-2}	$\mathbf{9.7 \times 10^{-7}}$
	$\lambda = 10^{-1}$	6.2×10^{-3}	3.4×10^{-6}
	$\lambda = 10^{-2}$	$\mathbf{1.9 \times 10^{-3}}$	2.9×10^{-5}
	$\lambda = 10^{-3}$	-1.1×10^{-2}	1.2×10^{-4}
	$\lambda = 10^{-4}$	7.2×10^{-2}	1.4×10^{-4}
	$\lambda = 10^{-5}$	-2.8×10^{-2}	1.1×10^{-4}
CMC		6.8×10^{5}	9.5×10^{5}
CMC, with alteration		1.6×10^{0}	2.0×10^{-6}

Table 1: Bias and variance of estimators of $\int z\pi(z)\mathrm{d}z$ for the log-normal toy example, obtained using global consensus Monte Carlo with various values of λ, and consensus Monte Carlo. In each column, the value of smallest magnitude is highlighted in bold.

as independent samples from a standard normal distribution. We consider estimation of the mean $\int z\pi(z)\mathrm{d}z$, for which the true value is approximately 1.20.

Table 1 shows the bias and variance of the estimators resulting from using GCMC for various values of λ, compared with CMC, with 10^6 samples used in each case; note that the variance values here are estimates based on batch means, computed using the `mcmcse` R package (Flegal et al., 2017).

The estimator formed using consensus Monte Carlo is incorrect by many orders of magnitude. GCMC performs far better, for a broad range of choices of λ. Note however that for the largest value of λ shown here ($\lambda = 10$), the bias is around 60% of the true value, since the approximation of π resulting from this choice of λ is poor.

It must be stressed that this is a rather extreme example, and indeed in this case the issues of using a fractionated prior could be solved by a simple reparametrisation of the problem (resulting in a Gaussian model). However, no such straightforward solution may exist in more general settings; it is disadvantageous that the effectiveness of the fractionated prior approach can depend so heavily on the parametrisation of the problem.

No general alternative to using a fractionated prior is forthcoming. In this case however, a possible alteration to the consensus Monte Carlo could be made. Usually, one runs and then combines b chains, each with target density proportional to $\mu(z)^{1/b} f_j(z)$. Instead, one could here run $b + 1$ chains, with the first having target density $\mu(z)$, and the remaining b having target densities proportional to each $f_j(z)$. Results for this are presented in the bottom row of Table 1 – while this performs far better than CMC in its usual form, it is still outperformed by the global consensus approach, likely due to the f_j being highly non-Gaussian. Note also that this *ad hoc* approach is not widely applicable, since in general settings the functions $f_j(z)$ may not be integrable, and so some prior information in each subposterior is necessary.

	True (MCMC)	GCMC, $\lambda = 10^{-2}$	CMC
z_1	-1.975	-1.979	-1.956
z_2	-0.027	-0.025	-0.039
z_3	-0.057	-0.061	-0.044
z_4	-0.064	-0.067	0.145
z_5	0.213	0.214	0.232
z_6	-1.121	-1.149	-0.558

Table 2: For the first logistic regression example, the true posterior mean value for each component of the parameter vector (as estimated from a long MCMC chain). Also presented are mean estimates over 5 runs, for GCMC with $\lambda = 10^{-2}$, and for CMC.

4.2 Logistic regression

Binary logistic regression models are commonly used in settings related to marketing. In web design for example, A/B testing may be used to determine which content choices lead to maximised user interaction, such as the user clicking on a product for sale.

We assume that we have a data set of size n formed of responses $\eta_i \in \{-1, 1\}$, and vectors $\xi_i \in \{0, 1\}^d$ of binary covariates, where $i = 1, \ldots, n$. Following Gelman et al. (2008), we apply a pre-processing step to the covariates, centring those that are not constant in value (i.e. those that do not correspond to an intercept term). Denoting the transformed covariates by $\tilde{\xi}_i \in \mathbb{R}^d$, the likelihood contribution of each block of data takes the form $f_j(z) = \prod_i \sigma(\eta_i z^{\mathsf{T}} \tilde{\xi}_i)$, $z \in \mathbb{R}^d$. Here, the product is taken over those indices i included in the jth block of data; σ denotes the logistic function, $\sigma(x) = (1 + e^{-x})^{-1}$.

For the prior μ, we use the Gaussian prior trialled by Chopin and Ridgway (2017). This is a product of independent zero-mean Gaussians, with standard deviation 20 for the parameter corresponding to the intercept term (if this exists), and 5 for all other parameters. For the Markov transition densities in GCMC, we use multivariate Gaussian densities: $K_\lambda(z, x) = \mathcal{N}(x; z, \lambda I)$.

We investigated several such simulated data sets, with the aim of estimating the posterior mean $\int z\pi(z)\mathrm{d}z$. As would be expected due to the Bernstein–von Mises theorem, in many cases the likelihood terms f_j exhibited near-Gaussianity; the consensus Monte Carlo approach therefore often outperformed global consensus, forming estimators with lower mean squared error. We here present some settings in which the use of global consensus Monte Carlo was observed to deliver clear improvements.

We first describe in example for a data set with $d = 6$ covariates, the first of which corresponds to the intercept term. The data comprise $n = 4096$ values, split into $b = 8$ equally-sized blocks. Each vector of covariates was generated independently, comprising draws from independent Bernoulli distributions. The probability of each covariate being active (i.e. the covariate taking the value 1) was the same for every datum in a given block, but differed between blocks, so that some covariates were more prevalent in some blocks than others. For each vector of covariates, the response was generated from the correct model, for some fixed underlying parameter vector.

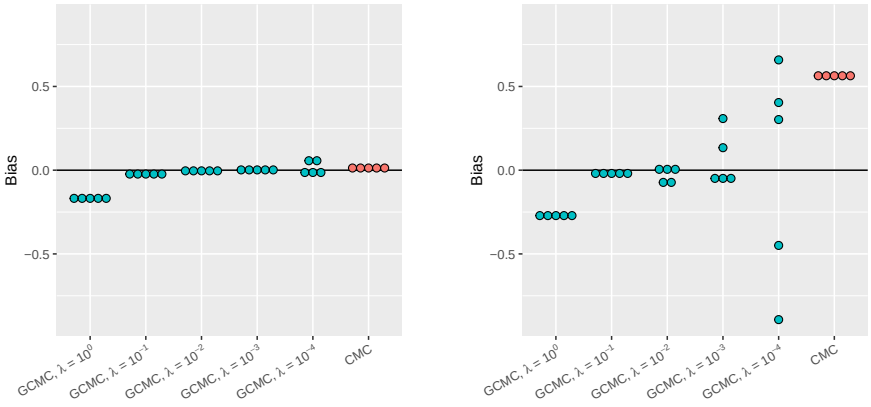

Figure 2: For the first logistic regression example, the bias of estimates of the posterior mean for two components of the parameter vector, z_3 (left) and z_6 (right). GCMC (blue) was run for several values of λ, and is compared with CMC (red); results for 5 runs of each setup are displayed.

To compare the algorithms, we ran a long MCMC chain on the true model to provide the 'ground truth'. We applied GCMC for values of λ between 10^{-4} and 1, and also applied CMC, with 5 runs in each case and $N = 500000$ samples retained after burn-in.

Table 2 shows the 'true' values of the posterior mean for each component of the parameter vector, alongside the mean estimates from GCMC for $\lambda = 10^{-2}$, and for CMC. While the values for GCMC are all close to their true values, CMC does not perform uniformly well for all components. While the CMC estimate for each component's posterior mean was found to be within its true 95% credible interval, we note that estimate for the posterior mean of z_4 has the wrong sign, and the estimate for z_6 is less than half the true value.

Figure 2 shows the biases of estimates of the posterior mean for two components. For component z_4, the effect on the value of λ in GCMC is clearly seen – for a value too large significant bias is introduced, but for a value too small there is increased variance. In contrast, estimation of the posterior mean for z_3 is less problematic for both algorithms; if λ is sufficiently small in the GCMC case then the two algorithms perform comparably.

Exploratory analysis of the data showed that the covariates corresponding to these two components were rarely 'active' (i.e. equal to 1) in some blocks of data; for most observations they were inactive. As such, some blocks were relatively uninformative about the corresponding parameters. In particular for component z_6, there were several blocks in which all those observations with the corresponding covariate being active had the same response. Investigation into the resulting subposteriors used in CMC showed that they had high variance and a large skew, and so the Gaussian assumptions on which it relies do not hold. This may also have contributed to the high estimator variance resulting from small λ values in GCMC.

To investigate these properties further, we constructed a second data set of size $n = 3000$ with $d = 4$ covariates (including an intercept), split into $b = 3$ blocks of equal size. While the sets of covariate vectors in each block were

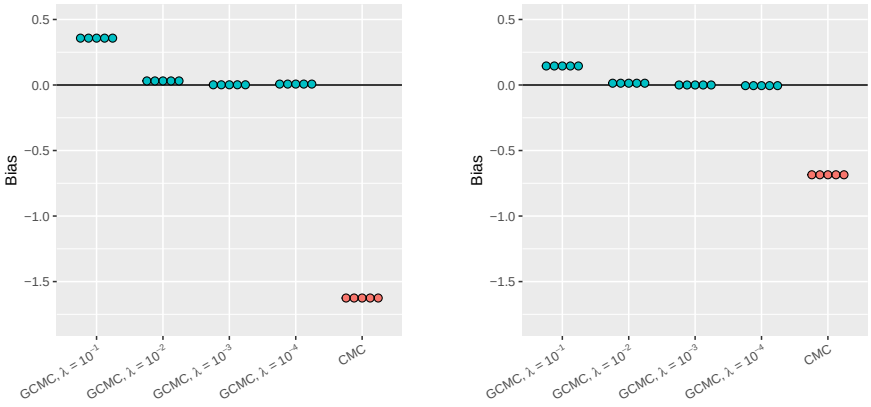

Figure 3: For the second logistic regression example, the bias of estimates of the posterior mean for two components of the parameter vector, z_3 (left) and z_4 (right). GCMC (blue) was run for several values of λ, and is compared with CMC (red); results for 5 runs of each setup are displayed.

identical, we artificially generated the responses in order that, of those observations with at least one 'active' covariate, a very high proportion had a positive response. The biases of the posterior mean estimates for two components are shown in Figure 3. In this case, GCMC worked well for a wide range of λ values, while CMC introduced significant biases; indeed in this case, the mean estimator for z_3 (depicted left) was below the 2.5% quantile of its true posterior distribution.

While this last example was constructed specifically to emphasise this behaviour, we note that the features exhibited by these synthetic data sets may be presented in real data. The manner in which the data are partitioned into blocks may be pre-determined, perhaps due to the way in which the data were collected and stored. In such cases, there may be differences between the data in each block – some covariates may be more commonly 'active' in some blocks, and not present at all in others – and this will lead to differences in the likelihood contributions f_j. As a further note, such settings may lead to some parameter components not being identifiable within a block of the data. For example, this may occur if a covariate is not at all active in a particular block, even if it is identifiable within the entire data set.

For large distributed data sets, permuting and re-partitioning the data may not be a computationally feasible solution. Even if this were possible, this would likely have not solved the problem in the second example here, were a very large number of observations in each block had a common response. Similarly, if a particular covariate is rarely observed in its active state in the full (unpartitioned) data set, then the same is likely to be true in several blocks in a random partition of the data, with the corresponding likelihood contributions f_j possibly being poorly approximated by Gaussians.

5 Topics for further investigation

5.1 SMC implementation

As an alternative to implementing the global consensus algorithm for a specific choice of λ, one could choose to generate samples from π_λ for several values of λ in a sequence. Sequential Monte Carlo (SMC) describes a class of algorithms that employ sequential importance sampling and resampling to generate sequences of weighted empirical measures. Having first been introduced for the problem of particle filtering for hidden Markov models (Gordon et al., 1993), they have since found far wider application; Doucet and Johansen (2011) provide a detailed summary of their uses and properties.

In order for such methods to be used to produce samples from each in an arbitrary sequence of target distributions, Del Moral et al. (2006) proposed a methodology referred to as SMC sampling. By artificially defining a sequence of backward-in-time Markov kernels, one may construct a sequence of distributions on spaces of increasing dimension, which admit each of the distributions of interest as marginals. Application of sequential importance sampling and resampling to this sequence of auxiliary distributions allows collections of weighted particles to be formed, and the resulting weighted empirical measures approximate each of the target distributions.

Specifically, suppose we wish to generate samples according to each distribution $\eta_0, \eta_1, \ldots$ in a given sequence. We commence by generating N particles $(\zeta_0^i)_{i=1}^N$ from η_0, to which each is assigned the equal weight $W_0^i = 1$. Thereafter, suppose we have a collection of particles $(\zeta_{p-1}^i)_{i=1}^N$ and associated weights $(W_{p-1}^i)_{i=1}^N$, forming an approximation of η_{p-1}. In order to form an approximation of η_p, we form new weights $(W_p^i)_{i=1}^N$ by multiplying each of the existing weights W_{p-1}^i by some incremental weight w_p^i. A η_p-invariant Markov kernel M_p is then applied to each of the particles in order to generate a new collection of particles $(\zeta_p^i)_{i=1}^N$. In order to prevent particle degeneracy, a resampling step is occasionally carried out immediately before the application of the Markov kernel.

The form of the incremental weights depends on the backward-in-time Markov kernels used in the construction of the SMC sampler. A typical choice is to take these to be the time-reversals of the Markov kernels, leading to incremental weights of a form proposed by Gilks and Berzuini (2001). Supposing that each distribution η_p has a density that can be computed only up to some normalising constant, say $\eta_p(x) = \gamma_p(x)/Z_p$, one may in this case express the incremental weights simply as $w_p^i = w_p(\zeta_{p-1}^i)$, where

$$w_p(\zeta) = \frac{\gamma_p(\zeta)}{\gamma_{p-1}(\zeta)}.$$

In the case of global consensus Monte Carlo, we have as previously noted that the z-marginal of the target π_λ will converge in total variation to the true target π as $\lambda \to 0$, though we would in many cases expect the mixing of the algorithm to worsen for decreasing λ. One could therefore use an SMC sampler to produce approximations of a sequence of distributions $\pi_{\lambda_0}, \pi_{\lambda_1}, \ldots \pi_{\lambda_n}$, where the sequence of λ values is decreasing, beginning with some large λ_0 and ending with λ_n close to 0. Given an initial sample from π_{λ_0}, which should place particles

over a large part of the state space, the application of each π_{λ_p}-invariant Markov kernel in turn will gradually move these particles to areas of higher mass, until one eventually has a particle approximation of π_{λ_n}. This may result in better mixing, and better exploration of the state space, compared to directly running a π_{λ_n}-invariant MCMC chain.

To make this explicit, the particles in such an implementation would be of the form $(z, x_{1:b}) \in \mathsf{E} \times \mathsf{E}^b$, and a Markov kernel invariant with respect to each π_{λ_p} may be constructed in the manner earlier described. The incremental weights in this case take the convenient form

$$w_p(z, x_{1:b}) = \frac{\prod_{j=1}^{b} K_{\lambda_p}(z, x_j)}{\prod_{j=1}^{b} K_{\lambda_{p-1}}(z, x_j)}; \tag{14}$$

this is independent of the functions f_j, and therefore the weights may be evaluated on a central computing node given the particles' current values. With regard to the initial distribution π_{λ_0}, if it is not possible to draw from this directly, one could for example use samples obtained by running a thinned π_{λ_0}-invariant Markov chain, or an importance sample.

Regression on λ

Suppose one wishes to estimate the expectation of some function φ with respect to π. If samples may be drawn from a sequence of distributions $\pi_{\lambda_0}, \pi_{\lambda_1}, \ldots \pi_{\lambda_n}$ in this way, one could form estimates of the form (5) using each of these approximations of the true target. From these, one could then regress on the parameter λ, extrapolating to obtain an estimate of the true value of the expectation (obtained when $\lambda = 0$). Such an approach could avoid the biases that may result from using a single value of λ; however, any such extrapolation procedure may be affected by possible poor mixing as λ approaches 0, for which the resulting estimators may have high variance. Investigation into such a possible approach remains outstanding.

Automatic selection of λ

It may be noted that the SMC sampler described allows for adaptive selection of the sequence of λ values. Specifying the sequence of distributions for an SMC sampler in an adaptive manner is commonly applied in the setting of tempering, though such approaches are also applicable here. Given a particle approximation of $\pi_{\lambda_{p-1}}$, the incremental weights (14) used in the next time step may be considered as a function of λ_p; one may therefore select λ_p (and therefore π_{λ_p}) in order to control some property of the resulting particle weights. For example, Jasra et al. (2011) propose a procedure that controls the decay of the particles' effective sample size, while a more recent proposal of Zhou et al. (2016) aims to control directly the dissimilarity between successive distributions. Either procedure could be used in this case to generate a sequence of distributions until some stopping rule is satisfied.

An adaptive SMC sampler of this nature could form the basis of a procedure to automatically select an appropriate value of λ for use in a given setting. To achieve this, one may consider procedures that have been proposed in the literature for estimating the variances of approximations formed using SMC methods.

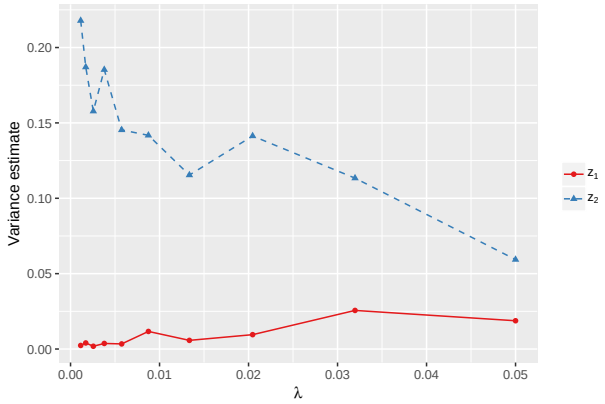

Figure 4: Estimated variance values for the posterior mean of two components of the parameter vector z in a logistic regression example. Note that in the SMC sampler for which these values were produced, the sequence of λ values used was decreasing, beginning with $\lambda_0 = 0.05$; that is, these values were generated 'from right to left'.

Given an SMC algorithm used to form an estimator of a given expectation, one could assess the variance of this estimator by producing multiple i.i.d. replicates, obtained from repeated runs of the algorithm. Since this incurs high computational cost, several approaches have been proposed to estimate these variances using the output of a single run of the algorithm. A first such procedure was proposed by Chan and Lai (2013), giving a consistent asymptotic variance estimator for hidden Markov model setting. A collection of estimators for use in broader settings was later proposed by Lee and Whiteley (2016), of which the former is a special case. More recent work by Olsson and Douc (2017) has focused on avoiding numerical instabilities that these estimators may exhibit, as the set of particles degenerates.

In order to choose a value of λ in an automated way, one could therefore run an SMC sampler with adaptive selection of the sequence of λ values. After each iteration, the required estimator is formed, and its variance estimated using one of the aforementioned methods. By assessing these values as λ decreases, one could form some criterion to determine when a 'good' value of λ has been obtained – that is, when the value of the estimator has converged, and its variance is sufficiently low.

If one wishes to use the MCMC algorithm to form the required estimator, then this procedure could be carried out as an initial step in order to choose a suitable value of λ. That is, one could run this initial SMC procedure with relatively few particles, in order to determine a value of λ that may be used in the MCMC algorithm, with which the desired estimator may be formed.

Some initial investigations into a possible procedure have been carried out so far, but significant further investigation is required. To provide an example, the SMC implementation was applied to a small logistic regression model, on the parameter space $\mathsf{E} = \mathbb{R}^4$. Given an initial λ value of $\lambda_0 = 0.05$, samples were drawn from each of a sequence of distributions $\pi_{\lambda_0}, \pi_{\lambda_1}, \ldots \pi_{\lambda_n}$, with $n = 9$. The decreasing sequence of λ values was chosen adaptively, via the procedure

of Zhou et al. (2016). After each iteration, the posterior mean was estimated, with the asymptotic marginal variances approximated by one of the estimators of Lee and Whiteley (2016).

Figure 4 illustrates these asymptotic marginal variance estimates for two components of the resulting estimator. Note that the asymptotic variance of component z_2 appears to increase as λ approaches 0, in line with the asymptotic results for Gaussian distributions previously presented; the same was observed to be true of components z_3 and z_4 (not depicted). However, the opposite is true of z_1, which appears to have decreasing variance as λ becomes small. The general behaviour of such variances for varying λ remains unclear; further analysis is required in order that a robust parameter selection procedure may be established.

5.2 Assigning a regularisation parameter to each block

Considering the form (2) of the joint density at $(z, x_{1:b})$, a natural generalisation is to allow a different regularisation parameter λ_j to be assigned to each block of the data. The resulting joint density may then be expressed as

$$\pi_{\lambda_{1:b}}(z, x_{1:b}) \propto \mu(z) \prod_{j=1}^{b} K_{\lambda_j}(z, x_j) f_j(x_j). \tag{15}$$

This general form represents an alteration to the instrumental hierarchical model that was earlier proposed. Considering the local variables x_j as perturbed versions of the global variable z, this allows us to encode prior beliefs that the relative sizes of these perturbations should be different in magnitude, perhaps due to prior knowledge of differences between the blocks of data.

In an MCMC implementation this would result in the necessity of tuning each parameter λ_j independently; however, removing the constraint that the tuning parameter for each block must be equal may result in significant improvements. Indeed, given an automatic approach to selecting the regularisation parameters such as those considered in the previous section, it may be possible to extend it to this setting.

An alternative that has some connections to the above suggestion is to use a different family of transition kernels for each block. Formally, suppose that $\{K_{j,\lambda} : \lambda \in \mathbb{R}_+\}$ is a family of Markov transition densities for each $j = 1, \ldots, b$. Then we could consider the joint density

$$\pi_{\lambda}(z, x_{1:b}) \propto \mu(z) \prod_{j=1}^{b} K_{j,\lambda}(z, x_j) f_j(x_j). \tag{16}$$

This maintains the same hierarchical model interpretation given in Section 2, with one single top-level parameter λ, though suggests that we believe *a priori* that the perturbations of x_j about z should be qualitatively different in some way.

To provide a simple example of this setting, consider a vector of positive real numbers $c_{1:b}$. For some common family of Markov transition densities $\{K_\lambda : \lambda \in \mathbb{R}_+\}$, one could take $K_{j,\lambda} \coloneqq K_{\lambda c_j}$. Note that this is a specific case of (15) in which we fix the relative sizes of the block-level regularisation parameters λ_j.

To provide an example of why this might be advantageous, consider the Gaussian case introduced in Section 3, but with the use of the improper prior $\mu(z) \propto 1$ (obtained by taking $\sigma_0^2 \to \infty$). The true target density in this case is

$$\pi(z) = \mathcal{N}\left(z; \frac{\sum_{j=1}^{b} \frac{\mu_j}{\sigma_j^2}}{\sum_{j=1}^{b} \frac{1}{\sigma_j^2}}, \frac{1}{\sum_{j=1}^{b} \frac{1}{\sigma_j^2}}\right).$$

By using $K_{j,\lambda}(z, x) \coloneqq \mathcal{N}(x; z, \lambda c_j)$ as the Markov transition density for each block in (16), the resulting z-marginal of π_λ has density

$$\pi_\lambda(z) = \mathcal{N}\left(z; \frac{\sum_{j=1}^{b} \frac{\mu_j}{\sigma_j^2+\lambda c_j}}{\sum_{j=1}^{b} \frac{1}{\sigma_j^2+\lambda c_j}}, \frac{1}{\sum_{j=1}^{b} \frac{1}{\sigma_j^2+\lambda c_j}}\right).$$

Notably, if one takes $c_j = \sigma_j^2$ for all $j = 1, \ldots, b$, we notice that this distribution has exactly the same mean as the true target density (albeit with a larger variance), allowing unbiased estimation of the mean $\int z\pi(z)\mathrm{d}z$. This suggests that choosing the regularisation parameter of each block to be proportional to the 'variance' of f_j may be beneficial; however, these 'variances' are not known *a priori*. Additionally, if unbiased estimation of the mean is desirable, it is unclear how to choose the relative sizes of the regularisation parameters if the prior μ takes a more general form.

As another example, consider the multidimensional state space $\mathsf{E} = \mathbb{R}^d$. Defining a collection of positive semi-definite matrices $\Psi_{1:b}$, one could take $K_{j,\lambda}(z, x) \coloneqq \mathcal{N}(x; z, \lambda\Psi_j)$, thereby giving greater control of the covariance structure of each x_j given z. Similarly to the one-dimensional Gaussian case, if one has the improper prior $\mu(z) \propto 1$, and $f_j(z) \coloneqq \mathcal{N}(z; \mu_j, \Sigma_j)$ for $j = 1, \ldots, b$, then choosing $\Psi_j \coloneqq \Sigma_j$ for all j results in unbiased estimation of the mean.

Following this last idea, empirical experiments have been carried out in non-Gaussian settings, to assess the use of Markov transition kernels having a similar covariance structure to that of the likelihood terms f_j. These investigations have so far proved inconclusive: while in some more benign cases this has resulted in a slight improvement over the general setting, in others it has resulted in poorer performance. A possible factor in this is that generally one does not have prior knowledge of the covariance structure of the f_j, and so such tuning would require one to estimate this covariance structure in advance. We have considered using the inverse of the observed Fisher information (in the spirit of the Bernstein–von Mises theorem), which one would expect to be less effective if the f_j are not approximately Gaussian. Indeed, such an approach is unlikely to effective or even feasible in many settings of interest.

5.3 Concluding remarks

The framework presented here describes a novel approach to sampling in distributed settings, of which the MCMC algorithm described is one possible implementation. Much work to date has involved experimentation with various models on simulated data sets, in order to determine the settings in which one would expect it to outperform other algorithms for use in this setting; the examples presented in this report cover only a small subset of those that have so far been considered. Having used these simple examples to gain insight, we

intend to explore the algorithm's behaviour on more complex models in the near future.

While the consensus Monte Carlo algorithm provides a useful benchmark to compare against, as previously noted there are a number of other related approaches to sampling in this setting. Some of the advantages that global consensus has over consensus Monte Carlo would be expected to apply equally to other embarrassingly parallel algorithms – the treatment of the prior, for example. Further simulation studies are required, however, in order to fully compare our framework with some of these alternatives.

The primary motivation behind consensus Monte Carlo and other embarrassingly parallel algorithms is the need to minimise the computational cost of communication between computing nodes. In order for a fully fair comparison to be between our framework and such other approaches, it will be necessary for future work to take this into account. Realistically, we expect that our algorithm will be most effective compared to embarrassingly parallel approaches when the likelihood terms are expensive to evaluate – in such cases, the overhead costs of communication can be relatively small in comparison to the cost of each MCMC kernel application. For given computational resources, this may allow our algorithm to outperform many embarrassingly parallel approaches, with regard to minimising the mean squared error of resulting estimators.

References

Bardenet, R., Doucet, A., and Holmes, C. (2014). Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 405–413.

Bertsekas, D. P. and Tsitsiklis, J. N. (1989). *Parallel and Distributed Computation: Numerical Methods*, volume 23. Prentice-Hall.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122.

Chan, H. P. and Lai, T. L. (2013). A general theory of particle filters in hidden Markov models and some applications. *The Annals of Statistics*, 41(6):2877–2904.

Chopin, N. and Ridgway, J. (2017). Leave Pima Indians alone: binary regression as a benchmark for Bayesian computation. *Statistical Science*, 32(1):64–87.

Dean, J. and Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436.

Doucet, A. and Johansen, A. M. (2011). A tutorial on particle filtering and smoothing: Fifteen years later. In Crisan, D. and Rozovskii, B., editors, *Handbook of Nonlinear Filtering*. Cambridge University Press.

Flegal, J. M., Hughes, J., Vats, D., and Dai, N. (2017). *mcmcse: Monte Carlo Standard Errors for MCMC*. Riverside, CA, Denver, CO, Coventry, UK, and Minneapolis, MN. R package version 1.3-2.

Gelman, A., Jakulin, A., Pittau, M. G., and Su, Y.-S. (2008). A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics*, 2(4):1360–1383.

Gilks, W. R. and Berzuini, C. (2001). Following a moving target – Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(1):127–146.

Gordon, N. J., Salmond, D. J., and Smith, A. F. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *Radar and Signal Processing, IEE Proceedings F*, volume 140, pages 107–113. IET.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Jasra, A., Stephens, D. A., Doucet, A., and Tsagaris, T. (2011). Inference for Lévy-driven stochastic volatility models via adaptive sequential Monte Carlo. *Scandinavian Journal of Statistics*, 38(1):1–22.

Korattikara, A., Chen, Y., and Welling, M. (2014). Austerity in MCMC land: Cutting the Metropolis-Hastings budget. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*.

Lee, A. and Whiteley, N. (2016). Variance estimation in the particle filter. *arXiv preprint arXiv:1509.00394*.

Maclaurin, D. and Adams, R. P. (2014). Firefly Monte Carlo: Exact MCMC with subsets of data. In *Proceedings of the 30th International Conference on Uncertainty in Artificial Intelligence (UAI-14)*, pages 543–552.

Minsker, S., Srivastava, S., Lin, L., and Dunson, D. (2014). Scalable and robust Bayesian inference via the median posterior. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1656–1664.

Neiswanger, W., Wang, C., and Xing, E. (2014). Asymptotically exact, embarrassingly parallel MCMC. In *Proceedings of the 30th International Conference on Uncertainty in Artificial Intelligence (UAI-14)*, pages 623–632.

Olsson, J. and Douc, R. (2017). Numerically stable online estimation of variance in particle filters. *arXiv preprint arXiv:1701.01001*.

Rabinovich, M., Angelino, E., and Jordan, M. I. (2015). Variational consensus Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 1207–1215.

Scott, S. L. (2017). Comparing consensus Monte Carlo strategies for distributed Bayesian computation. *Brazilian Journal of Probability and Statistics*.

Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I., and McCulloch, R. E. (2016). Bayes and big data: The consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management*, 11(2):78–88.

Srivastava, S., Cevher, V., Dinh, Q., and Dunson, D. (2015). WASP: Scalable bayes via barycenters of subset posteriors. In *Artificial Intelligence and Statistics*, pages 912–920.

van der Vaart, A. W. (2000). *Asymptotic Statistics*, volume 3 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press.

Wang, X. and Dunson, D. B. (2013). Parallelizing MCMC via Weierstrass sampler. *arXiv preprint arXiv:1312.4605*.

Wang, X., Guo, F., Heller, K. A., and Dunson, D. B. (2015). Parallelizing MCMC with random partition trees. In *Advances in Neural Information Processing Systems*, pages 451–459.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688.

Williams, D. (1991). *Probability with Martingales*. Cambridge University Press.

Xu, M., Lakshminarayanan, B., Teh, Y. W., Zhu, J., and Zhang, B. (2014). Distributed Bayesian posterior sampling via moment sharing. In *Advances in Neural Information Processing Systems*, pages 3356–3364.

Zhou, Y., Johansen, A. M., and Aston, J. A. (2016). Towards automatic model comparison: an adaptive sequential Monte Carlo approach. *Journal of Computational and Graphical Statistics*, 25(3):701–726.