HHS Public Access

Author manuscript

Stat Comput. Author manuscript; available in PMC 2020 January 01.

Published in final edited form as:

Stat Comput. 2019 January ; 29(1): 23–32. doi:10.1007/s11222-017-9791-1.

Double-Parallel Monte Carlo for Bayesian Analysis of Big Data

Jingnan Xue [Graduate Student] and

Department of Statistics, Texas A&M University, College Station, TX 77843

Faming Liang [professor]

Department of Statistics, Purdue University, West Lafayette, IN 47907

Abstract

This paper proposes a simple, practical and efficient MCMC algorithm for Bayesian analysis of big data. The proposed algorithm suggests to divide the big dataset into some smaller subsets and provides a simple method to aggregate the subset posteriors to approximate the full data posterior. To further speed up computation, the proposed algorithm employs the population stochastic approximation Monte Carlo (Pop-SAMC) algorithm, a parallel MCMC algorithm, to simulate from each subset posterior. Since this algorithm consists of two levels of parallel, data parallel and simulation parallel, it is coined as “Double Parallel Monte Carlo”. The validity of the proposed algorithm is justified mathematically and numerically.

Keywords

Embarrassingly Parallel; Divide-and-Combine; MCMC; Pop-SAMC; Subset Posterior Aggregation

1 Introduction

The MCMC method has proven to be a very powerful tool for analyzing data of complex structures. However, it is difficult to be applied to big data problems for which complex models are often needed. The difficulty comes from two aspects. The first one is on data storage; the dataset can be too large for a single computer to store and process. The second one is on computational time; the MCMC method can be very time consuming for simulating from the posterior of a large data set, which typically requires a large number of iterations and a complete scan of the full dataset for each iteration. However, thanks to the strategy of embarrassingly parallel computing, the two issues can now be solved simultaneously.

The strategy of embarrassingly parallel computing is to divide a large dataset into a number of smaller subsets such that each subset can be stored in a single machine, and then conduct the Bayesian analysis for each subset separately. Finally, the posterior samples generated for each subset are aggregated in some way such that correct inference can be made for the full data posterior. During the past few years, this strategy has been pursued by a few groups

Correspondence to: Faming Liang.

enthusiastically, and several algorithms have been developed to address the issue of subset posterior aggregation.

To be a little more detailed, suppose that a large dataset has been partitioned into k subsets, and N posterior samples have been generated for each subset. Let $\{\boldsymbol{\theta}_1^{(i)}, \ldots \boldsymbol{\theta}_N^{(i)}\}$ denote the posterior samples generated from subset i. Based on the Bernstein-von Mises theorem, which states that the posterior tends to a normal distribution centered around the true parameter value $\boldsymbol{\theta}^*$ as the number of observations grows, Scott *et al.* (2016) proposed to use the weighted average $\sum_{i=1}^{k} w_i \boldsymbol{\theta}_j^{(i)}$ to approximate a full data posterior sample, where the weight w_i is the inverse of the covariance matrix of $\{\boldsymbol{\theta}_1^{(i)}, \ldots \boldsymbol{\theta}_N^{(i)}\}$. This algorithm is exact when the subset posterior is Gaussian. Based on the same theory, Neiswanger *et al.* (2014) proposed to fit the posterior samples generated for each subset by a Gaussian density, denoted the fitted density by $\hat{p}_i$ for $i = 1, \ldots, k$, and then draw samples from the product density $\hat{p}_1 \ldots \hat{p}_k$. As an extension of this approach, Neiswanger *et al.* (2014) proposed to estimate the subset posterior density using a Gaussian kernel density estimation method or a semiparametric density estimation method. Another method that makes use of kernel approximation is by Minsker *et al.* (2014), where the subset posteriors are combined by estimating a probability distribution that minimizes a loss function defined in the reproducing kernel Hilbert space embedding the subset posteriors. In addition, Wang and Dunson (2013) proposed a Weierstrass refinement sampler, where the samples from an initial approximation to the full data posterior (e.g., obtained via variational approximation or other methods) are refined using the information obtained from the subset posterior samples within a Weierstrass approximation. Later, Wang *et al.* (2015) proposed to combine the subsets using random partition trees. These methods generally work well, but their accuracy can vary significantly depending on how close the subset posteriors are to Gaussian, or the choice of kernel and its bandwidth, or the accuracy of density estimators for the subset posteriors, or whether the subset posteriors have non-overlapping supports. In particular, their accuracy can be low when the dimension of $\boldsymbol{\theta}$ is high. Quite recently, the so-called WASP method was proposed by Srivastava *et al.* (2015a) and Srivastava *et al.* (2015b), where each subset posterior is approximated by an empirical measure and they are combined by estimating their barycenter in the Wasserstein space of probability measures. This method does not depend on the kernel density estimation any more, but computing the Wasserstein barycenter needs to solve a huge linear programming problem which often requires a lot of computer memory. To resolve the computational problem, Li *et al.* (2017) proposed a simple and general posterior interval estimation algorithm (PIE) by averaging quantiles estimated from each subset. However, this method was devised to estimate certain quantiles and can not be directly used to estimate the whole distribution.

In this paper, we propose a new method for aggregating subset posterior samples. The new method is surprisingly simple, which is to first simulate from some modified subset posteriors, for which the log-likelihood functions are appropriately scaled according to their sample size, and then recenter the subset posterior samples to their global mean. Under mild conditions, we show that the aggregated samples have the same convergence rate toward the true parameter $\boldsymbol{\theta}^*$ as those drawn from the full data posterior. The numerical results indicate

that the new method can be rather accurate compared to the existing ones. In order to further speed up computation, we suggest to use the Pop-SAMC algorithm(Song *et al.*, 2014), rather than traditional single chain MCMC algorithms, to draw samples from each subset posterior. Since the proposed method consists of two levels of parallel, data parallel and simulation parallel, it is coined as "Double Parallel" Monte Carlo.

The remainder of this article is organized as follows. Section 2 presents the proposed sample aggregation method and describes its theoretical properties. Section 3 first gives a brief review of the pop-SAMC algorithm, and then discusses the Double Parallel strategy. Sections 4 and 5 present some numerical results along with comparisons with some existing methods. Section 6 concludes the article with a brief discussion.

2 Subset Posterior Aggregation

Suppose that a random sample $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ has been collected from the distribution $f(x|\boldsymbol{\theta}^*)$, where $\boldsymbol{\theta}^* \in \Theta \subset \mathbb{R}^p$ and Θ is the parameter space. Let $g(\boldsymbol{\theta})$ denote the prior distribution of $\boldsymbol{\theta}$. Then the posterior distribution of $\boldsymbol{\theta}$ is given by

$$\pi(\boldsymbol{\theta}|\boldsymbol{X}) = \frac{\prod_{i=1}^{n} f(X_i|\boldsymbol{\theta}) g(\boldsymbol{\theta})}{\int_{\Theta} \prod_{i=1}^{n} f(X_i|\boldsymbol{\theta}) g(\boldsymbol{\theta}) d\boldsymbol{\theta}}. \quad (1)$$

In most cases, $\pi(\boldsymbol{\theta}|\boldsymbol{X})$ is analytically intractable and we have to approximate it using the Markov chain Monte Carlo(MCMC) method. However, as mentioned previously, when n is very large, the MCMC method is computationally prohibitive because it requires a large number of scans of the dataset.

To address this issue, we divide the data into k subsets, each containing about the same number of samples. Let $\boldsymbol{X}_{[j]} = (X_{j1}, \ldots, X_{jm_j})$ denote the jth subset, where m_j denotes the sample size of $\boldsymbol{X}_{[j]}$. Let $\pi(\boldsymbol{\theta}|\boldsymbol{X}_{[j]})$ denote the posterior distribution corresponding to the subset $\boldsymbol{X}_{[j]}$, for which the variance is approximately n/m_j times the variance of the full data posterior $\pi(\boldsymbol{\theta}|\boldsymbol{X})$. To adjust the variance, for each subset, we instead work on a modified subset posterior

$$\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X}_{[j]}) = \frac{\prod_{i=1}^{m_j} f^{n/m_j}(X_{ji}|\boldsymbol{\theta}) g(\boldsymbol{\theta})}{\int_{\Theta} \prod_{i=1}^{m_j} f^{n/m_j}(X_{ji}|\boldsymbol{\theta}) g(\boldsymbol{\theta}) d\boldsymbol{\theta}}, \quad (2)$$

where each sample is duplicated n/m_j times. Such a modification, first introduced in Minsker *et al.* (2014), ensures that $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X}_{[j]})$ has about the same variance as the full data posterior. In what follows, we refer to $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X}_{[j]})$ as a subposterior and, without loss of generality, assume that $m_1 = m_2 = \cdots = m_k = m$ holds.

Let $\boldsymbol{\mu}^{(1)},\cdots,\boldsymbol{\mu}^{(k)}$ denote the mean vectors of the respective subposteriors, and let $\hat{\boldsymbol{\mu}} = \frac{1}{k}\sum_{j=1}^{k}\boldsymbol{\mu}^{(j)}$ denote their averages. We propose to recenter each of the subposteriors to $\hat{\boldsymbol{\mu}}$ and then use the following mixture of re-centered subposteriors to approximate the full data posterior $\pi(\boldsymbol{\theta}|\boldsymbol{X})$:

$$\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X}) = \frac{1}{k}\sum_{j=1}^{k}\tilde{\pi}(\boldsymbol{\theta} - \hat{\boldsymbol{\mu}} + \boldsymbol{\mu}^{(j)}|\boldsymbol{X}_{[j]}). \quad (3)$$

To quantify the accuracy of the approximation, we make the following assumptions:

A1 The log-likelihood function $\sum_{i=1}^{n}\log f(X_i|\boldsymbol{\theta}) = -nu_n(\boldsymbol{\theta})$ is six-times continuously differentiable on Θ and has local minima $\{\hat{\boldsymbol{\theta}}_n : n = 1, 2, \ldots\}$. Let $B_\delta(\boldsymbol{\theta})$ denote an open ball of radius δ centered at $\boldsymbol{\theta}$, and let $D^2u_n(\boldsymbol{\theta})$ denote the Hessian of u_n at $\boldsymbol{\theta}$. Further suppose there exist positive numbers ε, M, η and an integer n_0 such that when $n \geq n_0$,

- **i.** For all $\boldsymbol{\theta} \in B_\varepsilon(\hat{\boldsymbol{\theta}}_n)$ and all $1 \leq j_1,\ldots, j_d \leq p$ with $0 \leq d \leq 6$, $|\partial_{j_1,\ldots,j_d} u_n(\boldsymbol{\theta})| < M$.
- **ii.** $det(D^2u_n(\hat{\boldsymbol{\theta}}_n)) > \eta$.
- **iii.** for all δ for which $0 < \delta < \varepsilon$, $B_\delta(\hat{\boldsymbol{\theta}}_n) \subseteq \Theta$ and

$$\lim_{n\to\infty}\sup_{\boldsymbol{\theta}\in\Theta\setminus B_\delta(\hat{\boldsymbol{\theta}}_n)}\{u_n(\hat{\boldsymbol{\theta}}_n) - u_n(\boldsymbol{\theta})\} < 0.$$

A2 $\boldsymbol{\theta}^*$ is an interior point of Θ, $g(\boldsymbol{\theta}^*) > 0$, and $g(\boldsymbol{\theta})$ is four times continuous differentiable on Θ. In addition, $\int_\Theta g(\boldsymbol{\theta})\|\boldsymbol{\theta}\|_2^2\exp\{-nu_n(\boldsymbol{\theta})\}d\boldsymbol{\theta} < \infty$ for $n = 1, 2,\ldots$

A3 The number of subsets k can increase slowly with n, but should be at the rate of $o(n^{1/2})$.

Since the quantification involves posterior expansions based on Laplace's method, the Laplace regularity condition (A1) is assumed. Refer to Kass *et al.* (1990) for more details. This condition is standard and generally holds for the exponentially family. Under the above conditions, we have the following theorem, whose proof is given in the appendix.

Theorem 1

If the conditions A1–A3 are satisfied, then we have

$$E_{\tilde{\pi}}(\boldsymbol{\theta}) - E_\pi(\boldsymbol{\theta}) = O_p(m^{-1}), \quad (4)$$

$$Var_{\tilde{\pi}}(\boldsymbol{\theta}) - Var_{\pi}(\boldsymbol{\theta}) = o_p(n^{-1}), \quad (5)$$

$$d^2(\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X}), \delta_{\boldsymbol{\theta}^*}) - d^2(\pi(\boldsymbol{\theta}|\boldsymbol{X}), \delta_{\boldsymbol{\theta}^*}) = o_p(n^{-1}), \quad (6)$$

where E_{π} and $E_{\tilde{\pi}}$ denote the expectations with respect to $\pi(\boldsymbol{\theta}|\boldsymbol{X})$ and $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X})$, respectively; Var_{π} and $Var_{\tilde{\pi}}$ denote the variances with respect to $\pi(\boldsymbol{\theta}|\boldsymbol{X})$ and $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X})$, respectively; and $d^2(\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X}), \delta_{\boldsymbol{\theta}^*}) = \int_{\Theta} \|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_2^2 \tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X}) d\boldsymbol{\theta}$ is the square of the Wasserstein distance of order 2 between $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X})$ and the Dirac measure at $\boldsymbol{\theta}^*$.

Equations (4) and (5) measure the accuracy of the approximation of $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X})$ to $\pi(\boldsymbol{\theta}|\boldsymbol{X})$ in terms of mean and variance, respectively. Usually, $d^2(\pi(\boldsymbol{\theta}|\boldsymbol{X}), \delta_{\boldsymbol{\theta}^*})$ is $O(n^{-1})$, thus equation (6) implies that $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X})$ and $\pi(\boldsymbol{\theta}|\boldsymbol{X})$ share the same convergence rate toward the true value $\boldsymbol{\theta}^*$. In other words, the subposterior aggregation does not lose much information about the data.

Rather than $\boldsymbol{\theta}$ itself, sometimes we are interested in $h(\boldsymbol{\theta})$, a $\mathbb{R}^p \mapsto \mathbb{R}^q$ function of $\boldsymbol{\theta}$. A similar result, which measures the accuracy of the approximation $\tilde{\pi}(h(\boldsymbol{\theta})|\boldsymbol{X})$, can be obtained if we further have the following assumption:

A4 $h(\boldsymbol{\theta})$ is four times continuously differentiable on Θ and satisfies $\int_{\Theta} g(\boldsymbol{\theta}) \|h(\boldsymbol{\theta})\|_2^2 \exp\{-nu_n(\boldsymbol{\theta})\} d\boldsymbol{\theta} < \infty$ for $n = 1, 2, \ldots$

Corollary 1

If the conditions A1–A4 are satisfied, then we have

$$E_{\tilde{\pi}} h(\boldsymbol{\theta}) - E_{\pi} h(\boldsymbol{\theta}) = O_p(m^{-1}),$$

$$Var_{\tilde{\pi}} h(\boldsymbol{\theta}) - Var_{\pi} h(\boldsymbol{\theta}) = o_p(n^{-1}),$$

$$d^2(\tilde{\pi}(h(\boldsymbol{\theta})|\boldsymbol{X}), \delta_{h(\boldsymbol{\theta}^*)}) - d^2(\pi(h(\boldsymbol{\theta})|\boldsymbol{X}), \delta_{h(\boldsymbol{\theta}^*)}) = o_p(n^{-1}).$$

The proof is similar to that of Theorem 1, which is based on the expansion for the posterior mean of $h(\boldsymbol{\theta})$ and thus omitted in the paper.

3 Double Parallel Monte Carlo

In this section, we first give a brief review of the Pop-SAMC algorithm and discuss its implementation on the OpenMP platform. Then we describe the Double Parallel Monte Carlo scheme.

3.1 Pop-SAMC algorithm and its OpenMP implementation

As aforementioned, although MCMC is powerful for analyzing the data of complex structures, its computer-intensive nature precludes its use for big data analysis. To accelerate computation, one feasible way is to conduct parallel MCMC simulations. People have debated for a long time to make a single long run or many short runs. For conventional MCMC algorithms, such as the Metropolis-Hastings algorithm (Metropolis *et al.*, 1953; Hasting, 1970) and the Gibbs sampler (Geman and Geman, 1984), parallel runs may not provide any theoretical advantages over a single long run. In general, if one cannot get a good answer with a long run, then one cannot get a good answer with many short runs either. However, this situation differs for the population stochastic approximation Monte Carlo (pop-SAMC) algorithm (Song *et al.*, 2014), where it is shown that running pop-SAMC with κ chains (in parallel) for T iterations is asymptotically more efficient than running a single SAMC chain for κT iterations when the gain factor sequence decreases slower than $O(1/t)$, where t indexes iterations. This is due to that the chains in pop-SAMC interact with each other intrinsically.

The pop-SAMC algorithm consists of two steps, population sampling and $\boldsymbol{\xi}$-updating, where $\boldsymbol{\xi}$ denotes a vector of adaptive parameters evolving with iterations. In the population sampling step, each chain is updated independently for one or a few iterations. In the $\boldsymbol{\xi}$-updating step, $\boldsymbol{\xi}_t$ (i.e., the value of $\boldsymbol{\xi}$ at iteration t) is updated based on the collected information from individual chains, which enforces interactions between different chains and, consequently, improves the efficiency of the algorithm. To make the paper self-contained, the algorithm is briefly described as follows.

Suppose that we are interested in simulating samples from a density function $p(\boldsymbol{\theta})$, $\boldsymbol{\theta} \in \Theta$, and Θ has been partitioned into M subregions: $E_1 = \{\boldsymbol{\theta}: U(\boldsymbol{\theta}) < u_1\}$, $E_2 = \{\boldsymbol{\theta}: u_1 \le U(\boldsymbol{\theta}) < u_2\}$, …, $E_{M-1} = \{\boldsymbol{\theta}: u_{M-2} \le U(\boldsymbol{\theta}) < u_{M-1}\}$, and $E_M = \{\boldsymbol{\theta}: U(\boldsymbol{\theta}) \ge u_{M-1}\}$, where $U(\boldsymbol{\theta})$ is a pre-specified function of $\boldsymbol{\theta}$, e.g., $U(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta})$, and $u_1 < u_2 < \cdots < u_{M-1}$ are pre-specified numbers. To explain the concept of SAMC, we assume for the time being that all the subregions are non-empty; that is, $z_i = \int_{E_i} p(\boldsymbol{\theta}) d\boldsymbol{\theta} > 0$ for all $i = 1, \ldots, M$. However, as explained in Liang *et al.* (2007), the algorithm does allow the existence of empty subregions. Let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_M)$ denote the desired sampling distribution of the M subregions, where $\sum_{i=1}^{M} \pi_i = 1$ and $\pi_i > 0$ for all $i = 1, \ldots, M$. Given the partition and the desired sampling distribution, Pop-SAMC seeks to draw samples from the distribution

$$p_{\boldsymbol{z}}(\boldsymbol{\theta}) \propto \sum_{i=1}^{M} \frac{\pi_i p(\boldsymbol{\theta})}{z_i} I(\boldsymbol{\theta} \in E_i).$$

If z_i's are known and the space is partitioned appropriately, e.g., the energy bandwidth of each subregion is small enough, then the sampling will lead to a random walk in the space of subregions and thus the local-trap problem can be overcome essentially. However, since z_1, …, z_M are generally unknown, Pop-SAMC employs the stochastic approximation algorithm (Robbins and Monro, 1951) to learn their values (up to a constant factor) in an adaptive way.

Let κ denote the population size, i.e., the number of parallel Markov chains contained in Pop-SAMC, and let $\boldsymbol{\theta}_t = (\boldsymbol{\theta}_{t1}, \ldots, \boldsymbol{\theta}_{t\kappa})$ denote the current state of the κ chains. Let $\boldsymbol{\xi}_t = (\xi_{t1}, \ldots, \xi_{tM})$ denote the working estimate of $(\log(z_1/\pi_1), \ldots, \log(z_M/\pi_M))$ obtained at iteration t. One iteration of the algorithm consists of the following steps:

1. (Population sampling) For $i = 1, \ldots, \kappa$, generate a new sample $\boldsymbol{\theta}_{t,i}$ starting from $\boldsymbol{\theta}_{t-1,i}$ by a single MH update with the target distribution given by

$$p_{\xi_{t-1}}(\boldsymbol{\theta}) \propto \sum_{j=1}^{M} \frac{p(\boldsymbol{\theta})}{e^{\xi_{t-1,j}}} I(\boldsymbol{\theta} \in E_j). \quad (7)$$

2. ($\boldsymbol{\xi}$-update) Set $\boldsymbol{\xi}_t = \boldsymbol{\xi}_{t-1} + \gamma_t(\boldsymbol{H}_t - (1/M)\mathbf{1})$, where $\boldsymbol{H}_t = (\sum_{i=1}^{\kappa} I(\boldsymbol{\theta}_{t,i} \in E_1)/\kappa, \cdots, \sum_{i=1}^{N} I(\boldsymbol{\theta}_{t,i} \in E_M)/\kappa)^T$, and γ_t is a gain factor.

To ensure the convergence of the algorithm, the gain factor $\{\gamma_t\}$ is required to satisfy the conditions:

$$\sum_{t=1}^{\infty} \gamma_t = \infty, \quad \frac{\gamma_{t+1} - \gamma_t}{\gamma_t} = O(\gamma_{t+1}^{\tau}), \quad \sum_{t=1}^{\infty} \frac{\gamma_t^{(1+\tau')/2}}{\sqrt{t}} < \infty,$$

for some $\tau \in [1, 2)$ and $\tau' \in (0, 1)$. For example, one can set $\gamma_t = O(1/t^{\zeta})$ for $\zeta \in (1/2, 1]$. To accommodate the case that $\boldsymbol{\xi}_t$ takes values in an unbounded space, a varying truncation version of the algorithm can be considered as in Andrieu *et al.* (2005).

Like the SAMC algorithm(Liang *et al.*, 2007), Pop-SAMC possesses the self-adjusting mechanism, which operates based on past samples and enables the simulation to be immune to local traps. This can be considered as a significant advantage over conventional MCMC algorithms, such as the Metropolis-Hastings algorithm and the Gibbs sampler. Also, we would like to state that the pop-SAMC algorithm is essentially a dynamic importance sampling algorithm for which the trial distribution, i.e., the working target distribution (7), changes from iteration to iteration, and the quantities of interest can be estimated through weighted averaging as in conventional importance sampling (Liang, 2009). That is, Pop-SAMC generates a sequence of importance samples $\{(\boldsymbol{\theta}_{t,1}, e^{\xi_{t,J(\boldsymbol{\theta}_{t,1})}}), \ldots, (\boldsymbol{\theta}_{t,\kappa}, e^{\xi_{t,J(\boldsymbol{\theta}_{t,\kappa})}})\}$, where $J(\boldsymbol{\theta}_{t,i})$ denotes the index of the subregion that $\boldsymbol{\theta}_{t,i}$ belongs to, and $e^{\xi_{t,J(\boldsymbol{\theta}_{t,i})}}$ Specifies the importance weight of $\boldsymbol{\theta}_{t,i}$.

OpenMP is an application programming interface (API) for parallel programming on multi-core CPUs which are now available in regular desktops/laptops. It works in a shared memory mode with the fork/join parallelism, and is particularly suitable for pop-SAMC. To

be precise, the population sampling step of pop-SAMC can be carried out in parallel through the pragma *omp parallel* to fork multiple threads with each thread running for an individual Markov chain. After the parallel execution, the threads join back to the master thread, where $\boldsymbol{\xi}_t$ is updated based on the information collected from the multiple threads. Since OpenMP works in a shared memory mode, distributing the updated $\boldsymbol{\xi}_t$ to different threads is avoided. Since the population sampling steps cost the major portion of the CPU, the parallel execution provides a nearly linear speedup for the simulation.

3.2 Double Parallel Monte Carlo

Based on the subposterior aggregation theory studied in Section 2 and the Pop-SAMC algorithm, we propose the following Double Parallel Monte Carlo algorithm for Bayesian analysis of big data.

- (Data Parallel) Divide the dataset into k subsets with each being about of the same sample size.
- (Simulation Parallel) Run Pop-SAMC for each subposterior $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X}_{[i]})$ separately. Let $\{(\boldsymbol{\theta}_1^{(i)}, w_1^{(i)}), \ldots, (\boldsymbol{\theta}_N^{(i)}, w_N^{(i)})\}$ denote the importance samples generated by Pop-SAMC from $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X}_{[i]})$ for $i = 1, \ldots, k$. Let $\hat{\boldsymbol{\mu}}^{(i)} = \sum_{j=1}^{N} w_j^{(i)} \boldsymbol{\theta}_j^{(i)} / \sum_{j=1}^{N} w_j^{(i)}$ denote the mean of the subposterior $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X}_{[i]})$.
- (Sample aggregation) Calculate the global mean $\hat{\boldsymbol{\mu}} = \sum_{i=1}^{k} \hat{\boldsymbol{\mu}}^{(i)}/k$, recenter the importance samples as $\{(\boldsymbol{\theta}_1^{(i)} - \hat{\boldsymbol{\mu}}^{(i)} + \hat{\boldsymbol{\mu}}, w_1^{(i)}), \ldots, (\boldsymbol{\theta}_N^{(i)} - \hat{\boldsymbol{\mu}}^{(i)} + \hat{\boldsymbol{\mu}}, w_N^{(i)})\}$ for $i = 1, \ldots, k$.

Then, for each $i = 1, 2, \ldots, k$, the re-centered samples can be viewed as a batch of importance samples generated from the full data posterior. For any function $h(\boldsymbol{\theta})$ that satisfies $A4$, the expectation $\rho = E_{\pi} h(\boldsymbol{\theta})$ can be naturally estimated by $\hat{\rho}_1 = \sum_{i=1}^{k} \hat{\rho}_1^{(i)}/k$, where $\hat{\rho}_1^{(i)} = \sum_{j=1}^{N} w_j^{(i)} h(\boldsymbol{\theta}_j^{(i)} - \hat{\boldsymbol{\mu}}^{(i)} + \hat{\boldsymbol{\mu}}) / \sum_{j=1}^{N} w_j^{(i)}$. Alternatively, ρ can be estimated by

$$\hat{\rho}_2 = \frac{\sum_{i=1}^{k} \sum_{j=1}^{N} w_j^{(i)} h(\boldsymbol{\theta}_j^{(i)} - \hat{\boldsymbol{\mu}}^{(i)} + \hat{\boldsymbol{\mu}})}{\sum_{i=1}^{k} \sum_{j=1}^{N} w_j^{(i)}}.$$

Let $U_i = \sum_{j=1}^{N} w_j^{(i)} h(\boldsymbol{\theta}_j^{(i)} - \hat{\boldsymbol{\mu}}^{(i)} + \hat{\boldsymbol{\mu}}), S_i = \sum_{j=1}^{N} w_j^{(i)}$, $S = E(S_i)$ and $V_i = U_i - \rho S_i$. Following from the property of SAMC, the variances of U_i and V_i are both finite. Then the standard error of $\hat{\rho}_2$ can be calculated as for the ratio estimate (Ripley, 1987). The V_i's can be treated as iid random variables with zero mean and finite variance, and its variance can be estimated by $\hat{\sigma}_V^2 = 1/k \sum_{i=1}^{k} V_i^2$. The law of large numbers implies that $1/\sqrt{k} \sum_{i=1}^{k} V_i$ is asymptotically normal $N(0, \sigma_V^2)$ and that

$$\sqrt{k}(\hat{\rho}_2 - \rho) = \frac{\frac{1}{\sqrt{k}}\sum_{i=1}^{k} V_i}{\frac{1}{k}\sum_{i=1}^{k} S_i} \to N(0, \sigma^2),$$

where $\sigma^2 = \sigma_V^2/S^2$, and it can be estimated by $\hat{\sigma}_V^2/\hat{S}^2$ with $\hat{S} = \sum_{i=1}^{k} S_i/k$.

4 Simulation Study

4.1 Logistic Regression

The first example is very simple, whose goal is to show the validity of the proposed subposterior aggregation method. The example is adopted from Srivastava *et al.* (2015b). It is for a logistic regression with $n = 10^4$ and the true parameter $\boldsymbol{\theta}^* = (1, -1)^T$. The covariates Z_1 and Z_2 are drawn from the standard Gaussian distribution. The prior distribution of $\boldsymbol{\theta}$ is $N(0, I_2)$. To follow the notation in Section 2, we let $X = (Y, Z_1, Z_2)$.

To implement the proposed Double Parallel algorithm, we randomly divided the dataset into 10 subsets with each consisting of 1000 samples. Then Pop-SAMC was run for each subset. Specifically, for each subset, we partitioned the parameter space Θ according to the energy function $U(\boldsymbol{\theta}) = -\log p_j(\boldsymbol{\theta})$ with an equal bandwidth $\Delta u = 0.5$ into five subregions $E_1 = \{\boldsymbol{\theta}: U(\boldsymbol{\theta}) < u+0.5\}$, $E_2 = \{\boldsymbol{\theta}: u+0.5 \le U(\boldsymbol{\theta}) < u+1\}$, $E_3 = \{\boldsymbol{\theta}: u + 1 \le U(\boldsymbol{\theta}) < u + 1.5\}$ $E_4 = \{\boldsymbol{\theta}: u + 1.5 \le U(\boldsymbol{\theta}) < u + 2\}$, and $E_5 = \{\boldsymbol{\theta}: U(\boldsymbol{\theta}) \ge u + 2\}$, where p_j denote the subposterior of the jth subset, and u was chosen as the smallest value of $U(\boldsymbol{\theta})$ obtained in a preliminary trial. The gain factor γ_t was set as $100/\max(100, t)$. The proposal was set as a Gaussian random walk distribution with the covariance matrix $0.2^2 I_2$. The population size was set to $N = 10$ and the number of iterations was set to $T = 10^5$. The first 10^4 iterations were discarded for the burn-in process, and samples were collected from the remainder of the run at every 5 iterations. In total, we had 1.8×10^5 importance samples collected at the end of each run.

Figure 1 shows the contour plots of the full data posterior $\pi(\boldsymbol{\theta}|\boldsymbol{X})$, each subposterior $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X}_{[j]})$, and the proposed mixture posterior $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X})$. The R package KernSmooth (Wand, 2015) was used to generate the corresponding binned kernel density estimates. The plots indicate that each subposterior has a similar shape with the full data posterior, however, most of them have a notably biased center from the true parameter $\boldsymbol{\theta}^*$. By shifting the mean of each subposterior to the global mean, the bias was successfully removed. The mixture posterior $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X})$ closely matches the full data posterior $\pi(\boldsymbol{\theta}|\boldsymbol{X})$.

4.2 Nonlinear Regression

We use the following nonlinear regression example to compare the approximation accuracy of the proposed Double Parallel algorithm, posterior interval estimation(PIE) (Li *et al.*, 2017) and Consensus Monte Carlo (Scott *et al.*, 2016):

$$y_i = \theta_1\sqrt{\exp(\theta_2 z_{i1})} + \theta_3\sqrt{\exp(\theta_4 z_{i2} + \theta_5 z_{i3})} + \theta_6 z_{i4} + \theta_7 \sin(\theta_8 z_{i5} + \pi/6) + \theta_9 \frac{\exp(\theta_{10} z_{i6})}{1 + \exp(\theta_{10} z_{i6})} + \varepsilon_i,$$

for $i = 1, 2, \ldots, n$, where $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8, \theta_9, \theta_{10}) = (2, 2, 2, 2, -2, -2, 5, 2, 4, 2)$ are the true parameters, and $\varepsilon_1, \cdots, \varepsilon_n$ are i.i.d. normal random errors with mean 0 and variance $\sigma^2 = 0.25$. The covariates z_1 and z_2 are drawn from the standard normal distribution independently. Set the covariates $z_3 = 0.7z_2 + 0.3e$ and $z_4 = 0.7z_1 + 0.3e'$, where e and e' are also drawn from the standard normal distribution. Under this setting, z_2 and z_3 are highly correlated with a correlation coefficient of 0.919, so do z_1 and z_4. The covariates z_5 and z_6 are drawn independently from a t-distribution with the degree of freedom 10. We generated $n = 10^4$ samples from this model and used the non-informative prior $g(\boldsymbol{\theta}) \propto 1$ for all the parameters. To follow the notation in Section 2, we set $X_i = (y_i, z_{i1}, z_{i2}, z_{i3}, z_{i4}, z_{i5}, z_{i6})$.

For the Double Parallel algorithm, we randomly divided the dataset into 10 subsets with each consisting of 1000 samples. Pop-SAMC was run for each subset separately with the same setting as for the previous example except that the energy bandwidth was set to $\Delta u = 2$ and the covariance matrix of the Gaussian random walk proposal distribution was set to $0.01^2 I_{10}$. For comparison, Consensus Monte Carlo and PIE were also applied to this example. Note for Consensus Monte Carlo, the subset posterior is defined as

$$\frac{\prod_{i=1}^{m} f(X_{ji}|\boldsymbol{\theta}) g^{1/k}(\boldsymbol{\theta})}{\int_{\Theta} \prod_{i=1}^{m} f(X_{ji}|\boldsymbol{\theta}) g^{1/k}(\boldsymbol{\theta}) d\boldsymbol{\theta}}, \quad (8)$$

which is slightly different from the subposterior defined in (2), the one used in PIE and Double Parallel.

To evaluate the approximation accuracy of Double Parallel, PIE and Consensus Monte Carlo, we utilized two criteria. The first criterion is the credible interval. The similarity between the credible intervals constructed by a method and those constructed by the full data-based MCMC method indicates the approximation accuracy of the method. We measured the similarity by the p-value of the paired-t test for the length of credible intervals of different parameters. The second criterion is more direct, which uses the following function

$$\text{Accuracy}(\pi_1, \pi_2) = 1 - \frac{1}{2} \int |\pi_1(\theta) - \pi_2(\theta)| d\theta, \quad (9)$$

to measure the similarity between two density functions $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$. This function lies in [0, 1], with a larger value indicating higher similarity. In our case, $\boldsymbol{\pi}_1$ is the posterior density obtained by the method under evaluation, and $\boldsymbol{\pi}_2$ is the posterior density generated based on the full data. These densities can be estimated based on the posterior samples using the bkde function in the R package KernSmooth (Wand, 2015).

Table 1 and Table 2 show, respectively, the 90% and 80% credible intervals produced by different methods. As indicated by the p-values, Double Parallel and PIE produced very similar credible intervals to the full data-based MCMC method, while Consensus Monte

Carlo did not. Hence, in this criterion, we conclude that Double Parallel and PIE outperforms Consensus Monte Carlo for this example.

Table 3 shows the approximation accuracy of Double Parallel and Consensus Monte Carlo to the full data posterior distribution, for each of the ten parameters. We don't include PIE here because (i) from previous results, we expect the approximation accuracy of PIE should be very similar to that of Double Parallel, and (ii) PIE is proposed for quantile approximation and cannot be directly used for density approximation. Table 3 shows that for some parameters, Consensus Monte Carlo has better performance, while for others, Double Parallel wins. We suspect that this may depend on the similarity of each parameter's posterior distribution with the normal distribution. In the average of approximation accuracy of the ten parameters, we find that Double Parallel is still slightly better than Consensus Monte Carlo for this example: Double Parallel produced an average of 0.924 with a standard deviation of 0.012, while Consensus Monte Carlo produced an average of 0.902 with a standard deviation of 0.015.

5 A Big Data Example

The goal of this example is to show how efficient the Double Parallel algorithm can be compared to the traditional single chain MCMC algorithm for a big data problem. For this purpose, we applied the Double Parallel algorithm to the MiniBooNE particle identification dataset, which is available at the UCI machine learning repository. This dataset records 130,064 events (observations), including 36,499 signal events and 93,565 background events. Each observation consists of the event type (signal event or background event) and 50 associated particle variables. The task of the problem is to explore the relationship between the event type and the associated particle variables. The more detailed description for the dataset and its physical background can be found in Roe *et al.* (2005).

The problem can be naturally treated using a logistic regression, where the event type is used as the response variable and the 50 associated particle variables are used as the predictors. To identify the important variables that are associated with the event type, we let the regression coefficients be subject to a heavy-tail distribution, $t(3)$, which belongs to the class of local shrinkage priors but is more moderate in shrinking large regression coefficients than the Lasso prior(Tibshirani, 1996). Other local shrinkage priors, such as the horseshoe prior(Carvalho *et al.*, 2010), can also be used here without affecting on the efficiency of the proposed algorithm. In data preprocessing, we first removed 468 samples with missing observations and then randomly divide the remaining samples into 10 subsets of nearly the same sample size. For each subset, Pop-SAMC was run with the population size $\kappa = 20$. The sample space was partitioned with an energy bandwidth $\Delta u = 1$ and the subregions determined through a preliminary run. The gain factor was set as $\gamma_t = \min(1, (t/1000)^{-0.6})$. The algorithm was run for 1.1×10^5 iterations, where the first 10^4 iterations were for the burn-in process and the samples collected from the remaining 10^5 iterations were used for inference. At each iteration, 20 parameters was randomly selected to be updated along a random direction with a step size of 0.1. The acceptance rate was around 0.16, which indicates the effectiveness of the simulation. On a high-end Dell Precision T7610

Workstation with 24 cores, one run of Pop-SAMC costs about 9 minutes (wall clock time) or 166 minutes total CPU time.

For comparison, the single chain Metropolis-Hastings (MH) algorithm was directly applied to simulate from the full data posterior. The algorithm was run for 2.2×10^6 ($= 20\times1.1\times10^5$) iterations correspondingly, where the first 2.0×10^5 were discarded for the burn-in process and the samples collected from the remaining iterations were used for inference. The MH algorithm used the same proposal as the Pop-SAMC and the resulting acceptance rate was also about 0.16. On the same computer, one run of the MH algorithm costs about 1,373 minutes (wall clock time) or 1,371 minutes CPU time. In wall clock time, the computational cost by the Double Parallel algorithm is only 0.67% of that by the single chain MH algorithm! We have also implemented Consensus Monte Carlo on this dataset. For each subset, we utilized MH algorithm to generate samples and for each subset, the MH algorithm was run for 2.2×10^6 iterations. To be fair to Consensus Monte Carlo, we only reported its results, but not its computational time, since Consensus Monte Carlo can also be implemented by Pop-SAMC, and in this case, it should use similar time with our method.

Table 4 shows the computational results produced by the three methods. For each method, we reported only the ten most significant variables, including their posterior mean and standard deviation. Here the significance of each variable was measured according to the ratio of its posterior mean and standard deviation. The results from the Double Parallel algorithm is surprisingly consistent with the result from the MH algorithm: All the variables have about the same posterior mean and standard deviation. The top 10 significant variables are exactly the same, even in the same order! This again indicates the validity of the Double Parallel algorithm for approximation of the full data posterior. Consensus Monte Carlo also selected the same top 10 significant variables, but in a slightly different order.

6 Discussion

This paper has proposed a simple, practical and efficient MCMC algorithm for Bayesian analysis of big data. The proposed algorithm has two innovations. First, it provides a simple and practical way to aggregate subposteriors to approximate the full data posterior. Second, it suggests to implement the Pop-SAMC algorithm to simulate from each subposterior. Since the whole algorithm consists of two levels of parallel, data parallel and simulation parallel, it is called the Double Parallel Monte Carlo algorithm. Theoretically, we have shown that the Double Parallel algorithm can produce a good approximation to the full data posterior distribution. Empirically, we have demonstrated that the results produced by the Double Parallel algorithm agree well with those generated from the full data posterior, while enabling massive speed-ups in computational time.

Compared to Consensus Monte Carlo, the Double Parallel algorithm has two advantages. First, it is more efficient in terms of quantities of effective samples generated in single unit CPU time. Recall that the Double Parallel algorithm converts the subset posterior samples to full-data posterior samples by simply recentering, while a weighted average applies for Consensus Monte Carlo. Second, it relaxes the requirement for the asymptotic normality of the posterior distributions by assuming the Laplace regularity for the log-likelihood function.

This relaxation may explain why the Double Parallel algorithm outperforms Consensus Monte Carlo, as shown in Tables 1, 2 and 4, for our examples.

The Double Parallel algorithm works based on Laplace's method, but it can also cover some problems that are traditionally treated as discrete, such as variable selection problems. As shown in Section 5, these problems can be treated as continuous by imposing a local shrinkage prior on the space of variable coefficients. A further extension of the proposed algorithm to general discrete parameter space will be of great interest.

Acknowledgments

Liang's research was supported in part by the grants DMS-1545202, DMS-1612924 and R01-GM117597. The authors thank the Editor, Associate Editor and two referees for their constructive comments which has led to significant improvement of this paper.

Appendix A

Proof of Theorem 1

Under conditions $A1$ and $A2$, we can expand the mean of each subposterior at the corresponding MLE, $\hat{\boldsymbol{\theta}}^{(j)}$, as follows:

$$\boldsymbol{\mu}^{(j)} = E_{\tilde{\pi}_j}(\boldsymbol{\theta}) = \hat{\boldsymbol{\theta}}^{(j)} + \frac{\hat{I}^{(j)-1}}{n}\left[\frac{\partial \log g(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big|_{\hat{\boldsymbol{\theta}}^{(j)}} - \frac{1}{2}\hat{H}^{(j)}\hat{I}^{(j)-1}\right] + O(n^{-2}),$$

where $\tilde{\pi}_j = \tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X}_{[j]})$, $\hat{I}^{(j)} = -\frac{1}{m}\frac{\partial^2 \log f(\boldsymbol{X}_{[j]}|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}^T}\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}^{(j)}}$, $\hat{H}^{(j)} = -\frac{1}{m}\frac{\partial^3 log\, f(\boldsymbol{X}_{[j]}|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}^T \partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}^{(j)}}$, and $\hat{H}^{(j)}\hat{I}^{(j)-1}$ is a vector whose rth element equals $\sum_{st}\hat{H}^{(j)}_{rst}\hat{I}^{(j)-1}_{st}$. To simplify the notation, we denote $\hat{I}^{(j)-1}[\partial \log g(\boldsymbol{\theta})/\partial \boldsymbol{\theta}|_{\hat{\boldsymbol{\theta}}^{(j)}} - \frac{1}{2}\hat{H}^{(j)}\hat{I}^{(j)-1}]$ by $\hat{\boldsymbol{\nu}}^{(j)}$. Here we would like to mention that although A1 and A2 are directly associated with the Laplace approximation of the full posterior, i they can also lead to the conditions that guarantee the Laplace approximation of each subposterior, as long as m goes to ∞ with n.

Moreover, for each $\hat{\boldsymbol{\theta}}^{(j)}$, we have

$$\hat{\boldsymbol{\theta}}^{(j)} = \boldsymbol{\theta}^* + \frac{\boldsymbol{\xi}^{(j)}}{\sqrt{m}} + O_p(m^{-1}),$$

where

$$\boldsymbol{\xi}^{(j)} = \frac{1}{\sqrt{m}} I^{-1}\sum_{i=1}^{m}\frac{\partial \log f(X_{ji}|\boldsymbol{\theta}^*)}{\partial \boldsymbol{\theta}}, \quad I = -E_{\boldsymbol{X}|\boldsymbol{\theta}^*}\frac{\partial^2 \log f(\boldsymbol{X}|\boldsymbol{\theta}^*)}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}^T}.$$

Therefore, the mean of the mixture distribution $\tilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X})$ is given by

$$E_{\tilde{\pi}}(\boldsymbol{\theta}) = \frac{1}{k}\sum_{j=1}^{k}\boldsymbol{\mu}^{(j)} = \boldsymbol{\theta}^{*} + \frac{1}{k}\sum_{j=1}^{k}\frac{\boldsymbol{\nu}^{(j)}}{n} + \frac{1}{k}\sum_{j=1}^{k}\frac{\boldsymbol{\xi}^{(j)}}{\sqrt{m}} + O_p(m^{-1}) + O(n^{-2}).$$

We can also implement a similar procedure on the full data posterior $\boldsymbol{\pi}(\boldsymbol{\theta}|\boldsymbol{X})$ and obtain

$$E_{\pi}(\boldsymbol{\theta}) = \boldsymbol{\theta}^{*} + \frac{\boldsymbol{\nu}}{n} + \frac{\boldsymbol{\xi}}{\sqrt{n}} + O_p(n^{-1}) + O(n^{-2}),$$

where

$$\boldsymbol{\nu} = \hat{I}^{-1}[\partial \log g(\boldsymbol{\theta})/\partial\boldsymbol{\theta}|_{\hat{\boldsymbol{\theta}}} - \frac{1}{2}\hat{H}\hat{I}^{-1}], \boldsymbol{\xi} = \frac{1}{\sqrt{n}}I^{-1}\sum_{i=1}^{n}\frac{\partial \log f(X_i|\boldsymbol{\theta}^{*})}{\partial\boldsymbol{\theta}}, \hat{I} = -\frac{1}{n}\frac{\partial^2 \log f(\boldsymbol{X}|\boldsymbol{\theta})}{\partial\boldsymbol{\theta}\,\partial\boldsymbol{\theta}^{T}}\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}, \hat{H},$$

$$= -\frac{1}{m}\frac{\partial^3 log\, f(\boldsymbol{X}|\boldsymbol{\theta})}{\partial\boldsymbol{\theta}\,\partial\boldsymbol{\theta}^{T}\,\partial\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$$

and $\hat{\boldsymbol{\theta}}$ denotes the MLE of θ calculated with the full dataset. By noting that $\frac{\boldsymbol{\xi}}{\sqrt{n}} = \frac{1}{k}\sum_{j=1}^{k}\frac{\boldsymbol{\xi}^{(j)}}{\sqrt{m}}$ and $\boldsymbol{\nu}$ are $O(1)$, $m < n$, equation (4) in Theorem 1 can thereby be verified:

$$E_{\tilde{\pi}}(\boldsymbol{\theta}) - E_{\pi}(\boldsymbol{\theta}) = \frac{1}{k}\sum_{j=1}^{k}\frac{\boldsymbol{\nu}^{(j)}}{n} + O_p(m^{-1}) - \frac{\boldsymbol{\nu}}{n} + O_p(n^{-1}) + O(n^{-2})$$
$$= O(n^{-1}) + O_p(m^{-1}) + O(n^{-1}) + O_p(n^{-1}) + O(n^{-2}) = O_p(m^{-1}).$$

The variance of each subposterior can be approximated as follows:

$$Var_{\tilde{\pi}_j}(\boldsymbol{\theta}) = \frac{\hat{I}^{(j)-1}}{n} + O(n^{-2}).$$

Therefore, the variance of the mixture distribution $\tilde{\boldsymbol{\pi}}(\boldsymbol{\theta}|\boldsymbol{X})$ is given by

$$Var_{\tilde{\pi}}(\boldsymbol{\theta}) = \frac{1}{k}\sum_{j=1}^{k}Var_{\tilde{\pi}_j}(\boldsymbol{\theta}) = \frac{1}{k}\sum_{j=1}^{k}\frac{\hat{I}^{(j)-1}}{n} + O(n^{-2}).$$

In addition, we have

$$Var_{\pi}(\boldsymbol{\theta}) = \frac{\hat{I}^{-1}}{n} + O(n^{-2}).$$

Since $\hat{I}^{(j)} = I + o_p(1)$ and $\hat{I} = I + o_p(1)$, we further have $\hat{I}^{(j)-1} = I^{-1} + o_p(1)$ and $\hat{I}^{-1} = I^{-1} + o_p(1)$. The equation (5) in Theorem 1 can thereby be verified:

$$Var_{\widetilde{\pi}}(\boldsymbol{\theta}) - Var_{\pi}(\boldsymbol{\theta}) = \frac{1}{k}\sum_{j=1}^{k}\frac{\hat{I}^{(j)-1}}{n} - \frac{\hat{I}^{-1}}{n} + O(n^{-2}) = \frac{I^{-1}}{n} + o_p(n^{-1}) - \frac{I^{-1}}{n} + o_p(n^{-1}) + O(n^{-2}) = o_p(n^{-1}).$$

Finally, for the square of the Wasserstein distance of order 2, we have

$$\begin{aligned}
&\left|d^2(\widetilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X}), \delta_{\boldsymbol{\theta}^*}) - d^2(\pi(\boldsymbol{\theta}|\boldsymbol{X}), \delta_{\boldsymbol{\theta}^*})\right| = \left|\int_{\Theta}\left\|\boldsymbol{\theta}-\boldsymbol{\theta}^*\right\|_2^2\widetilde{\pi}(\boldsymbol{\theta}|\boldsymbol{X})d\boldsymbol{\theta} - \int_{\Theta}\left\|\boldsymbol{\theta}-\boldsymbol{\theta}^*\right\|_2^2\pi(\boldsymbol{\theta}|\boldsymbol{X})d\boldsymbol{\theta}\right| \\
&= \left|\left\|E_{\widetilde{\pi}}(\boldsymbol{\theta}) - \boldsymbol{\theta}^*\right\|_2^2 + tr\left(Var_{\widetilde{\pi}}(\boldsymbol{\theta})\right) - \left\|E_{\pi}(\boldsymbol{\theta}) - \boldsymbol{\theta}^*\right\|_2^2 - tr\left(Var_{\pi}(\boldsymbol{\theta})\right)\right| \\
&\leq \left|\left\|E_{\widetilde{\pi}}(\boldsymbol{\theta}) - E_{\pi}(\boldsymbol{\theta}) + E_{\pi}(\boldsymbol{\theta}) - \boldsymbol{\theta}^*\right\|_2^2 - \left\|E_{\pi}(\boldsymbol{\theta}) - \boldsymbol{\theta}^*\right\|_2^2\right| + \left|tr\left(Var_{\widetilde{\pi}}(\boldsymbol{\theta}) - Var_{\pi}(\boldsymbol{\theta})\right)\right| \\
&\leq \left\|E_{\widetilde{\pi}}(\boldsymbol{\theta}) - E_{\pi}(\boldsymbol{\theta})\right\|_2^2 + 2\left\|E_{\widetilde{\pi}}(\boldsymbol{\theta}) - E_{\pi}(\boldsymbol{\theta})\right\|_2\left\|E_{\pi}(\boldsymbol{\theta}) - \boldsymbol{\theta}^*\right\|_2 + o_p(n^{-1}) \\
&= O_p(m^{-2}) + 2O_p(m^{-1})O_p(n^{-1/2}) + o_p(n^{-1}) = o_p(n^{-1}),
\end{aligned}$$

where the last equality follows from the fact $n = o(m^2)$. This verifies the equation (6) in Theorem 1.

References

Andrieu C, Moulines E, Priouret P. 2005; Stability of Stochastic Approximation under Verifiable Conditions. SIAM Journal of Control and Optimization. 44:283–312. DOI: 10.1137/S0363012902417267

Carvalho CM, Polson NG, Scott JG. 2010; The horseshoe estimator for sparse signals. Biometrika. 97:465–480.

Geman S, Geman D. 1984; Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. IEEE Trans. Pattern Anal. Mach. Intell. 6:721–741. DOI: 10.1109/TPAMI.1984.4767596 [PubMed: 22499653]

Hasting W. 1970; Monte Carlo sampling methods using Markov chains and their applications. Biometrika. 57:97–109. DOI: 10.1093/biomet/57.1.97

Kass, RE, Tierney, L, Kadane, JB. The validity of posterior expansions based on Laplace's method. In: Geisser, S, Hodges, JS, Press, SJ, ZeUner, A, editors. Bayesian and likelihood methods in statistics and econometrics: essays in honor of George A. Barnard. Vol. 7. 1990. 473–488.

Li C, Srivastava S, Dunson DB. 2017Simple, Scalable and Accurate Posterior Interval Estimation. Biometrika.

Liang F. 2009; On the use of Stochastic Approximation Monte Carlo for Monte Carlo integration. Statistics & Probability Letters. 79:581–587.

Liang F, Liu C, Carroll RJ. 2007; Stochastic Approximation in Monte Carlo Computation. Journal of the American Statistical Association. 102:305–320.

Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. 1953; Equation of State Calculations by Fast Computing Machines. The Journal of Chemical Physics. 21

Minsker S, Srivastava S, Lin L, Dunson DB. 2014Robust and Scalable Bayes via a Median of Subset Posterior Measures. ArXiv e-prints.

Neiswanger, W, Wang, C, Xing, EP. Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, UAI'14. AUAI Press; Arlington, Virginia, United States: 2014. Asymptotically Exact, Embarrassingly Parallel MCMC; 623–632. URL http://dl.acm.org/citation.cfm?id=3020751.3020816

Ripley, BD. Stochastic Simulation. John Wiley & Sons, Inc.; New York, NY, USA: 1987.

Robbins H, Monro S. 1951; A Stochastic Approximation Method. Ann. Math. Statist. 22:400–407. DOI: 10.1214/aoms/1177729586

Roe BP, Yang H-J, Zhu J, Liu Y, Stancu I, McGregor G. 2005; Boosted decision trees as an alternative to Artificial Neural Networks for particle identification. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. 543:577–584.

Scott SL, Blocker AW, Bonassi FV. 2016Bayes and Big Data: The Consensus Monte Carlo Algorithm. International Journal of Management Science and Engineering Management.

Song Q, Wu M, Liang F. 2014; Weak convergence rates of population versus single-chain Stochastic Approximation MCMC algorithms. Adv. in Appl. Probab. 46:1059–1083. DOI: 10.1239/aap/1418396243

Srivastava, S; Cevher, V; Tran-Dinh, Q; Dunson, DB. WASP: Scalable Bayes via barycenters of subset posteriors. AISTATS, volume 38 of JMLR: Workshop and Conference Proceedings. 2015a. (JMLR.org)

Srivastava S, Li C, Dunson DB. 2015bScalable Bayes via Barycenter in Wasserstein Space. ArXiv e-prints.

Tibshirani R. 1996; Regression Shrinkage and Selection via the Lasso. Journal of the Royal Statistical Society (Series B). 58:267–288.

Wand M. 2015KernSmooth: Functions for Kernel Smoothing Supporting Wand & Jones (1995). r package version 2.23–15.

Wang X, Dunson D. 2013Parallelizing MCMC via Weierstrass Sampler. ArXiv e-prints.

Wang, X, Guo, F, Heller, KA, Dunson, DB. Parallelizing MCMC with Random Partition Trees. In: Cortes, C, Lawrence, ND, Lee, DD, Sugiyama, M, Garnett, R, editors. Advances in Neural Information Processing Systems 28. Curran Associates, Inc.; 2015. 451–459. URL http://papers.nips.cc/paper/5986-parallelizing-mcmc-with-random-partition-trees.pdf

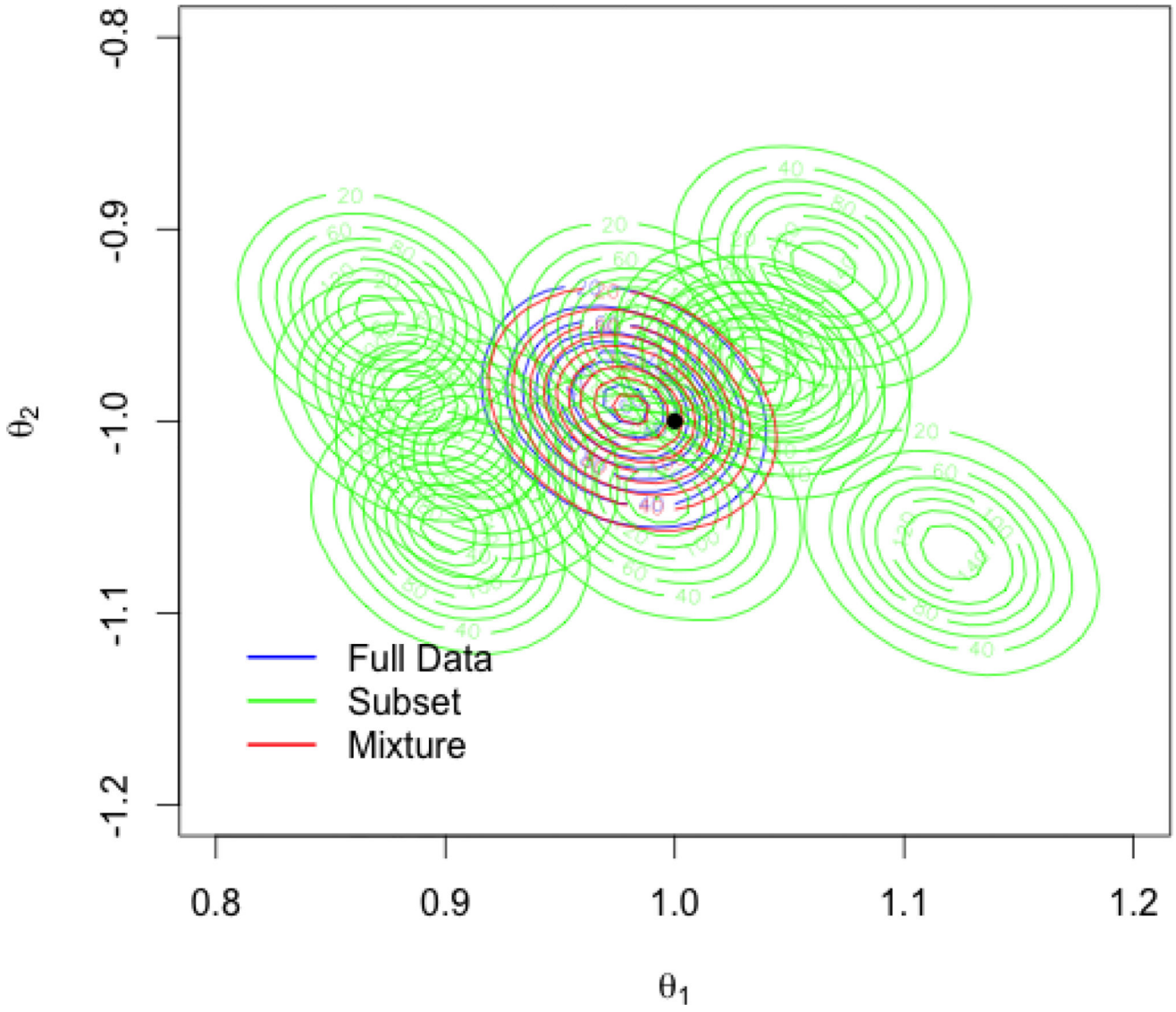

Figure 1.
Binned kernel posterior density estimates for the parameters of a logistic regression. The true parameter $\boldsymbol{\theta}^* = (1,-1)^T$ (black dot).

Table 1

90% credible intervals for each parameter produced by full data MCMC, Double Parallel, PIE and Consensus Monte Carlo, where the *p*-value is calculated by a paired-*t* test for the length of credible intervals in comparison with the full data MCMC method.

	MCMC	Double Parallel	PIE	Consensus Monte Carlo
θ_1	(1.965,2.016)	(1.966,2.014)	(1.966,2.014)	(1.958,2.018)
θ_2	(1.995,2.012)	(1.996,2.012)	(1.995,2.012)	(1.995,2.014)
θ_3	(1.968,2.040)	(1.967,2.038)	(1.967,2.038)	(1.970,2.055)
θ_4	(1.962,2.024)	(1.964,2.024)	(1.964,2.025)	(1.947,2.021)
θ_5	(−2.017,−1.950)	(−2.017,−1.951)	(−2.017,−1.950)	(−2.014,−1.932)
θ_6	(−2.024,−1.979)	(−2.022,−1.978)	(−2.022,−1.978)	(−2.023,−1.971)
θ_7	(4.984,5.007)	(4.984,5.007)	(4.984,5.007)	(4.983,5.007)
θ_8	(1.997,2.001)	(1.997,2.001)	(1.997,2.001)	(1.997,2.001)
θ_9	(3.972,4.060)	(3.972,4.066)	(3.972,4.066)	(3.956,4.051)
θ_{10}	(1.944,2.033)	(1.944,2.039)	(1.944,2.038)	(1.954,2.046)
p-value	—	0.769	0.586	2.68×10^{-3}

Table 2

80% credible intervals for each parameter produced by full data-based MCMC, Double Parallel, PIE and Consensus Monte Carlo, where the p-value is calculated by a paired-t test for the length of credible intervals in comparison with the full data MCMC method.

	MCMC	Double Parallel	PIE	Consensus Monte Carlo
θ_1	(1.972,2.012)	(1.971,2.008)	(1.971,2.008)	(1.968,2.011)
θ_2	(1.996,2.009)	(1.998,2.011)	(1.998,2.011)	(1.997,2.011)
θ_3	(1.975,2.031)	(1.976,2.030)	(1.976,2.030)	(1.978,2.044)
θ_4	(1.969,2.018)	(1.970,2.017)	(1.971,2.017)	(1.958,2.015)
θ_5	(−2.011,−1.957)	(−2.010,−1.958)	(−2.010,−1.958)	(−2.006,−1.945)
θ_6	(−2.019,−1.984)	(−2.017,−1.982)	(−2.017,−1.982)	(−2.018,−1.979)
θ_7	(4.986,5.004)	(4.987,5.005)	(4.987,5.005)	(4.986,5.004)
θ_8	(1.997,2.001)	(1.997,2.001)	(1.997,2.001)	(1.997,2.001)
θ_9	(3.981,4.050)	(3.982,4.056)	(3.982,4.056)	(3.969,4.041)
θ_{10}	(1.953,2.023)	(1.954,2.027)	(1.954,2.027)	(1.963,2.032)
p-value	—	0.901	0.811	0.016

Table 3

Accuracy of approximation to the full data-based posterior, measured in (9), by Double Parallel (DP) and Consensus Monte Carlo (CMC), for each parameter;

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}
Double Parallel	0.875	0.867	0.899	0.915	0.929	0.901	0.965	0.972	0.946	0.966
Consensus Monte Carlo	0.903	0.913	0.854	0.850	0.853	0.915	0.978	0.973	0.868	0.912

Table 4

Comparison of computational time (wall clock time) and parameter estimation for the MiniBooNE particle data set

MH Algorithm		Double Parallel(k = 10, κ = 20)		Consensus Monte Carlo	
wall clock time					
1,373 minutes		**9.2 minutes**		**–**	
Top 10 significant variables					
var13	−1.2140 (0.0158)	var13	−1.2273 (0.0153)	var13	−1.2349(0.0156)
var1	−1.3529 (0.0328)	var1	−1.3667 (0.0338)	var1	−1.3804(0.0325)
var16	3.3730 (0.0840)	var16	3.4174 (0.0861)	var16	3.3761(0.0818)
var4	−0.8788 (0.0254)	var4	−0.8825 (0.0234)	var32	1.0786(0.0290)
var32	1.0590 (0.0309)	var32	1.0733 (0.0329)	var4	−0.8960(0.0247)
var17	−2.1169 (0.0672)	var17	−2.1592(0.0709)	var17	−2.1685(0.0618)
var6	0.3825 (0.0139)	var6	0.3862 (0.0134)	var12	−0.7668(0.0281)
var12	−0.7503 (0.0278)	var12	−0.7624 (0.0281)	var6	0.3887(0.0145)
var34	−0.9239 (0.0355)	var34	−0.9335 (0.0344)	var34	−0.9568(0.0367)
var25	0.4004 (0.0206)	var25	0.4082 (0.0206)	var25	0.4065(0.0214)