Comparing Consensus Monte Carlo Strategies for Distributed Bayesian Computation

Steven L. Scott

December 8, 2016

Abstract

Consensus Monte Carlo is an algorithm for conducting Monte Carlo based Bayesian inference on large data sets distributed across many worker machines in a data center. The algorithm operates by running a separate Monte Carlo algorithm on each worker machine, which only sees a portion of the full data set. The worker-level posterior samples are then combined to form a Monte Carlo approximation to the full posterior distribution based on the complete data set. We compare several methods of carrying out the combination, including a new method based on approximating worker-level simulations using a mixture of multivariate Gaussian distributions. We find that resampling and kernel density based methods break down after 10 or sometimes fewer dimensions, while the new mixture-based approach works well, but the necessary mixture models take too long to fit.

1 Introduction

This article compares several implementations of consensus Monte Carlo methods for performing Monte Carlo based Bayesian inference in big data problems. By "big data" I mean a data set too large to be processed by a single machine. This is the definition used by computer engineers, who are largely responsible for creating the big data phenomenon by introducing widely used tools for managing massive data sets on distributed clusters (e.g. Dean and Ghemawat, 2008; Chang *et al.*, 2008). In principle these same tools could be used to implement scalable Bayesian inference on massive data sets stored in data centers.

Data centers are extremely large, shared, clusters of computers which can contain many tens of thousands of machines. Compute time in a data center is very cheap. Low end machines can be rented for as little as one or two cents per hour, which makes it inexpensive to pool hundreds or even thousands of machines to solve challenging computational problems. The data center computing model offers effectively infinite processing power, memory, and disk. The challenge is that communicating between machines in a data center is expensive. Scott *et al.* (2016) presented an example where repeated broadcast communications on a 500 machine cluster took a median of roughly .25 seconds per broadcast. This degree of latency makes parallel versions of traditional Bayesian computations untenable in a data center. For example, consider a data augmentation algorithm that alternates between worker machines imputing latent variables and a central node simulating parameters given complete data. With .25 second communication latency, such a scheme could produce at most 2 draws per second, even if none of the nodes actually did any computing.

If communication is expensive, a natural strategy is to avoid communicating altogether by partitioning the data among workers, running a full posterior simulation on each worker, and then combining the simulations as workers finish their simulation runs. Scott *et al.* (2016) named this approach "consensus Monte Carlo" (CMC). Motivating CMC is the fact that posterior distributions tend to be mathematically separable. Let $\mathbf{y}$ denotes the full data set, partitioned into S "shards" $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_S)$. Assuming conditional independence given θ across shards, the posterior distribution can be written as a product of independent distributions

$$\begin{aligned} p(\theta|\mathbf{y}) \propto p(\theta) \prod_{s=1}^{S} p(\mathbf{y}_s|\theta) &= \prod_{s} p(\mathbf{y}_s|\theta) p(\theta)^{1/S} \\ &\propto \prod_{s} p(\theta|\mathbf{y}_s). \end{aligned} \tag{1}$$

Wang and Dunson (2013) refer to $p(\theta|\mathbf{y}_s)$ as a "subposterior" distribution. Each data shard is assigned to a worker machine, which generates a Monte Carlo sample $\theta_1^{(s)}, \ldots, \theta_G^{(s)} \sim p(\theta|\mathbf{y}_s)$ from its associated subposterior. The main challenge with consensus Monte Carlo is how the simulated values from different workers can be combined into a global set of "consensus" draws approximating the full posterior distribution. This paper is primarily concerned with comparing different combination methods, which fall into three categories: averaging, resampling, and explicitly modeling the subposterior distributions. We also introduce a previously unexplored strategy of combining subposteriors using finite- and Dirichlet process mixture models.

CMC is orthogonal to other "big data" efforts in the Bayesian literature which focus on improving the poor scaling characteristics of typical MCMC algorithms. Dramatic speed increases can sometimes be achieved using subsampling methods (e.g. taking a random subsample of the data with each MCMC iteration, see for example Maclaurin and Adams (2014), Bardenet *et al.* (2014), Ahn *et al.* (2014), Quiroz *et al.* (2016), or Chen *et al.* (2016)), or by multi-threaded implementations of standard algorithms implemented on multi-core processors or massively multi-core graphics processing units (e.g. Suchard *et al.*, 2010; Lee *et al.*, 2010). These techniques are important complements to consensus Monte Carlo because they allow individual worker machines to quickly simulate from subposteriors based on larger data shards. However they only help with processor bottlenecks, not memory or disk, and because they assume the full data set is available they are largely irrelevant to the discussion of data center computing.

It should be mentioned that many statisticians rarely encounter problems where memory and disk are serious limitations. A standard laptop with 8GB of memory can hold 1 billion double-precision numbers. By any reasonable standard that is a lot of data, and most scientific questions can be resolved with less. If granted access to a data set with billions of observations, most statisticians would instinctively take a random sample of manageable size. However, there are applications where the full data set is needed, so randomly sampling the data is the wrong approach.

At many large Internet firms, such as Google, Amazon, and Netflix, the fundamental problem is to link individual users with specific content (e.g. a movie recommendation from Netflix, a shopping recommendation from Amazon, or the right web site in response to a Google search query). The number of users and the amount of content are incredibly large, and users' needs are both personal and evolving. Taking a sample in this setting to understand the preferences of "a typical user" might be helpful for certain specific analyses, but massive data are needed to fit the personalized user-level models required for effective service. While user-level modeling sounds like an embarrassingly parallel undertaking, some form of shrinkage across users and across content is needed to combat data sparsity, noting the obvious irony that even in big data problems, data about how a particular individual reacts to a particular stimulus remains scarce. Bayesian methods are motivated by the need for shrinkage, and because they provide uncertainty quantification which can be useful in managing the "explore-exploit tradeoff" associated with the online learning problem where models are repeatedly trained on data influenced by previously fit models (see, e.g. Scott, 2010, 2015).

The remainder of this article is structured as follows. Section 2 describes the consensus Monte Carlo algorithm in more detail, and reviews three existing methods for combining draws. Section 3 introduces a new combination method based on finite mixtures. Section 4 presents simulation experiments illustrating how each method performs in different scenarios, with particular attention paid to increasing problem dimensions and non-overlapping subposterior distributions. Section 5 gives a concluding discussion.

2 Consensus Monte Carlo

Equation (1) highlights the fact that the full posterior distribution is the product of S independent subposterior distributions. This suggests that one can independently obtain a Monte Carlo sample from each $p(\theta|\mathbf{y}_s)$, and then combine them to form the full posterior distribution. Imagine each worker as a member of a team tasked with doing a large analysis job. Each worker does part of the job based on partial information. When a worker's job is complete the finished product is sent off to the boss, who compiles the work done by all the employees into an organizational "consensus" belief reflecting the analysis done by all the individual team members. The algorithm is similar in spirit to meta-analysis, but with the constraint that the raw data from one worker cannot be arbitrarily accessed by another.

The consensus approach raises two questions. The first is how one should should deal with the prior distribution. Equation (1) suggests fractionating the prior and giving a piece to each worker. This can be a good strategy in some cases, but it can also lead to trouble in others. For example, when $p(\theta)$ is a weak-but-informative proper prior, $p(\theta)^{1/S}$ could be improper, which could endanger the propriety of $p(\theta|\mathbf{y}_s)$. We recognize this issue, but will not focus on it further. The second issue, and our primary concern, is how to combine the draws from the worker-level subposterior distributions. Three methods that have emerged in the literature are averaging, resampling, and subposterior modeling. These are described below.

2.1 Consensus through averaging (CMC)

Scott *et al.* (2016) showed that if all subposteriors are Gaussian, then averaging the draws produces draws from the full posterior. To see this, imagine $x_1 \sim p_1 = \mathcal{N}(\mu_1, \Sigma_1)$ is a draw from worker 1 and $x_2 \sim p_2 = \mathcal{N}(\mu_2, \Sigma_2)$ is an independent draw from worker 2. Imagining p_1 as the "prior" and p_2 as the "likelihood" leads to the familiar result that the full-data posterior is $p_1 p_2 \propto \mathcal{N}\left(\tilde{\mu}, \tilde{\Sigma}\right)$ where

$$\tilde{\Sigma}^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1} \qquad \text{and} \qquad \tilde{\mu} = \tilde{\Sigma}\left(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2\right). \tag{2}$$

Now consider the deviate $z = \tilde{\Sigma}\left(\Sigma_1^{-1}x_1 + \Sigma_2^{-1}x_2\right)$, which is a weighted average of x_1 and x_2. Clearly z is normal with mean $\tilde{\mu}$. Expanding the variance of the linear combination gives

$$\begin{aligned} Var(z) &= \tilde{\Sigma}\left(\Sigma_1^{-1}\Sigma_1\Sigma_1^{-1} + \Sigma_2^{-1}\Sigma_2\Sigma_2^{-1}\right)\tilde{\Sigma} \\ &= \tilde{\Sigma}(\Sigma_1^{-1} + \Sigma_2^{-1})\tilde{\Sigma} \\ &= \tilde{\Sigma}. \end{aligned}$$

For regular models with large sample sizes, the Bernstein-von Mises theorem (the "Bayesian central limit theorem") assures approximately Gaussian posteriors. Note that Gaussian subposteriors are a sufficient condition to justify averaging, but necessary conditions have yet to be established. Scott *et al.* (2016) provided several examples of non-Gaussian subposteriors where averaging is able to capture salient features of the posterior distribution, such as skewness, that would be missed by a direct normal approximation (i.e. by approximating each subposterior by its moments, and then combining the moments).

Averaging is a method with obvious flaws, but also powerful advantages. It is robust in the sense that it does not depend on an algorithm that might fail. It is computationally inexpensive, and it is invariant to dimension. The last point should not be taken lightly, because models that require big data tend to involve very large numbers of parameters. The main disadvantage is that there are situations where averaging does not make sense prima facie, such as when the posterior distribution is discrete or multi-modal.

2.2 Sequential Consensus Monte Carlo (SCMC)

Sequential Monte Carlo (SMC) is a seemingly natural way to combine draws from different workers by viewing data on remote machines as information with which to sequentially update locally produced MCMC draws.

2.2.1 Background on SMC

Sequential Monte Carlo methods (Doucet *et al.*, 2001) operate by exploiting the relationship between implicitly and explicitly weighted Monte Carlo samples. An explicitly weighted Monte Carlo sample describes a distribution p using a collection of points θ_j (often called "particles") and a corresponding set of weights w_j, satisfying

$$\frac{\sum_j f(\theta_j) w_j}{\sum_j w_j} \to E_p(f(\theta)). \tag{3}$$

An implicitly weighted sample weights each θ_j implicitly by its frequency, so the w_j's in equation (3) become uniform. An explicitly weighted sample can be turned into an implicitly weighted sample at any time by sampling $\{\theta_j\}$ with replacement using $\{w_j\}$ as sampling weights. Resampling the particles in this way produces multiple values of some θ_j, while others are omitted.

The canonical SMC algorithm (called "sampling with importance resampling") represents a posterior distribution $p(\theta|y_1, \ldots, y_t)$ using an implicitly weighted set of particles $\{\theta_j^{(t)}\}$. The distribution is updated to reflect a new observation y_{t+1} by attaching weight $w_j^{(t+1)} = p(y_{t+1}|\theta_j^{(t)})$ to $\theta_j^{(t)}$. An unweighted sample $\{\theta_j^{(t+1)}\}$ is then obtained by sampling $\{\theta_j^{(t)}\}$ with replacement using sampling weights $w_j^{(t+1)}$. Notice that the resampling step can be delayed if desired. For example, if y_{t+1} and y_{t+2} were observed simultaneously, one could incorporate them in a single update using $w_j^{(t+2)} = p(y_{t+2}|\theta_j^{(t)})p(y_{t+1}|\theta_j^{(t)})$. The resampling step is typically thought of as good housekeeping because it removes particles with inconsequential weights. However, there is a risk that a particle which is dropped during resampling (because its current weight is small) might have been viewed as important (and thus assigned a much larger weight) by future data. A related issue is that the particle ensemble can collapse, meaning that almost all weight attaches to a small subset of particles, or even a single particle. In some time series problems there are natural ways to perturb the particles between resampling steps, which can help prevent ensemble collapse, but opportunities to do this are very much problem dependent. The primary method of preventing ensemble collapse is to start the algorithm with a very large number of particles.

2.2.2 SCMC for big data

SMC can be applied to the consensus Monte Carlo problem as follows. As before, worker s generates J draws $\{\theta_j^{(s)}\} \sim p(\theta|\mathbf{y}_s)$, independently from other workers, using Markov chain Monte Carlo or some other Monte Carlo method. Each draw is then weighted by

$$w_j^{(s)} = \prod_{r \neq s} p(\mathbf{y}_r|\theta_j^{(s)}),$$

and resampled using these weights. Each worker in the SCMC algorithm produces J draws (not all of which are unique) so the algorithm as a whole yields $S \times J$ deviates.

SCMC requires one additional communication relative to the algorithm in Section 2.1, in which each worker broadcasts its particles to all the others. The extra communication is only a minor burden. SCMC also requires more computation, because each worker will have $J(S-1)$ likelihood calculations to perform in the weighting step. If necessary the extra work can be done in parallel, by replicating workers and distributing likelihood computations among replicates. However likelihood evaluation is typically much faster than simulation, so replicating workers will often be unnecessary.

SCMC is the most theoretically pure of the methods we consider. It requires no assumptions about the shape of the distributions being combined. It can be applied to arbitrary parameter spaces, so discrete parameter spaces can be handled gracefully, for example. It also allows the full prior distribution to be used in the initial MCMC step, so adjustments to the prior which are required by methods based on averaging can be avoided. The primary drawback is the potential for ensemble collapse, which experiments in Section 4 show to occur at disappointingly low dimensions. Increasing the number of draws J is unlikely to help, because the original particles are the result of a presumably expensive MCMC algorithm, which would make substantially longer initial runs infeasible.

2.3 Approximating the subposterior densities with kernels (KCMC)

The averaging method from Section 2.1 gives exact simulations from the desired distribution when the subposteriors are Gaussian. Neiswanger *et al.* (2013) observed that the method could be extended to non-Gaussian subposteriors by decomposing each subposterior into a mixture of Gaussians. The specific mixture chosen by Neiswanger *et al.* (2013) was a kernel density estimate, which is problematic for several reasons. The first is that it implies a very large number of mixture components, with one component centered on every subposterior draw. Second, the variances of the mixture components are chosen *a priori* without regard to the covariance structure of the data. A practical consequence of these issues is that kernel density estimates become unreliable after a relatively low number of dimensions. Scott and Sain (2005) suggest the limit is as low as six, although they note that KDE's might still be useful in higher dimensions for certain applications, such as classification. To deal with the very large number of mixture components, Neiswanger *et al.* (2013) employ a second MCMC algorithm when combining draws across multiple chains.

3 A consensus procedure based on mixtures (MxCMC)

Some of the issues that make kernel density estimates difficult to apply in high dimensions can be more gracefully handled by finite mixture models. Finite mixtures use many fewer mixture components than kernel density estimates, in part because the variance parameters in the mixture components adapt to fit the data. Though slower to fit than kernel density estimates, special purpose implementations of finite mixtures have been applied to problems of much larger dimension. Examples include Tadesse *et al.* (2005) ($d = 1000$) and McLachlan *et al.* (2003) ($d = 2000$), both of which involve problems where non-Gaussian structure is limited to a low dimensional subset or projection of the data.

Consider two workers which have produced draws $\theta_1^{(1)}, \ldots, \theta_{N_1}^{(1)} \sim p_1 = p(\theta|\mathbf{y}_1)$ and $\theta_1^{(2)}, \ldots, \theta_{N_2}^{(2)} \sim p_2 = p(\theta|\mathbf{y}_2)$. Approximate $p_1 \approx \pi_1 f_1 + \cdots + \pi_K f_K$ and $p_2 \approx w_1 g_1 + \cdots + w_M g_M$, where $(\pi_1, \ldots, \pi_K)$ and $(w_1, \ldots, w_M)$ are discrete probability distributions and $f_1, \ldots, f_K, g_1, \ldots, g_M$ are multivariate normal distributions with parameters $f_k = \mathcal{N}(\mu_{1k}, \Sigma_{1k})$, and $g_m = \mathcal{N}(\mu_{2m}, \Sigma_{2m})$. Then the product $p_1 p_2$ is also an approximate mixture of normals,

$$p_1 p_2 \approx \sum_k \sum_m \pi_k w_m f_k g_m \propto \sum_k \sum_m \tilde{w}_{km} \tilde{f}_{km}. \tag{4}$$

The mixture components from equation (4) are $\tilde{f}_{km} = \mathcal{N}\left(\tilde{\mu}_{km}, \tilde{\Sigma}_{km}\right)$, where

$$\tilde{\Sigma}_{km}^{-1} = \Sigma_{1k}^{-1} + \Sigma_{2m}^{-1} \qquad \text{and} \qquad \tilde{\mu}_{km} = \tilde{\Sigma}_{km}\left(\Sigma_{1k}^{-1}\mu_{1k} + \Sigma_{2m}^{-1}\mu_{2m}\right). \tag{5}$$

The mixing weight $\tilde{w}_{km}$ is not simply $\pi_k w_m$, but proportional to $\pi_k w_m \int f_k(\theta) g_m(\theta)\ d\theta$. Let $Q_{sk}(\mu) = (\mu_{sk} - \mu)^T \Sigma_{sk}^{-1}(\mu_{sk} - \mu)$. Then the mixing weights can be written

$$\tilde{w}_{km} \propto \pi_k w_m \frac{|\Sigma_{1k}^{-1}|^{\frac{1}{2}} |\Sigma_{2m}^{-1}|^{\frac{1}{2}}}{|\Sigma_{1k}^{-1} + \Sigma_{2m}^{-1}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}[Q_{1k}(\tilde{\mu}_{km}) + Q_{2m}(\tilde{\mu}_{km})]\right). \tag{6}$$

Equation (6) shows that equation (4) gives greater weight to pairs of components that are both heavily weighted in their respective subposterior mixture approximations (so that both π_k and w_m are large), have similar means (where both μ_{1k} and μ_{2m} are close to $\tilde{\mu}_{km}$), and similarly sized information matrices (where both Σ_{1k}^{-1} and Σ_{2m}^{-1} are large relative to their sum).

3.1 Combining draws from pairs of workers using local averaging

Although the mixture approximation in equation (4) could be sampled directly, for reasons analogous to the discussion of averaging in Section 2.1 it is preferable to combine the original draws rather than sample from an approximate model of their distribution. The mixture approximations of p_1 and p_2 are presumably imperfect, so the hope is that minor deviations from normality that would be lost by directly sampling from the approximation will be preserved through averaging.

If we were to simulate directly from equation (4), an obvious method would be to first simulate a draw (k, m) from the discrete distribution with probabilities $\{\tilde{w}_{km}\}$, then simulate a deviate from $\tilde{f}_{km}$. The proposed local averaging procedure replaces the $\tilde{f}_{km}$ draw with a weighted average of a random draw from component k of worker 1 and component m from worker 2. To identify these draws, let $\mathbf{P}_1$ and $\mathbf{P}_2$ be the matrices of posterior allocation probabilities for the samples from workers 1 and 2, respectively. That is, $\mathbf{P}_1$ is a matrix with K columns and N_1 rows, with elements $p_{ik} \propto \pi_k f_k(\theta_i^{(1)})$, with $\mathbf{P}_2$ defined similarly for worker 2. Sample an index i with sampling weights proportional to column k of $\mathbf{P}_1$, and a random index j with weights proportional to column m of $\mathbf{P}_2$. The consensus draw is obtained by averaging $\theta_i^{(1)}$ and $\theta_j^{(2)}$ with weights proportional to Σ_{1k}^{-1} and Σ_{2m}^{-1} as in Section 2.1.

3.2 Choosing a mixture decomposition

There are several potentially important details regarding the finite mixture decomposition that are beyond the scope of the current investigation. A finite mixture approximation of $p(\theta|\mathbf{y}_s)$ is typically non-unique, and it is not clear how one would want to resolve a tie between two mixtures that approximate $p(\theta|\mathbf{y}_s)$ roughly equally well. Mixtures with smaller numbers of components should probably be preferred, as should mixtures with components that are well populated. It would also be helpful if this stage of model fitting required little human supervision. Dirichlet process mixtures are appealing because they do not require the number of mixture components to be specified. Oversaturated finite mixtures (Rousseau and Mengersen, 2011) can also be used to handle uncertainty about the number of mixture components.

The parameters of the mixture approximation can be obtained either using MCMC or as point estimates through maximum likelihood or maximum *a posteriori* estimation (e.g. McLachlan and Peel, 2000), or by variational methods (e.g. Blei *et al.*, 2006; McGrory and Titterington, 2007). The advantage of point estimates is that there is only a single mixture decomposition to consider, which speeds up computing in the local averaging procedure. A single multinomial draw of size N can replace N individual draws from $\{\tilde{w}_{km}\}$, the matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ need only be formed once, and subsamples from $\{\theta_i^{(1)}\}$ and $\{\theta_j^{(2)}\}$ can be taken in batch. To the extent that some mixture decompositions work better than others, and we don't know *a priori* which ones they will be, it is more conservative to fit the mixture approximation using MCMC methods that can average over multiple decompositions. MCMC is potentially much slower because the procedure from Section 3.1 must be done one draw at a time, with different parameters for each draw.

The time required to fit a finite mixture distribution can be considerable in high dimensions, with a major piece of the computational cost arising from determinants and quadratic forms involving the component variances Σ_k. Suchard *et al.* (2010) have demonstrated that GPU-based algorithms can reduce the computing time in low dimensional mixtures, and their method can likely be used in much higher dimensions, although some form of parametric modeling for Σ_k will become be necessary as the problem dimension grows.

3.3 Combining draws from many workers

If there are only a few workers, then equation (4) can be extended in obvious ways to accommodate a product of more subposterior densities. However the number of mixture components in the approximating density is the product of the number of components in each subposterior mixture, which rapidly becomes untenable as the number of workers grows. An alternative is to apply the pairwise combination procedure from Section 3.1 multiple times. Doing so requires $\log_2 S$ rounds of combination, with a finite mixture approximation being fit to the draws produced at the end of each round of combination, other than the last.

4 Examples

We now turn to a series of experiments comparing the various methods of forming consensus described above. We study three scenarios. In each, the subposteriors are chosen so that their product will have a known closed form which can be easily and exactly sampled, so that the consensus results can be compared with ground truth.

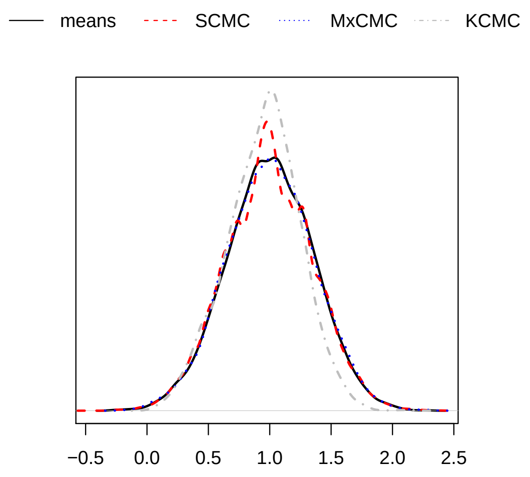
All examples assume eight synthetic "workers." Computations for KCMC used the implementation provided by Miroshnikov and Conlon (2014). Three rounds of pairwise combinations were used for the mixture-based consensus algorithm MxCMC. Mixtures were fit with a Dirichlet process mixture of multivariate normal distributions using the collapsed Gibbs sampler, described as "Algorithm 3" in Section 3 of Neal (2000). The Gibbs sampler was hand coded in C++, but none of the other optimizations mentioned in Section 3.2 were attempted.

4.1 Multivariate normal posteriors

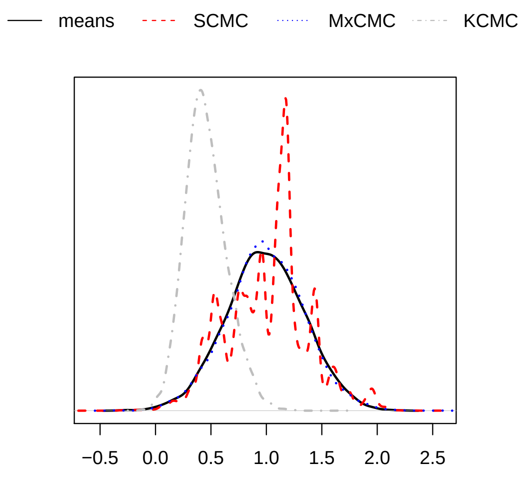
The first example focuses on multivariate normal posteriors of increasing dimension. In this setting averaging gives perfect simulations, so the point is to study how SMC, kernel, and mixture based methods perform in regular "well behaved" problems. Simulations from each subposterior distribution are drawn from identical $\mathcal{N}(\mu, \Sigma)$ distributions, where $\mu = (1, \ldots, d)$, for different values of d, and Σ is a "random effects" matrix with off-diagonal elements all set to .9 and diagonal elements set to 1.0. Each worker begins by simulating 10,000 independent draws from its subposterior distribution.

Figure 1 shows kernel density estimates for the first element of the consensus posterior draws produced by the different consensus methods. The remaining $d-1$ elements are qualitatively similar, albeit with different means. There are 10,000 consensus draws for the mean-based and mixture based CMC algorithms, but 80,000 draws (10,000 from each worker) for the SCMC algorithm. When $d = 5$ all three algorithms perform acceptably well. As the dimension grows to 10 and 20 the discreteness from SCMC becomes unacceptably large. In another context one might argue that the discreteness could be smoothed out by increasing the number of initial particles, but keep in mind that this is a proxy for a hypothetical subposterior distribution for which simulating 10,000 draws might represent a considerable amount of work. The KCMC method performs well when $d = 5$, but it gets the location, scale, and shape wrong in higher dimensions. The MxCMC algorithm works well in all cases, as the underlying mixture model correctly detects that only a single mixture component is needed. This experiment suggests that both SCMC and KCMC should be used with caution in ten or more dimensions.

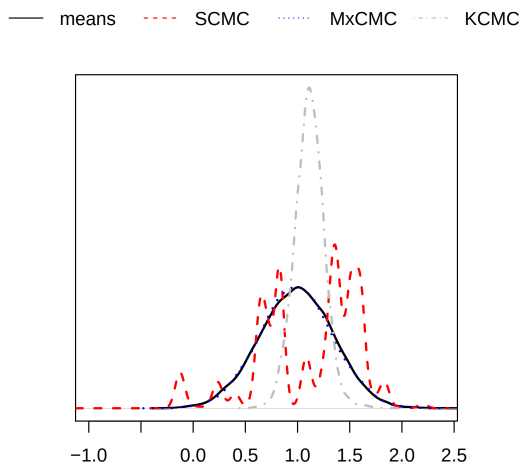
In regards to time, both CMC and SCMC took less than 1 second to run, whereas MxCMC needed several minutes (25 in the $d = 20$ case) to fit the necessary mixtures. The time required to fit the mixtures for MxCMC might be reduced by a more efficient fitting algorithm, but it is a potentially serious issue with the method, which otherwise performs quite well in this scenario.



(a)

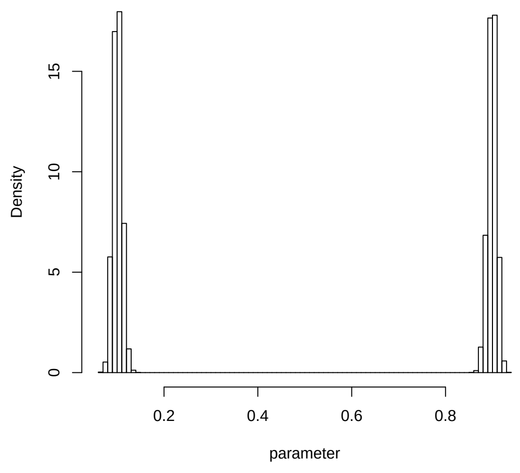


(b)

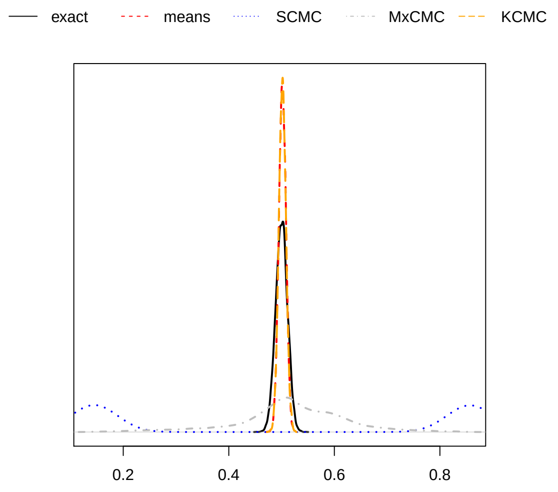


(c)

Figure 1: *Monte Carlo density estimates for the first element of a d-dimensional multivariate normal deviate. (a)* $d = 5$*, (b)* $d = 10$*, (c)* $d = 20$*. The means and MxCMC results overplot one another.*



(a)



(b)

Figure 2: *Subposterior and consensus draws for the beta distribution example in Section 4.2. (a) Subposterior draws from workers with very different data. (b) Consensus Monte Carlo approximations to the exact posterior density. The means and kernel based methods are nearly identical. SCMC is bimodal and far from the center. MxCMC is centered in the right place but the variance is much too wide.*

4.2 Non-overlapping beta posteriors

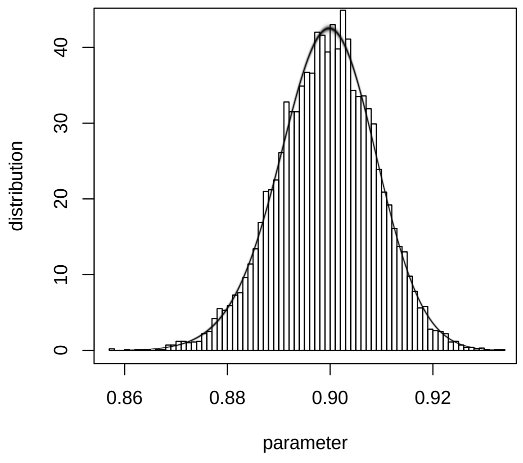
The second example considers subposterior distributions represented by non overlapping draws. The example is in one dimension, but it illustrates a phenomenon that one expects in higher dimensions because of the curse of dimensionality. This scenario challenges all four consensus methods.

Consider two subposterior distributions based on binomial data. The first has $y_1 = 900$ successes out of $n_1 = 1000$ trials. The second has $y_2 = 100$ successes out of $n_2 = 1000$ trials. Each subposterior is assigned a $Be(.5, .5)$ prior, corresponding to a $Be(1, 1)$ prior for the overall system, though in this case the sample sizes are large enough that the prior has little effect. Figure 2(a) shows histograms of draws from the subposteriors being combined. Figure 2(b) compares the four consensus methods with the distribution of exact draws. The means and kernel density methods both capture the mean and essential shape of the correct distribution, but fail to describe the variance correctly. The failure occurs because the variance of the beta distribution is a function of the mean, and the estimated variances at the subposterior level are computed near zero and one, where the variance is smaller.

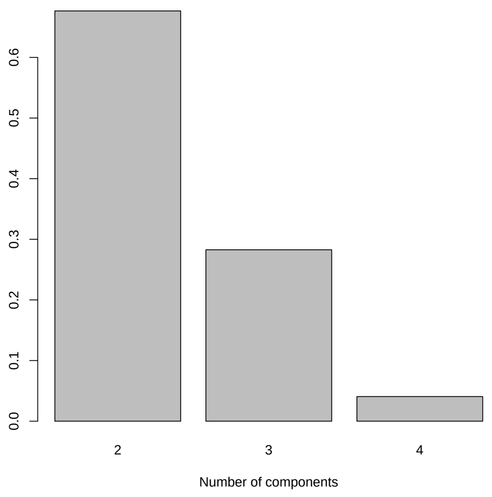
All the mass of the full posterior distribution is in a region that is far into the tails of both subposteriors. This causes problems for both the SCMC and MxCMC methods. SCMC winds up choosing the smallest draws from worker 1 and the largest draws from worker 2, with neither set of draws in a relevant region for the full posterior. The result is a bimodal approximation when the truth is unimodal. MxCMC produces draws centered on the correct region, but with unreasonably large variance. To understand why MxCMC struggles, examine Figure 3, which describes the mixture decomposition of the subposterior draws from worker 1, the right-hand mode in Figure 2(a). Figure 3(a) shows the histogram of the draws being fit by the mixture, while superimposing pointwise marginal density estimates of the density curve implied by the mixture model, verifying that the mixture is fitting the distribution well. Panel (b) plots the posterior distribution of the number of mixture components in the Dirichlet process, showing that the mixture is using 2, 3, or 4 components. Panels (c) and (d) plot the marginal distributions of the mean and variance parameters describing the mixture components in the 2-component case. Plots for 3 and 4 components are not shown, but are similar. Collectively, Figure 3 shows that the mixture approximation captures the slight left skew of the beta distribution by adding normal components with smaller mean and larger variance than the primary component. As a mirror image to worker 1, the mixture decomposition of worker 2 has primary component describing data near 0.1, with extra components capturing the skewness in the upper tail.

When the MxCMC algorithm combines the subposterior draws, equation (6) overwhelmingly favors the high variance components with means closer to 0.5. The uncertainty in the variance parameters, which are the weights used in the computation, translates into the wide distribution seen in Figure 2(b). The mixture approximation is obviously doing a good job of describing the data in regions of high density, but it does a poor job of modeling the distant tails where the full posterior has virtually all of its mass. I also tried replacing the MCMC ensemble by a single mixture model parameters set to MAP estimates. This resulted in consensus estimates that had lower variance than MxCMC in Figure 2(b), but with substantial bias.

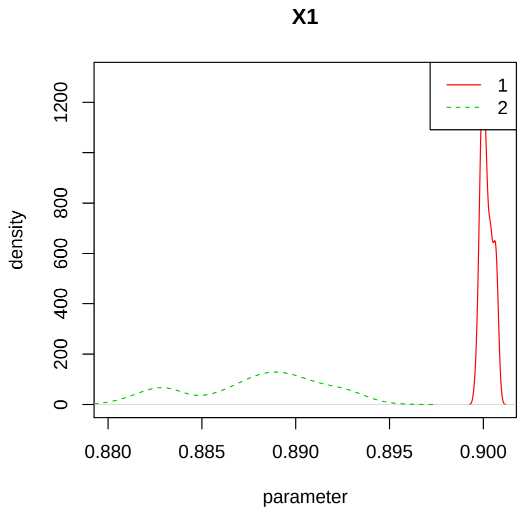
This experiment suggests that sophisticated methods of combining draws can fail when the subposterior draws do not have sufficient overlap, which is likely to occur for at least some low



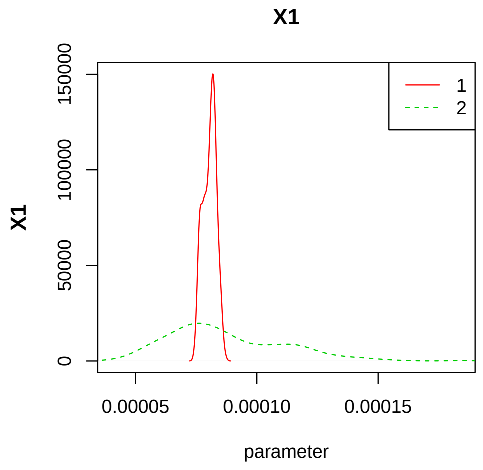
(a)



(b)



(c)



(d)

Figure 3: *Mixture approximation to the worker level data from Figure 2. (a) The marginal distribution of the density curve superimposed on the histogram of subposterior draws . (b) Distribution of the number of mixture components. (c) Distribution of mean parameters, and (d) variances parameters conditional on 2 mixture components.*

dimensional projections of parameters in high dimensional problems.

4.3 Inverse Wishart

The third simulation experiment assumes inverse Wishart subposterior distributions of dimension $d \times d$, denoted $\mathcal{IW}_d(\nu, S)$, where ν can be interpreted as a sample size and S as a sum of squares matrix. The inverse Wishart distribution is a multivariate distribution with a Gaussian limit, but it is decidedly non-Gaussian for moderate values of ν. For the results presented here, subposterior i has $\nu = (d+1) + 20$ and $S = S_0 + S_i$, where S_0 is the random effects variance matrix used in Section 4.1, and S_i is the matrix sum of squares from 20 $\mathcal{N}(0, S_0)$ deviates.

Two test problems were considered, one with $d = 3$ and one with $d = 5$. Deviates from the inverse Wishart distribution are symmetric positive definite matrices, which were vectorized so that only unique elements were considered. Thus the 3×3 case is a 6-dimensional problem and the 5×5 case is 15 dimensional. Increasing d much beyond this proved to be a challenge for the Dirichlet process implementation used for this exercise.

Figure 4 plots marginal density estimates for the first four unique elements in the 3×3 example, while Figure 5 plots the first four elements of the 5×5 example. As with previous experiments, the SCMC estimates begin to fall apart in relatively low dimensions. The kernel based estimate exhibits a consistent bias in both examples. Interestingly, the kernel density estimate gets the shape of the consensus density about right in Figure 4. While the kernel method gets the shape wrong in Figure 5 it at least matches the shape produced by MxCMC.

The CMC and MxCMC posterior estimates are very close in Figure 4, but they begin to diverge in Figure 5, where MxCMC does a better job describing the shape of the marginal distributions, but mean-based CMC does a better job of capturing the distribution's center. The location shift in the MxCMC algorithm is much smaller than KCMC, but it is still evident in Figure 5, and the direction of the bias is the same for the two methods.

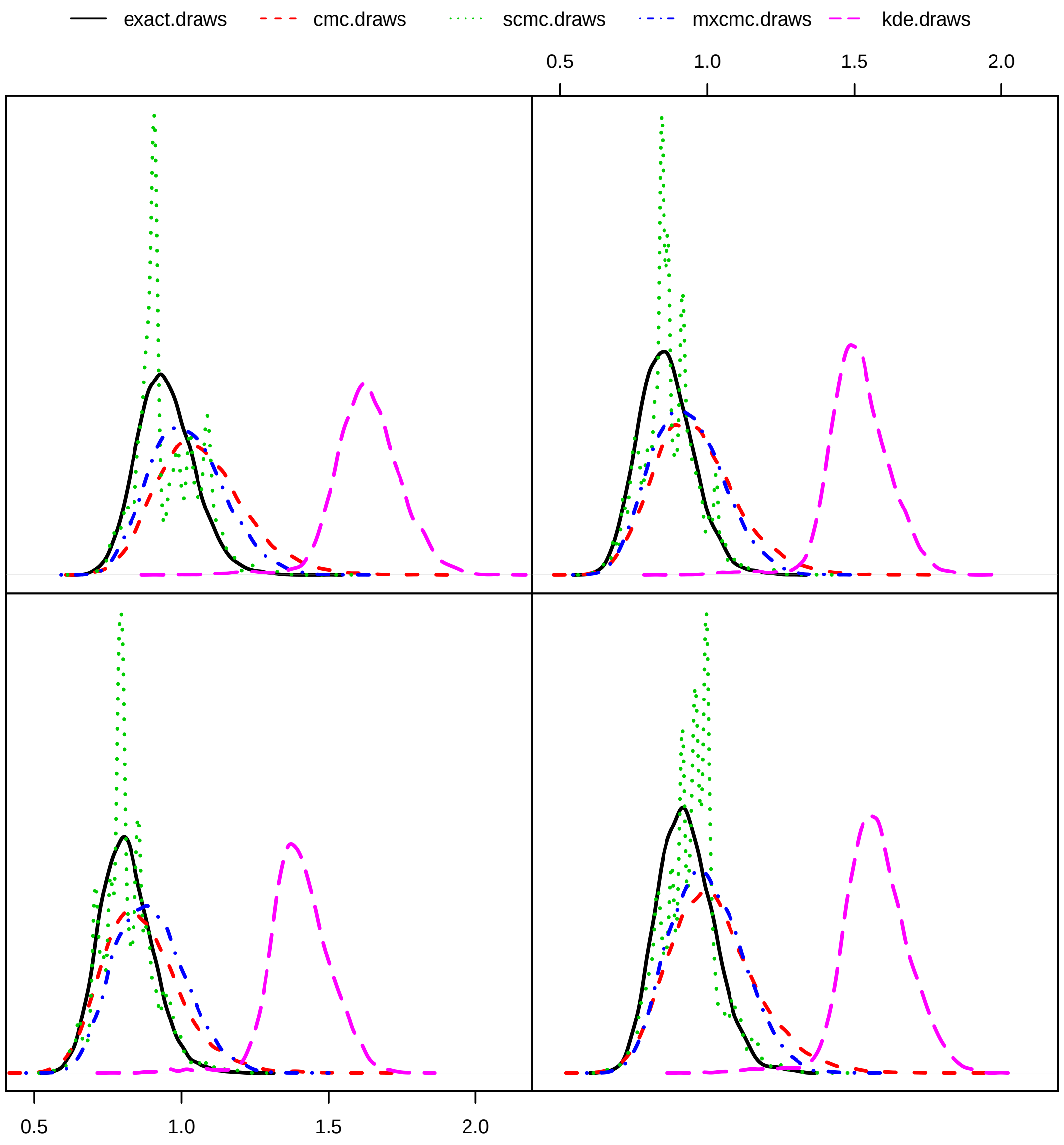


Figure 4: *First four elements in Inverse Wishart draws for a* 3×3 *random variable.*

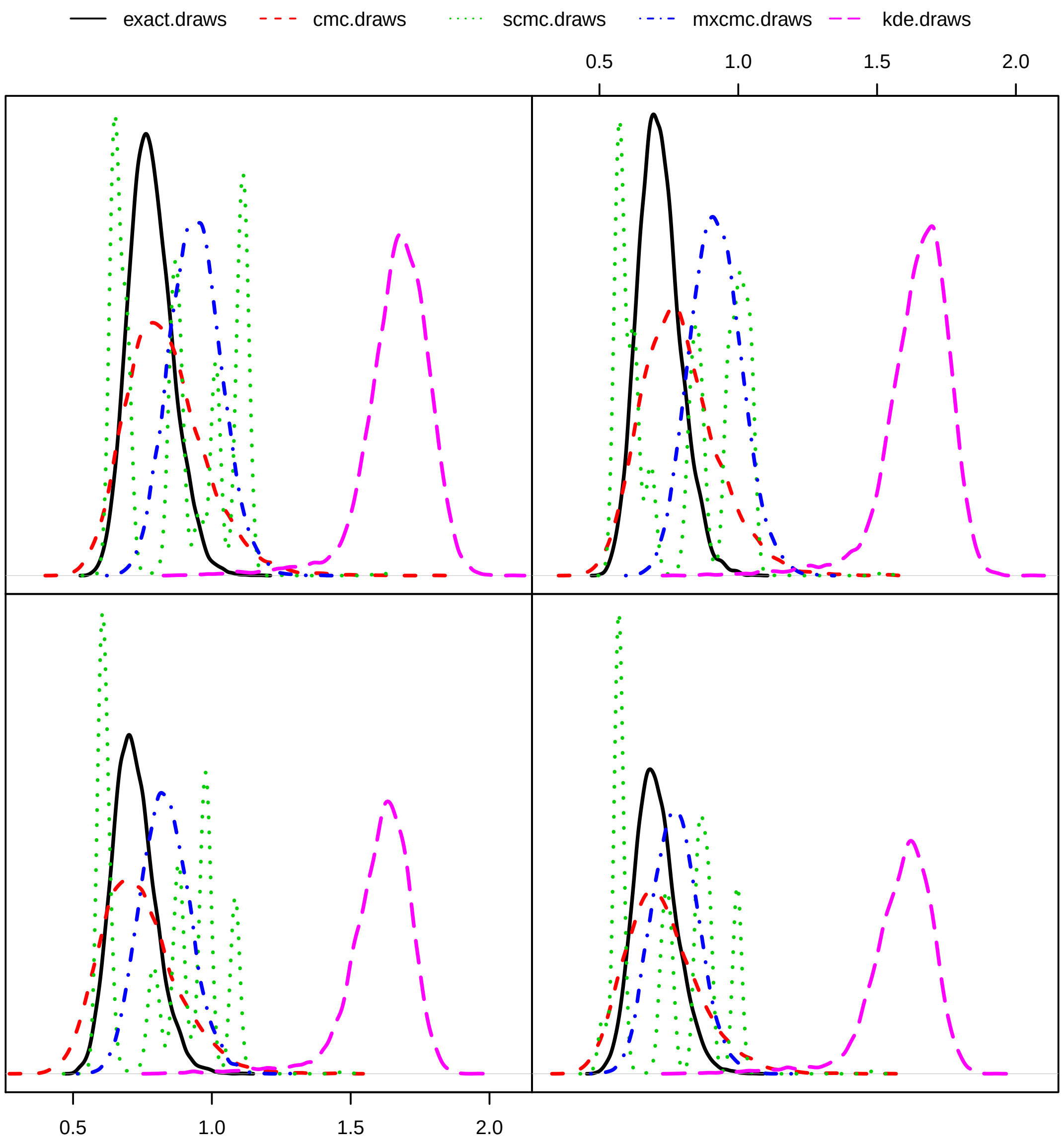


Figure 5: *First four elements in Inverse Wishart draws for a 5×5 random variable.*

5 Conclusion

The simulations in Section 4 were all performed under scenarios where averaging had a reasonable shot of doing well. The subposteriors were unimodal, and the parameters varied smoothly over subsets of $\mathbb{R}^d$. In these settings averaging is hard to beat, primarily because it can be applied to problems of arbitrary dimension. It is rare to encounter a problem with only 5 or 6 parameters that is sufficiently complex to require a big data solution, yet this seems to be the threshold where SMC and kernel methods break down.

Averaging has obvious limitations, and it is not hard to come up with examples where it does not perform very well. Several such examples can be found in Srivastava *et al.* (2015), Wang *et al.* (2015), and Neiswanger *et al.* (2013), among others. Thus there is clearly a need for alternative consensus methods to averaging. However all the methods we have considered seem to fail for one reason or another as the dimension grows. Particle based resampling methods are well known to be challenged by dimension, and there nothing obvious about the structure of large distributed data sets that lends itself to particle regeneration. Likewise, kernel based methods also fail in relatively low dimensions, so approximating equation (1) with a product of kernel density estimates leads to the poor results we have seen for KCMC.

The mixture based MxCMC method introduced here is theoretically interesting, but the mechanics of fitting mixtures to high dimensional distributions is a serious practical challenge. Yet of the general methods considered, its challenges are the most likely to be overcome.

References

Ahn, S., Shahbaba, B., Welling, M., *et al.* (2014). Distributed stochastic gradient MCMC. In *ICML*, 1044–1052.

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

Blei, D. M., Jordan, M. I., *et al.* (2006). Variational inference for Dirichlet process mixtures. *Bayesian analysis* **1**, 121–144.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)* **26**, 4.

Chen, H., Seita, D., Pan, X., and Canny, J. (2016). An efficient minibatch acceptance test for Metropolis-Hastings. *arXiv preprint arXiv:1610.06848v1* .

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

Doucet, A., De Frietas, N., and Gordon, N. (2001). *Sequential Monte Carlo in Practice*. Springer.

Lee, A., Yao, C., Giles, M. B., Doucet, A., and Holmes, C. C. (2010). On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Journal of Computational and Graphical Statistics* **19**, 769–789.

Maclaurin, D. and Adams, R. P. (2014). Firefly Monte Carlo: Exact MCMC with subsets of data.

McGrory, C. A. and Titterington, D. (2007). Variational approximations in Bayesian model selection for finite mixture distributions. *Computational Statistics & Data Analysis* **51**, 5352–5367.

McLachlan, G. J. and Peel, D. (2000). *Finite Mixture Models*. John Wiley & Sons, New York.

McLachlan, G. J., Peel, D., and Bean, R. (2003). Modelling high-dimensional data by mixtures of factor analyzers. *Computational Statistics & Data Analysis* **41**, 379–388.

Miroshnikov, A. and Conlon, E. (2014). *parallelMCMCcombine: Methods for combining independent subset Markov chain Monte Carlo (MCMC) posterior samples to estimate a posterior density given the full data set*. R package version 1.0.

Neal, R. M. (2000). Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics* **9**, 249–265.

Neiswanger, W., Wang, C., and Xing, E. (2013). Asymptotically exact, embarrassingly parallel MCMC. *arXiv preprint arXiv:1311.4780* .

Quiroz, M., Villani, M., and Kohn, R. (2016). Exact subsampling MCMC.

Rousseau, J. and Mengersen, K. (2011). Asymptotic behaviour of the posterior distribution in overfitted mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **73**, 689–710.

Scott, D. W. and Sain, S. R. (2005). Multidimensional density estimation. *Handbook of statistics* **24**, 229–261.

Scott, S. L. (2010). A modern Bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry* **26**, 639–658. (with discussion).

Scott, S. L. (2015). Multi-armed bandit experiments in the online service economy. *Applied Stochastic Models in Business and Industry* **31**, 37–45.

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**, 78 – 88.

Srivastava, S., Li, C., and Dunson, D. B. (2015). Scalable bayes via barycenter in wasserstein space. *arXiv preprint arXiv:1508.05880* .

Suchard, M. A., Wang, Q., Chan, C., Frelinger, J., Cron, A., and West, M. (2010). Understanding GPU programming for statistical computation: Studies in massively parallel massive mixtures. *Journal of Computational and Graphical Statistics* **19**, 419–438.

Tadesse, M. G., Sha, N., and Vannucci, M. (2005). Bayesian variable selection in clustering high-dimensional data. *Journal of the American Statistical Association* **100**, 602–617.

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

Wang, X., Fangjian, G., Heller, K. A., and Dunson, D. B. (2015). Parallelizing MCMC with random partition trees. *resaerchgate.net DOI: 10.13140/RG.2.1.2921.4883* .