Merging MCMC Subposteriors through Gaussian-Process Approximations

Christopher Nemeth and Chris Sherlock

Department of Mathematics and Statistics, Lancaster University, U.K.

May 30, 2016

Abstract

Markov chain Monte Carlo (MCMC) algorithms have become powerful tools for Bayesian inference. However, they do not scale well to large-data problems. Divide-and-conquer strategies, which split the data into batches and, for each batch, run independent MCMC algorithms targeting the corresponding subposterior, can spread the computational burden across a number of separate workers. The challenge with such strategies is in recombining the subposteriors to approximate the full posterior. By creating a Gaussian-process approximation for each log-subposterior density we create a tractable approximation for the full posterior. This approximation is exploited through three methodologies: firstly a Hamiltonian Monte Carlo algorithm targeting the expectation of the posterior density provides a sample from an approximation to the posterior; secondly, evaluating the true posterior at the sampled points leads to an importance sampler that, asymptotically, targets the true posterior expectations; finally, an alternative importance sampler uses the full Gaussian-process distribution of the approximation to the log-posterior density to re-weight any initial sample and provide both an estimate of the posterior expectation and a measure of the uncertainty in it.

Keywords: Big data; Markov chain Monte Carlo; Gaussian processes; distributed importance sampling.

1 Introduction

Markov chain Monte Carlo (MCMC) algorithms are popular tools for sampling from Bayesian posterior distributions in order to estimate posterior expectations. They benefit from theoretical guarantees of asymptotic convergence of the estimators as the number of MCMC samples grows. However, whilst asymptotically exact, they can be computationally expensive when
applied to datasets with a large number of observations $n$. Indeed, the cost of generating one sample from the MCMC algorithm is at best $O(n)$ as the posterior distribution of the model parameters, conditional on the entire data set, must be evaluated at each MCMC iteration. For very large $n$, therefore, MCMC algorithms can become computationally impractical.

Research in the area of MCMC for big data can be broadly split into two streams: those which utilise one core of the central processing unit (CPU) and those that distribute the work load across multiple cores, or machines. For the single processor case, the computational cost of running MCMC on the full data set may be reduced by using a random subsample of the data at each iteration (Quiroz et al., 2014; Maclaurin and Adams, 2014; Bardenet et al., 2014); however, the mixing of the MCMC chain can suffer as a result. Alternatively, the Metropolis-Hastings acceptance step can be avoided completely by using a stochastic gradient algorithm (Welling and Teh, 2011; Chen et al., 2014), where subsamples of the data are used to calculate unbiased estimates of the gradient of the log-posterior. Consistent estimates of posterior expectations are obtained as the gradient step-sizes decrease to zero (Teh et al., 2014). While popular, subsampling methods do have the drawback that the data must be independent and the whole data set must be readily available at all times, and therefore data cannot be stored across multiple machines.

Modern computer architectures readily utilise multiple cores of the CPU for computation, but MCMC algorithms are inherently serial in implementation. Parallel MCMC, where multiple MCMC chains, each targeting the full posterior, are run on separate cores, or machines, can be easily executed (Wilkinson, 2005), however this does not address the big-data problem as each machine still needs to store and evaluate the whole data set. In order to generate a significant computational speed-up the data set must be partitioned into disjoint batches, where independent MCMC algorithms are executed on separate batches on independent processors (Huang and Gelman, 2005). Using only a subset of the entire data means that the MCMC algorithm is targeting a partial posterior, herein referred to as a subposterior. This type of parallelisation is highly efficient as there is no communication between the parallel MCMC chains. The main challenge is to then reintegrate the samples from the separate MCMC chains to approximate the full posterior distribution. Scott et al. (2013) create a Gaussian approximation for the full posterior by taking weighted averages of the means and variances of the MCMC samples from each batch; this procedure is exact when each subposterior is Gaussian, and can work well approximately in non-Gaussian scenarios. Neiswanger et al. (2013) avoid the Gaussian assumption by approximating the subposteriors using kernel density estimation, however, kernel density approximations scale poorly in high dimensions (Liu et al., 2007). Also, the upper bounds on the mean squared error given in Neiswanger et al. (2013) grow exponentially with the number of batches, which is problematic in big data scenarios where the computational benefit of parallelisation is proportional to the number of available processors.

Previous approaches used to merge the product of subposterior densities have solely relied on the parameter samples outputted from each MCMC algorithm, but have neglected to utilise
the subposterior densities which are calculated when evaluating the Metropolis-Hastings ratio. We place Gaussian-process (GP) priors on the log-density of each subposterior. The resulting approximation to the log of the full posterior density is a sum of Gaussian-processes, which is itself a Gaussian-process. From this we may obtain not only a point estimate of any expectation of interest, but also a measure of uncertainty in this estimate.

Starting from this Gaussian-process approximation to the full log-posterior density, we investigate three approaches to approximating the posterior. Firstly, an efficient Hamiltonian Monte Carlo (HMC) algorithm (Neal, 2010) which targets the expectation of the posterior density (the exponential of the combined GP); samples from this provide our first means of estimating expectations of interest. Secondly, the HMC sample values may be sent to each of the cores, with each core returning the true log-subposterior at each of the sample points. Combining these coincident log-subposterior values provides the true posterior at the sampled points, which in turn provides importance weights for the HMC sample, leading to asymptotically consistent estimates of posterior expectations. The practitioner may wish to avoid the complexities and computational expense of running HMC on the expectation of the exponential of the GP and of calculating the true sub-posteriors at a sample of points. We, therefore, also consider an importance proposal based upon any approximation to the true posterior and obtain repeated samples of importance weights by repeatedly sampling realisations of the GP approximation to the log-posterior. This provides both an estimate of any expectation of interest and a measure of its uncertainty.

This paper is structured as follows. Section 2 reviews the parallel MCMC approach for sampling from the posterior, the HMC algorithm and importance sampling. Section 3 then outlines the creation of our Gaussian-process approximation for each of the individual subposteriors, and for combining these. In Section 4 we detail three methods for approximating posterior expectations, each utilising the combined Gaussian-process approximation. Section 5 highlights through two toy models, and two large scale logistic regression problems, that our method offers significant improvements over competing methods when approximating non-Gaussian posteriors. We conclude with a discussion.

2 Bayesian inference and MCMC

Consider a data set \( \mathcal{Y} = \{y_1, y_2, \ldots, y_n\} \) where we assume that the data are conditionally independent with a likelihood \( \prod_{i=1}^{n} p(y_i|\vartheta) \), where \( \vartheta \in \Theta \subseteq \mathbb{R}^d \) are model parameters. Assuming a prior \( p(\vartheta) \) for the parameters, the posterior distribution for \( \vartheta \) given \( \mathcal{Y} \) is

\[
\pi(\vartheta) = p(\vartheta|\mathcal{Y}) \propto p(\vartheta) \prod_{i=1}^{n} p(y_i|\vartheta). \tag{1}
\]

Alternatively, the data set \( \mathcal{Y} \) can be partitioned into \( C \) batches \( \{\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_C\} \) where we
define a subposterior operating on a subset of the data $Y_c$ as

$$
\pi_c(\theta) = p(\theta|Y_c) \propto p(Y_c|\theta)p(\theta)^{1/C},
$$

(2)

where $p(\theta)$ is chosen so that $p(\theta)^{1/C}$ is proper. The full posterior is given as the product of the subposteriors $\pi(\theta) \propto \prod_{c=1}^{C} \pi_c(\theta)$. In this setting we no longer require conditional independence of the data, but rather independence between the batches $\{Y_c\}_{c=1}^{C}$, where now the data in each batch can exhibit an arbitrary dependence structure.

Creating an approximation to the posterior, $\pi(\theta)$, commences with sampling from each of the subposteriors $\pi_c(\theta)$ independently in parallel, where, given the independence between data subsets, there is no communication exchange between the MCMC algorithms operating on the subposteriors. This type of parallelisation is often referred to as embarrassingly parallel (Neiswanger et al., 2013). The challenge then lies in combining the subposteriors, for which we propose using Gaussian-process approximations.

In this paper we introduce the Hamiltonian Monte Carlo (HMC) algorithm as one possible MCMC algorithm that can be used to sample from $\pi_c(\theta)$. Moreover, we use HMC in Section 4 to sample from an approximation to the full posterior, $\pi(\theta)$. Other MCMC algorithms, including the random walk Metropolis (Roberts et al., 1997), Metropolis adjusted Langevin algorithm (Roberts and Rosenthal, 1998) and adaptive versions of these (e.g. Andrieu and Thoms, 2008) can also be used.

### 2.1 Hamiltonian Monte Carlo

We now provide a brief overview of Hamiltonian Monte Carlo and its application in this paper; the interested reader is referred to Neal (2010) for a full and detailed review. The HMC algorithm considers the sampling problem as the exploration of a physical system with $-\log \pi(\theta)$ corresponding to the potential energy at the position $\theta$. We then introduce artificial momentum variables $\varphi \in \mathbb{R}^D$, with $\varphi \sim \mathcal{N}(0,M)$ being independent of $\theta$. Here $M$ is a mass matrix that can be set to the identity matrix when there is no information about the target distribution. This scheme now augments our target distribution so that we are now sampling $(\theta, \varphi)$ from their joint distribution

$$
\pi(\theta, \varphi) \propto \exp \left( \log \pi(\theta) - \frac{1}{2} \varphi M^{-1} \varphi \right),
$$

(3)

the logarithm of which equates to minus the total energy of the system. Samples from the marginal distribution of interest, $\pi(\theta)$, are obtained by discarding the $\varphi$ samples.

We can sample from the target distribution by simulating $\theta$ and $\varphi$ through fictitious time $\tau$ using Hamilton’s equations (see Neal (2010) for details)

$$
d\theta = M^{-1} \varphi d\tau,
\quad
d\varphi = \nabla_{\theta} U(\theta) d\tau.
$$

(4)
The differential equations in (4) are intractable and must be solved numerically. Several numerical integrators are available which preserve the volume and reversibility of the Hamiltonian system (Girolami and Calderhead, 2011), the most popular being the leapfrog, or Stormer-Verlet integrator. The leapfrog integrator takes $L$ steps, each of size $\epsilon$, on the Hamiltonian dynamics (4), with one step given as follows:

$$
\varphi_{\tau + \frac{\epsilon}{2}} = \varphi_{\tau} + \frac{\epsilon}{2} \nabla \log \pi(\vartheta_{\tau})
$$

$$
\vartheta_{\tau + \epsilon} = \vartheta_{\tau} + \epsilon M^{-1} \varphi_{\tau + \frac{\epsilon}{2}}
$$

$$
\varphi_{\tau + \epsilon} = \varphi_{\tau + \frac{\epsilon}{2}} + \frac{\epsilon}{2} \nabla \log \pi(\vartheta_{\tau + \epsilon})
$$

Using a discretisation introduces a small loss or gain in the total energy, which is corrected through a Metropolis-Hastings accept/reject step. The full HMC algorithm is given in Algorithm 3 in Appendix A.

The HMC algorithm has a step-size parameter $\epsilon$ and number of leap frog steps $L$ which need to be tuned. The performance of the algorithm is highly dependent on the tuning of the parameters. One way to tune the algorithm is to optimise the parameters such that the acceptance rate is approximately 65% (Beskos et al., 2013). Alternatively, the parameters could be adaptively tuned (Wang et al., 2013); for this paper we use the popular NUTS sampler Hoffman and Gelman (2014), which tunes the trajectory length $L$ to avoid the sampler doubling back on itself. The HMC algorithm can be efficiently implemented using the popular STAN software package. The STAN modelling language automatically tunes the HMC algorithm, and by using efficient automatic differentiation, the user need only express their posterior model.

### 2.2 Importance sampling

A popular alternative to MCMC for estimating posterior expectations is the importance sampler (Robert and Casella, 1999). Given a proposal density, $q(\theta)$, and an unnormalised posterior density, $\pi(\theta)$, importance sampling (e.g. Geweke, 1989) aims to estimate expectations of some measurable function of interest, $h(\theta)$ by sampling from $q$. The starting point is

$$
\mathbb{E}_{\pi/Z}[h(\theta)] = \frac{1}{Z} \int h(\theta) \frac{\pi(\theta)}{q(\theta)} q(\theta) d\theta = \frac{1}{Z} \mathbb{E}_q[h(\theta) \overline{\pi}(\theta)],
$$

(5)

where $\overline{\pi}(\theta) := \pi(\theta)/q(\theta)$ and $Z := \int \pi(\theta) d\theta$ is the normalisation constant.

Consider a sequence, $\{\theta_i\}_{i=1}^{\infty}$ with marginal density $q$. Provided that a strong law of large numbers (SLLN) applies, setting $h(\theta) = 1$ in the above equation implies that $\hat{Z}_N :=$

1http://mc-stan.org
\[ \frac{1}{N} \sum_{i=1}^{N} \pi(\theta_i) \rightarrow Z, \text{ almost surely, and hence, almost surely,} \]
\[ \hat{E}_N(h) := \frac{1}{N} \sum_{i=1}^{N} w_N(\theta_i) h(\theta_i) \rightarrow \mathbb{E}_{\pi/Z}[h(\theta)]. \]

where \( w_N(\theta) := \pi(\theta)/\hat{Z}_N \). In Section 4 we will use importance sampling to estimate expectations with respect to the combined posterior distribution.

3 A Gaussian-process approximation to posterior

3.1 Gaussian-process approximations to the subposteriors

Parallelising the MCMC procedure over \( C \) computing nodes results in \( C \) subposteriors \( \{\pi_c(\vartheta)\}_{c=1}^{C} \). From each subposterior, \( c \), where the MCMC algorithm has been iterated \( J \) times, we have \( \mathcal{D}_c = \{\vartheta_{c,j}, \log \pi_c(\vartheta_{c,j})\}_{j=1}^{J} \), where each pair consists of a sample from the Markov chain with its associated log-subposterior, up to some fixed additive constant. We wish to convert this limited information on a finite set of points to information about \( \log \pi_c \) over the whole support of \( \vartheta \). We therefore treat the whole log-subposterior (up to the same additive constant), \( \mathcal{L}_c(\vartheta) \), as random with a Gaussian-process prior distribution:

\[ \mathcal{L}_c(\vartheta) \sim \mathcal{GP}(m(\vartheta), K(\vartheta, \vartheta')), \]

where \( m : \vartheta \rightarrow \mathbb{R} \) and \( K : \vartheta \times \vartheta \rightarrow \mathbb{R} \) are, respectively, the mean and covariance functions. For computational convenience, we further assume that the log-subposteriors are independent of each other. We model \( \log \pi_c(\vartheta) \) rather than \( \pi_c(\vartheta) \), so that our approximation to the overall log-posterior will be a sum of Gaussian-process (Section 3.2); modelling the log-posterior also avoids the need for non-negativity constraints when fitting the GP\(^2\).

The mean function and covariance function are chosen by the user. A mean function of zero, \( m(\vartheta) = 0 \), would be inappropriate in this setting as our prior must be the logarithm of a probability density function up to a finite additive constant. We ensure that \( \int \exp \{ \mathcal{L}_c(\vartheta) \} \, d\vartheta < \infty \) almost surely through a quadratic mean function of the form

\[ m(\vartheta) = \beta_0 + \vartheta_1^\top \beta_1 + \text{diag}(\vartheta^\top V^{-1} \vartheta) \beta_2, \quad \beta_2 < 0. \]

Here \( V \) is the empirical covariance of \( \vartheta \) obtained from the MCMC sample and \( \beta_i, \ (i = 0, 1, 2) \) are unknown constants.

The covariance function \( K(\cdot, \cdot) \), determines the smoothness of the log-subposterior, which we shall assume is continuous with respect to \( \vartheta \). A popular choice is the squared-exponential

\(^2\text{Gaussian-process approximations are fitted using the Julia package https://github.com/STOR-i/GaussianProcesses.jl}\)
function (e.g. Rasmussen and Williams, 2006)
\[ K(\vartheta, \vartheta') = \omega^2 \exp \left( -\frac{1}{2} (\vartheta - \vartheta')^T \Lambda^{-1} (\vartheta - \vartheta') \right), \]  
(7)

where \( \Lambda \) is a diagonal matrix and \( \omega \) are hyperparameters. In this paper we analytically marginalise \( \beta_0 \) and \( \beta_1 \) (O’Hagan, 1978) and estimate \( \beta_2 \) and the kernel hyperparameters through maximum likelihood (details given in Chapter 5 of Rasmussen and Williams (2006)). We have found that our choice of mean and covariance function work well in practice, however, alternative functions can be applied and may be more appropriate depending on the characteristics of the log-subposterior.

Given the choice of prior, \( \mathcal{D}_c \) are observations of this Gaussian-process, leading to a posterior distribution,
\[ p(\ell_c(\vartheta)|\mathcal{D}_c) \propto p(\mathcal{D}_c|\ell_c(\vartheta))p(\ell_c(\vartheta)). \]  
(8)

Define \( \mathcal{L}_c(\vartheta_{1:J}) := \{ \mathcal{L}_c(\vartheta_1), \ldots, \mathcal{L}_c(\vartheta_J) \} \) and, for some parameter, or parameter vector, \( \theta := \theta_{1:N} := (\vartheta_1, \ldots, \vartheta_N) \), define \( \mathcal{L}_c(\theta_{1:N}) := \{ \mathcal{L}_c(\vartheta_1), \ldots, \mathcal{L}_c(\vartheta_N) \} \). We require the posterior distribution of \( \mathcal{L}_c(\theta_{1:N})|\mathcal{D}_c \), that is the conditional distribution \( \mathcal{L}_c(\theta_{1:N})|\{ \mathcal{L}_c(\vartheta_{1:J}) = \ell_c(\vartheta_{1:J}) \} \). Since the joint distribution between \( \mathcal{L}_c(\vartheta_{1:J}) \) and \( \mathcal{L}_c(\theta_{1:N}) \) is multivariate Gaussian, the conditional, \( \mathcal{L}_c(\theta_{1:N})|\{ \mathcal{L}_c(\vartheta_{1:J}) = \ell_c(\vartheta_{1:J}) \} \) is also multivariate Gaussian,
\[ \mathcal{L}_c(\theta_{1:N})|\mathcal{D}_c \sim \mathcal{N}(\mu_c(\theta_{1:N}), \Sigma_c(\theta_{1:N})) \]  
(9)

with,
\[ \mu_c(\theta_{1:N}) = m_c(\theta_{1:N}) + K_{*}^\top \tilde{K}^{-1}(\mathcal{L}_c(\vartheta_{1:J}) - m_c(\vartheta_{1:J})) \]
\[ \Sigma_c(\theta_{1:N}) = K_{**} - K_{*}^\top \tilde{K}^{-1} K_{*}, \]  
(10)

and where \( \tilde{K} = K(\vartheta_{1:J}, \vartheta_{1:J}) \), \( K_{*} = K(\theta_{1:N}, \vartheta_{1:J}) \) and \( K_{**} = K(\theta_{1:N}, \theta_{1:N}) \).

The posterior distribution for the GP, \( \mathcal{L}_c(\theta_{1:N})|\mathcal{D}_c \), is a random approximation of the log-subposterior surface.

### 3.2 Merging the subposteriors

Our next goal is to approximate the full posterior \( \pi(\vartheta) \propto \prod_{c=1}^{C} \pi_c(\vartheta) \) by merging the subposteriors together. The approximation of each of the \( C \) subposteriors as independent Gaussian-processes, \( \mathcal{L}_c(\theta) \sim \mathcal{GP}(\cdot, \cdot) \ (c = 1, \ldots, C) \) leads directly to the approximation of the full log-posterior (up to an additive constant) as the sum of \( C \) Gaussian-processes,
\[ \mathcal{L}(\theta)|\mathcal{D} := \sum_{c=1}^{C} \mathcal{L}_c(\theta)|\mathcal{D}_c = \mathcal{GP} \left( \sum_{c=1}^{C} \mu_c(\theta), \sum_{c=1}^{C} \Sigma_c(\theta) \right) . \]  
(11)
Our assumption that the Gaussian-processes representing the log-subposteriors \( \{L_c\}_{c=1}^{C} \) are independent \textit{a priori} was made for computational convenience. This may not be true in practice since gross deviations from the quadratic prior mean, \( m_c(\vartheta) \), such as any skewness, may be repeated across subposteriors. However, \textit{a posteriori} these gross deviations should be accounted for through the posterior mean \( \mu_c(\vartheta) \). Variability in the original partitioning of the data into batches, and variability in the sample points, \( \vartheta_{1:J} \), across batches will both contribute to the more subtle variations of the GPs about their individual posterior means, so that the posterior correlation should be much smaller than the prior correlation.

### 3.3 Illustration

The Gaussian-process subposterior, provides estimates of the uncertainty in the log-subposterior at points, \( \theta \), where the log-subposterior has not been evaluated. This contrasts with current approaches (e.g. Scott et al., 2013; Neiswanger et al., 2013; Wang and Dunson, 2013) to approximate the subposterior which give no measure of uncertainty. This is illustrated below and used in Sections 4.3 and 5.3 to gauge the uncertainty in our estimates of posterior expectations.

To illustrate how the sum of Gaussian-processes is used to approximate \( L(\theta) \) we sample \( n = 20,000 \) data points from a Normal(\( \theta,1.5^2 \)) distribution with \( \theta = 2 \) and evenly split the data across \( C = 2 \) processors. Independent HMC algorithms are run on each subposterior targeting the posterior for \( \theta \). A Gaussian-process approximation, as shown in Figure 1, is fitted to each of the subposteriors, where the blue line is the GP mean and the blue band gives a 95% confidence interval of the uncertainty in the approximation at unobserved regions of the parameter space. Using (11), the approximation to the full posterior is given by summing the means and covariances of the Gaussian-process approximations to the subposteriors.
4 Approximating the full posterior

We now detail three methods for approximating posterior expectations, all of which utilise our Gaussian-process approximation to the full posterior density.

4.1 The expected posterior density

Here we approximate the full posterior density (up to an unknown normalising constant) by its expectation under the Gaussian-process approximation:

\[
\hat{\pi}_E(\theta) \propto \mathbb{E}[\exp \mathcal{L}(\theta)] = \exp \left\{ \sum_{c=1}^{C} \mu_c(\theta) + \frac{1}{2} \Sigma_c(\theta) \right\},
\]

(12)

using the properties of the log-Normal distribution. If the individual GPs provide a good approximation to the individual log-subposteriors, then \(\mathbb{E}[\mathcal{L}(\theta)]\) will be a good approximation to the full log-posterior.

The HMC algorithm then provides an efficient mechanism for obtaining an approximate sample, \(\{\theta_i\}_{i=1}^{N}\) from \(\hat{\pi}_E\). Evaluating the GP approximation at each iteration of this MCMC algorithm is significantly faster than evaluating the true full posterior, \(\pi(\theta)\), directly. As is apparent from the leapfrog dynamics, HMC requires the gradient of \(\log \pi_E\), and here the tractability of our approximation is invaluable, since

\[
\nabla \log \hat{\pi}_E = \sum_{c=1}^{C} \frac{\partial}{\partial \theta} \mu_c(\theta) + \frac{1}{2} \frac{\partial}{\partial \theta} \Sigma_c(\theta)
\]

\[
= \sum_{c=1}^{C} \frac{\partial}{\partial \theta} m(\theta) + \frac{\partial K^\top_c}{\partial \theta} K_c^{-1}(\ell_c(\vartheta_{1:J}) - m(\vartheta_{1:J})) + \frac{1}{2} \frac{\partial}{\partial \theta} K_{s,s} - \frac{\partial}{\partial \theta} K^\top_{s} K^{-1}. \]

Given a sufficiently large sample from \(\hat{\pi}_E\), approximations of posterior expectations can be highly accurate if the individual GPs provide a good approximation to the log-subposteriors. Moreover, the approximation \(\hat{\pi}_E(\theta)\) to the full posterior can be further improved by using importance sampling on the true posterior.

4.2 Distributed importance sampling

Unlike the proposal, \(q\), in Section 2.2, samples generated from the HMC algorithm represent an approximate, correlated sample from an approximation to the true posterior, instead of exact, independent samples from an approximation. Nonetheless, we may still correct for inaccuracies in \(\hat{\pi}_E\) using importance sampling while spreading the computational burden across all \(C\) cores.

The full sample from the HMC algorithm targeting \(\hat{\pi}_E\), \(\{\theta_i\}_{i=1}^{N}\), is sent to each of the \(C\) cores. Each of the \(C\) cores then evaluates the true subposterior at each \(\theta_i\). A single core then combines the subposterior densities for each \(\theta_i\) to provide the full true posterior density:
Algorithm 1 Distributed Importance Sampler

**Input:** Proposal distribution \( q(\theta_{1:N}) \), which has marginal \( q_i(\theta_i) \) identical for all \( \theta_i, \ i = 1, \ldots, n \).
- Sample \( \theta_{1:N} \sim q(\cdot) \).
- For \( c = 1, \ldots, C \) evaluate each subposterior \( \pi_c(\theta_{1:N}) \).
- Set \( \pi(\theta_i) = \prod_{c=1}^{C} \pi_c(\theta_i) \).
- Weight the samples, \( w_i = \frac{\pi(\theta_i)}{q(\theta_i)} \).

**Output:** Weighted sample \( \{w_i, \theta_i\}_{i=1}^{N} \) approximating \( \pi(\theta) \).

\( \pi(\theta_i) = \prod_{c=1}^{C} \pi_c(\theta_i) \). To be clear, each sub-posterior is evaluated at the *same* set of \( \theta \) values, allowing them to be combined exactly. In contrast, the original HMC runs, performed on each individual subposterior, created a different set of \( \theta \) values for each subposterior so that a straightforward combination was not possible.

Each value from the sample, \( \theta_i \), is then associated with an unnormalised weight, \( \bar{w}(\theta_i) = \pi(\theta_i)/\tilde{\pi}_E(\theta_i) \). Defining \( \tilde{Z}_N \) and \( w_N(\theta) \) as in Section 2.2 provides an approximation \( \tilde{E}_N(h) \) to \( E_\pi[h(\theta)] \) as defined in (6).

Since the unknown normalising constants for both \( \pi \) and \( \tilde{\pi}_E \) appear in both the numerator and the denominator of this expression, they are not needed. Almost sure convergence of \( \tilde{E}_N(h) \) to \( E_\pi[h(\theta)] \) as the HMC sample size, \( N \to \infty \) relies on the strong law of large numbers (SLLN) for Markov chains (e.g. Tierney, 1996, Theorem 4.3). In addition, if desired, an unweighted approximate sample from \( \pi \) may be obtained by resampling \( \theta_i \) with a probability proportional \( w_i \).

We expect our HMC importance proposal to be especially efficient, since it mimics the true posterior. However, other proposal distributions based on competing algorithms for merging subposteriors (e.g. Scott et al., 2013; Neiswanger et al., 2013; Wang and Dunson, 2013) can be used instead; these are compared in Section 5. Algorithm 1 describes this general distributed importance sampler.

### 4.3 Gaussian-process importance sampler (GP-IS)

Finally, we present an importance sampler that uses the full posterior distribution of \( \mathcal{L} \), the GP approximation to the full unnormalised log-posterior conditional on \( \{\vartheta_{c,j}, \pi_c(\vartheta_{c,j})\}_{c=1,j=1}^{C,J} \). Compared with the importance sampler in Section 4.2, the set of points \( \{\theta_i\}_{i=1}^{N} \) is generated from a simple proposal distribution, rather than the HMC algorithm applied to \( \tilde{\pi}_E \). Moreover, given the set of points \( \{\theta_i\}_{i=1}^{N} \), the computationally-expensive evaluation of each subposterior at this set of values is replaced with repeated, but relatively cheap sampling of realisations of \( \mathcal{L} \) at these points. For a fixed number of GP training points, \( J \), estimates of posterior expectations are no-longer asymptotically exact in \( N \), however estimates of the uncertainty in these estimates are also supplied.

As in Sections 4.2 and 2.2 we are interested in \( I_h := E_\pi[h(\theta)] = \frac{1}{N} \int \pi(\theta)h(\theta)d\theta \). Here we
consider approximating this with
\[ I_h(\ell) := \frac{1}{Z(\ell)} \int \exp \{ \ell(\theta) \} h(\theta) d\theta, \]
where \( \ell \) is a realisation of \( \mathcal{L} \) from the distribution in (11) and \( Z(\ell) := \int \exp \{ \ell(\theta) \} d\theta \) is the associated normalisation constant.

First consider the hypothetical scenario where it is possible to store \( \ell \) completely and evaluate \( I_h(\ell) \). A set of \( M \) realisations of \( \mathcal{L} \), \( \{ \ell_m \}_{m=1}^M \) would lead to \( M \) associated estimates of \( I_h \), \( \{ I_h(\ell_m) \}_{m=1}^M \), which would approximate the posterior distribution of \( I_h \) under (11). The mean of these would then target the posterior expectation,
\[ I^E_h := \mathbb{E} \left[ \frac{1}{Z(\mathcal{L})} \int h(\theta) \exp (\mathcal{L}(\theta)) d\theta \right]. \]

As an alternative, robust, point estimate, the median of \( \{ I_h(\ell_m) \}_{m=1}^M \) would target the posterior median. Other posterior summaries for \( I_h \), such as a 95\% credible interval, could also be estimated from the sample.

Unfortunately it is not possible to store the infinite-dimensional object, \( \ell \); and even if it were, for moderate dimensions, numerical evaluation of \( I_h(\ell) \) would be computationally infeasible. Instead we use importance sampling. Consider a proposal distribution \( q(\theta) \) that approximately mimics the true posterior distribution, \( \pi(\theta) \) and sample \( N \) independent points from it: \( \theta_{1:N} := (\theta_1, \ldots, \theta_N) \). For each \( m \in \{1, \ldots, M\} \) we then sample the finite-dimensional object \( (\ell_m(\theta_1), \ldots, \ell_m(\theta_N)) \) from the joint distribution of the GP in (11). For each such realisation we then construct an approximation to the normalisation constant and to \( I_h(\ell) \):
\[ \hat{Z}(\ell_m) := \frac{1}{N} \sum_{i=1}^N w(\theta_i; \ell_m) \quad \text{and} \quad \hat{I}_h(\ell_m) := \frac{1}{N \hat{Z}(\ell_m)} \sum_{i=1}^N w(\theta_i; \ell_m) h(\theta_i), \]
where \( w(\theta; \ell) := \exp \{ \ell(\theta) \}/q(\theta) \). The set \( \{ \hat{I}_h(\ell_m) \}_{m=1}^M \) is then used in place of \( \{ I_h(\ell_m) \}_{m=1}^M \) for posterior inference on \( I_h \).

For the specific case of \( I^E_h \) a simplified expression for the approximation may be derived:
\[ \hat{I}_E^h = \frac{1}{N} \sum_{i=1}^N w_i h(\theta_i), \quad \text{where} \quad w_i := \frac{1}{Mq(\theta_i)} \sum_{m=1}^M \frac{\exp \{ \ell_m(\theta_i) \}}{\hat{Z}(\ell_m)}. \quad (13) \]
Algorithm 2 creates point estimates based upon this.

The proposal density \( q(\theta_i) \) should be a good approximation to the posterior density. To create a computationally cheap proposal, and with a similar motivation to the consensus Monte Carlo approximation (Scott et al., 2013), we make \( q(\theta_i) \) a multivariate Student-t distribution on 5 degrees of freedom with mean and variance matching those of the Gaussian posterior that would arise given the mean and variance of each subposterior and if each sub-posterior were
Algorithm 2 GP Importance Sampler

**Input:** GP approximation \( \mathcal{L}(\theta) \) and proposal distribution \( q(\cdot) \).
- Sample \( \theta_{1:N} \sim q(\cdot) \) iid.
- Sample \( m = 1, \ldots, M \) realisations of the GP approximation. to the log-posterior \( \mathcal{L}_m(\theta_{1:N}) \) in (11).
- Weight the samples according to (13).

**Output:** Weighted sample \( \left\{ w_i, \theta_i \right\}_{i=1}^N \), approximately from the marginal of \( \{ \mathcal{L}, \pi(\theta|\mathcal{L}) \} \).

Gaussian, Alternatively, it would be possible to use the output from the HMC algorithm of Section 4.1 in an analogous manner to the way it is used in Section 4.2.

Many aspects of our importance sampler can, if necessary, be parallelised: in particular, calculating \( \mu_c(\theta_{1:N}) \) and \( \Sigma_c(\theta_{1:N}) \), and then sampling \( \ell_1, \ldots, \ell_m \) and obtaining the sample \( \{ \hat{I}_h(\ell_m) \}_{m=1}^M \).

### 4.4 Computational cost

We briefly review some of the notation in the paper as a point of reference for this section.

- \( n := \# \) data points, \( y \).
- \( C := \# \) processing cores (i.e. number of batches).
- \( J := \# \) MCMC samples drawn from \( \pi_c(\theta) \), \( c = 1, \ldots, C \); for simplicity, we assume \( J \) samples are drawn from each subposterior.
- \( N := \# \) samples drawn from the approximation to the merged posterior \( \hat{\pi}_E(\theta) \), or, for GP-IS, from the Student-t proposal.

The overall computational cost of applying the methods in Sections 4.1 and 4.2 to create an approximate (weighted) sample from the full posterior can be summarised in three (four) steps:

- **Run MCMC on each subposterior** (see Section 2). This step is common to all parallel MCMC algorithms (e.g. Scott et al., 2013; Neiswanger et al., 2013; Wang et al., 2015) and has a cost of \( O(Jn/C) \).

- **Fit GP to each subposterior** (see Section 3). Fitting a Gaussian-process to each subposterior has a cost of \( O(J^3) \) due to the inversion of the \( J \times J \) matrix \( \tilde{K} \). One of the drawbacks of Gaussian-processes is the computational cost. Faster, approximate Gaussian-processes, referred to as sparse GPs (e.g Csató and Opper, 2002; Seeger et al., 2003; Quiñonero-candela et al., 2005; Snelson and Ghahramani, 2006) can be used to reduce the computational cost. \(^3\) In

\(^3\)Generally speaking, sparse GPs introduce \( p \) inducing points as training point locations \( \vartheta_P, i = 1, \ldots, p \), to fit the GP. The computational complexity of such an approach is reduced if \( P \ll J \) to give an overall cost of \( O(P^2 J) \).
this paper we apply the simpler speed-up technique of first thinning the subposterior Markov chain; for example, using only every twentieth sample. The thinned Markov chain has the same stationary distribution as the full chain, but the autocorrelation is reduced and, more importantly for us, the sample contains fewer points. Secondly we remove the duplicate samples from the subposterior as they provide no extra information for the GP approximation, and cause the kernel matrix $\tilde{K}$ to become singular. Fitting $C$ independent GPs to each of the subposteriors is embarrassingly parallel as the MCMC output from each subposterior is stored on a separate core.

- **Perform HMC on** $\hat{\pi}_E$ (see Section 4.1). Each iteration of the HMC algorithm requires an evaluation of $\mu_c$ and $\Sigma_c$ from (10) with $N = 1$, and multiple evaluations of the gradient terms given in Section 4.1. Since $\tilde{K}^{-1}$ has already been calculated, the total cost over all $N$ iterations of the HMC algorithm is $O(NJ^2)$. The cost of this step is equivalent to competing algorithms including (Neiswanger et al., 2013; Wang and Dunson, 2013), which also use an MCMC-type step to sample from the approximation to the posterior.

- **Re-weight GP samples for DIS** (see Section 4.2). Our distributed importance sampler weights the approximate samples from $\hat{\pi}_E$ according to the true posterior and requires the subposteriors to be re-evaluated at each point in the GP-HMC sample. More generally, a sample from any sensible proposal distribution could be used. This has a cost of $O(Nn/C)$.

- **GP-IS** (see Section 4.3). Taking the proposal, $q$, to be the Student-t distribution described at the end of Section 4.3, creating a sample of size $N$ has a cost of $O(N)$. For the sample, $\theta_{1:N}$, creation of each $\Sigma_c(\theta_{1:N})$ $(c = 1, \ldots, C)$ in parallel is $O(N^2J + NJ^2)$. Cholesky decomposition of their sum, $\Sigma$, is $O(N^3)$; however a spectral decomposition truncated to the largest $T$ eigenvalues is $O(N^2T)$. The $M$ multiplications $\Sigma^{1/2}z$ (where $z$ is a vector of independent standard Gaussians) that generate realisations of $L$ can be spread between processors, leading to a cost of $O(MN^2/C)$ (or $O(MNT/C)$ for a truncated spectral decomposition). GP-IS may, therefore, be preferable to DIS when $MN << n$ (or $MT << n$).

### 5 Experiments

In this section we compare our Gaussian-process algorithms for aggregating the subposteriors against several competing algorithms:

- **Consensus Monte Carlo** (Scott et al., 2013), where samples are weighted and aggregated.

- **Nonparametric density product** Neiswanger et al. (2013), where each subposterior is approximated using kernel density estimation.
• **Semiparametric density product**\(^4\) Neiswanger et al. (2013), similar to the nonparametric method, but where subposteriors are approximated semiparametrically as in Hjort and Glad (1995).

• **Weierstrass rejection sampler**\(^5\) (Wang and Dunson, 2013), where the nonparametric density estimates are passed through a Weierstrass transform to give the merged posterior.

We consider four interesting examples: a Bernoulli model with rare events which leads to a skewed posterior, a mixture of Laplace distributions which only becomes identifiable with a large amount of data, and two logistic regression models for large data sets. These examples highlight some of the challenges faced by merging non-Gaussian subposteriors and the computational efficiency of large-scale Bayesian inference.

Our Gaussian-process approximation method is implemented using \(J = 100\) samples from the thinned chain for each subposterior to fit the GPs for the Bernoulli and multimodal examples; for the logistic regression examples \(J = 500\). Both for our methods and for comparator methods, \(N = 5000\) samples from each merged posterior are created. To ensure a fair comparison, the sample from each \(\hat{\pi}_E\) that is used both directly and in our DIS algorithm is the unthinned output from the HMC run. The Student-t proposals for the Gaussian-process importance sampler are iid.

Weighted samples from DIS and GP-IS are converted to unweighted samples by resampling with replacement, where the probability of choosing a given \(\theta\) is proportional to its weight.

For each of the models studied in this section we denote the true parameter values by \(\theta^*\) (when known). We obtain an accurate estimate of the true posterior from a long MCMC run, thinned to a size of \(N\), with samples denoted \(\theta^f\) and the true posterior mean and variance \(m_f\) and \(V_f\), respectively. Samples from the approximation are denoted \(\theta^a\), and their mean and variance are \(m_a\) and \(V_a\). We use the following metrics to compare the competing methods:

- **Mahalanobis distance**, \(D_{Mah.} = \sqrt{(m_a - m_f)^\top V_f^{-1} (m_a - m_f)}\).

- **Kullback-Leibler divergence** for the Bernoulli and mixture example is calculated using a nearest neighbour search\(^6\) and for the logistic regression example, approximate multivariate Gaussian Kullback-Leibler divergence (see Wang and Dunson (2013) for details) between the true posterior \(\pi\) and aggregated posterior \(\hat{\pi}\) is calculated as

\[
D_{KL}(\hat{\pi}(\theta)||\pi(\theta)) = \frac{1}{2} (\text{tr}(V_f^{-1} V_a) + (m_f - m_a)^\top V_f^{-1} (m_f - m_a) - d - \log(|V_a|/|V_f|)).
\]

- **Posterior concentration ratio**, \(\rho = \sqrt{\sum_{i=1}^N ||\theta_i^a - \theta^*||_2^2/\sum_{i=1}^N ||\theta_i^f - \theta^*||_2^2}\) (Wang et al., 2015), which gives a measure for the posterior spread around the true value \(\theta^*\) (\(\rho=1\) being ideal).

\(^4\)Implemented using the parallelMCMCcombine R package
\(^5\)Implemented using the authors R package https://github.com/wwrechard/weierstrass
\(^6\)Implemented using the FNN R package
Table 1: Accuracy of various aggregation methods on a Bernoulli model with simulated rare event data. Kullback-Leibler results are reported as \((\times 10^1)\) and \(D_{Mah.}\) as \((\times 10^5)\). Results in brackets represent the DIS step.

| Algorithm                | \(D_{Mah.}\)  | \(D_{KL}(π||\hat{π})\) | \(D_{KL}(\hat{π}||π)\) | \(ρ\)   | \(η\)  |
|-------------------------|---------------|--------------------------|---------------------------|---------|--------|
| Consensus               | 1.28 (0.99)   | 1.82 (0.66)              | 1.32 (0.46)               | 1.27 (1.02) | 0.74 (0.06) |
| Nonparametric           | 1.78 (0.91)   | 7.45 (0.89)              | 10.71 (0.83)              | 0.94 (0.91) | 2.12 (0.14) |
| Semiparametric          | 1.27 (0.94)   | 2.04 (0.79)              | 2.82 (0.97)               | 0.98 (0.82) | 0.25 (0.16) |
| Weierstrass             | 1.53 (1.05)   | 2.61 (1.14)              | 1.95 (1.16)               | 1.05 (0.84) | 0.37 (0.28) |
| GP-HMC sampler          | 1.07 (0.85)   | 0.77 (0.80)              | 0.70 (0.76)               | 1.05 (1.01) | 0.12 (0.11) |
| GP-IS sampler           | 1.03          | 0.72                     | 0.55                      | 1.00     | 0.13   |

- Mean absolute skew deviation, \(η = \frac{1}{d} \sum_{i=1}^{d} |\gamma_i - \gamma_i^f|\), where \(i\) is the component, \(\gamma_i = \mathbb{E}[\{(θ_i - m_i)/\sqrt{V_{ii}}\}^3]\) is the third standardised moment, and the superscripts \(f\) and \(a\) denote empirical approximations obtained from the samples obtained using the true posterior and the approximation, respectively.

5.1 Rare Bernoulli events

The consensus Monte Carlo approach of Scott et al. (2013) is the optimal algorithm for merging the subposteriors when each subposterior is Gaussian. A popular example where this algorithm struggles is the Bernoulli model with rare events, where the subposterior distributions are skewed (e.g. Wang et al., 2015; Wang and Dunson, 2013; Scott et al., 2013). We sample \(n = 10,000\) Bernoulli random variables \(y_i \sim Bern(\vartheta)\) and split the data across \(C = 10\) processors. We set \(\vartheta = C/n\) so that the probability of observing an event is rare. In fact, each subset only contains 1 success on average. A \(Beta(2, 2)\) prior distribution is assumed for \(\vartheta\).

Figure 2 and Table 1 give the results of merging the subposteriors for the various algorithms. Both GP-HMC and GP-IS samplers produce good approximations to the posterior. All of the competing algorithms can reasonably identify the mode of the posterior, but do not adequately fit the tail of the distribution. The consensus algorithm gives a reasonable approximation to the body of the density but struggles to capture the posterior skew. The nonparametric method appears to perform the worst in this setting; this could be improved by hand-tuning the bandwidth parameter. However, doing so assumes that the full posterior is known, which is not the case in practice.

We can generate samples from the full posterior using the distributed importance sampler (Alg. 1), where samples from each of the aggregation methods can be used as a proposal. Figure 2 (right panel) shows that using the DIS improves the accuracy of all of the competing methods. This improvement is most noticeable for the consensus and nonparametric approximations. Ultimately, the overall accuracy of the approximation to the full posterior will depend on the quality of the proposal distribution.
For this experiment and \( \beta = 0 \), then the mixture components are non-identifiable, however, by setting \( \beta_1 = 1.01 \) and \( \beta_2 = 0.99 \) the parameter of interest, \( \theta \) can be identified from a sufficiently large dataset. For this experiment \( \theta = 0.05 \) and the data are split equally over \( C = 25 \) processors. The scale parameters, \( \beta_1 \) and \( \beta_2 \) are fixed at their true values and a \( \mathcal{N}(0, 1) \) prior is assumed for \( \theta \).

The left panel of Figure 3 depicts the full posterior and subposterior densities. The full pos-

### 5.2 Multimodal subposteriors

We create a concrete data scenario that could lead to a set of multimodal sub-posteriors similar to the artificial, perturbed multimodal sub-posteriors used in Wang et al. (2015). The example is a toy representation of a general situation where one or more parameters of interest are poorly identified, but as the size of the dataset increases towards ‘large’ \( n \), the parameters start to become identifiable. The subposteriors are multimodal but the full posterior is unimodal (see Figure 3, left panel).

We sample \( n = 1,000,000 \) observations from a mixture of two Laplace distributions

\[
\frac{1}{2} \frac{1}{\beta_1} \exp \left( -\frac{|y - \theta|}{\beta_1} \right) + \frac{1}{2} \frac{1}{\beta_2} \exp \left( -\frac{|y + \theta|}{\beta_2} \right).
\]

If \( \beta_1 = \beta_2 \) then the mixture components are non-identifiable, however, by setting \( \beta_1 = 1.01 \) and \( \beta_2 = 0.99 \) the parameter of interest, \( \theta \) can be identified from a sufficiently large dataset. Results in brackets represent the DIS step.
Figure 3: Left: The first ten subposteriors (orange dashed lines) plotted alongside the full posterior (red solid line). Right: Approximations of the full posterior given by each aggregation method.

The true posterior reveals, approximately, the true value for $\theta$, whereas $\theta$ is poorly identified by each of the subposteriors. The multimodality of the subposteriors results in a poor posterior approximation from the consensus Monte Carlo algorithm (right panel) as each subposterior is assumed to be approximately Gaussian. On the other hand, most of the nonparametric methods are able to capture the approximate shape of the posterior, but fail to correctly detect the posterior mode. Table 2 shows that the DIS step can slightly degrade the quality of the approximation if the proposal (e.g. semiparametric) under-represents the tail behaviour of the true posterior. As shown in Figure 3 (right panel), the GP-HMC and GP-IS samplers produce good approximations to the full posterior and unlike the nonparametric methods, the GP approximations concentrate around the posterior mode (see $\rho$ in Table 2).

5.3 Logistic regression

**Synthetic data set** We use a synthetic data set on internet click rate behaviour where one of the covariates is highly predictive, but rarely observed. The data set has $n = 10,000$, with 5 covariates and is identical to that in Section 4.3 of Scott et al. (2013). Partitioning the data across 10 machines means that only a few machines carry an observation for the highly predictive covariate, leading to some skewed subposterior distributions.

The consensus Monte Carlo algorithm struggles in this scenario because, even though the marginal of the full posterior for the coefficient of the rarely observed covariate is approximately Gaussian, the subposteriors are not. We also compare the standard set-up of the merging algorithms against their DIS equivalents and find that DIS improves the accuracy of the approximation. The improvement brought about by DIS applied to our GP-HMC sampler is relatively small because $\hat{\pi}_E$ is already an accurate approximation to the posterior. Nonetheless, as can be seen with the consensus approximation, the DIS step can lead to a significant improvement.
across million instances and discard the mass attribute in our model fit. The data is divided equally producing collisions from the background source. The full data set contains 10.5 millions

challenge here is to accurately classify the collisions of exotic particles by separating the particle-dimension of the parameter space increases.

methods, and as discussed in Wang and Dunson (2013), these methods begin to struggle as the

methods begin to struggle on this example. This is partly due to the difficulty of tuning these

if the approximation produced by the merging algorithm alone is poor. The nonparametric

methods begin to struggle as the dimension of the parameter space increases.

Table 3: Accuracy of various aggregation methods on logistic regression model with simulated data set. Results in brackets represent the DIS step. Kullback-Leibler results are reported as \((\times 10^2)\).

| Algorithm         | \(D_{Mah.}\) | \(D_{KL}(\pi\|\hat{\pi})\) | \(D_{KL}(\hat{\pi}\|\pi)\) | \(\rho\) | \(\eta\) |
|-------------------|--------------|-----------------------------|-----------------------------|---------|---------|
| Consensus         | 2.35 (2.23)  | 2.50 (0.52)                 | 2.95 (0.51)                 | 0.97 (1.00) | 0.08 (0.04) |
| Nonparametric     | 2.89 (1.76)  | 35.60 (37.10)               | 65.89 (30.87)               | 2.4 (2.34) | 0.09 (0.08) |
| Semiparametric    | 2.22 (1.79)  | 13.76 (14.27)               | 16.59 (37.88)               | 1.99 (2.04) | 0.09 (0.08) |
| Weierstrass       | 2.33 (1.71)  | 1.40 (1.06)                 | 1.54 (1.25)                 | 1.39 (1.16) | 0.08 (0.08) |
| GP-HMC sampler    | 2.23 (2.22)  | 0.53 (0.49)                 | 0.51 (0.47)                 | 0.99 (1.00) | 0.07 (0.06) |
| GP-IS sampler     | 2.25         | 0.75                        | 0.76                        | 0.98     | 0.05     |

Table 4: Accuracy of various aggregation methods on the logistic regression model with the HEPMASS dataset. Results in brackets represent the DIS step.

| Algorithm         | \(D_{Mah.}\) | \(D_{KL}(\pi\|\hat{\pi})\) | \(D_{KL}(\hat{\pi}\|\pi)\) | \(\eta\) |
|-------------------|--------------|-----------------------------|-----------------------------|---------|
| Consensus         | 6.13 (5.80)  | 13.65 (13.22)               | 13.36 (13.45)               | 0.31 (0.25) |
| Nonparametric     | 10.73 (5.59) | 17.23 (15.76)               | 33.32 (14.89)               | 0.34 (0.31) |
| Semiparametric    | 5.36 (5.14)  | 15.93 (15.01)               | 13.71 (14.23)               | 0.32 (0.30) |
| Weierstrass       | 6.48 (4.62)  | 14.75 (13.98)               | 13.61 (13.49)               | 0.32 (0.29) |
| GP-HMC sampler    | 5.90 (5.27)  | 13.28 (13.57)               | 13.81 (13.27)               | 0.23 (0.22) |
| GP-IS sampler     | 6.07         | 13.56                       | 13.98                       | 0.27     |

if the approximation produced by the merging algorithm alone is poor. The nonparametric

methods begin to struggle on this example. This is partly due to the difficulty of tuning these

methods, and as discussed in Wang and Dunson (2013), these methods begin to struggle as the dimension of the parameter space increases.

**Real data set** We conduct parallel MCMC experiments on the Hepmass\(^7\) data set. The challenge here is to accurately classify the collisions of exotic particles by separating the particle-producing collisions from the background source. The full data set contains 10.5 millions instances with 28 attributes representing particle features. In our experiments we use the first million instances and discard the mass attribute in our model fit. The data is divided equally across \(C = 20\) machines.

The results from Table 4 show that all methods approximate the full posterior with more or less the same level of accuracy, with the exception of the nonparametric method. As discussed in Neiswanger et al. (2013), nonparametric methods scale poorly with dimension with the Weierstrass and semiparametric algorithms performing better than the simple nonparametric method. The posterior concentration ratio is not reported as this is close to one for all methods. The subposteriors are approximately Gaussian and as a result all methods, including the consensus Monte Carlo algorithm, produce accurate approximations to the full posterior. As a result, and with the exception of the nonparametric method, applying the DIS step does

\(^7\)https://archive.ics.uci.edu/ml/datasets/HEPMASS
Table 5: Expectation and variance of $\theta_1$ and $\theta_{17}$ from the logistic regression model with the HEPMASS dataset. Mean, Median and quantile estimates of the quantities are calculated from 500 samples from the GP-IS sampler (i.e. $M=500$).

As described in Section 4.3, the GP-IS sampler draws multiple realisations from the posterior distribution of the GP approximation to the posterior. Each of these realisations provides an estimate of the expectation of interest, their centre (mean or median) provides a point estimate and their spread (2.5% and 97.5% quantiles) provide a measure of the uncertainty. In Table 5 we estimate the posterior mean and variance of two parameters and compare these estimates against the truth, as calculated from an MCMC run on the full posterior. Sampling $M = 500$ realisations from GP we report the mean, median and 95% CI for estimates of the mean and variance and find that these results are consistent with the truth.

6 Discussion

Aggregating subposteriors generated through parallel, independent MCMC simulations, to form the full posterior distribution is challenging. Currently, available methods either produce a Gaussian approximation to the posterior, or utilise nonparametric estimators which are difficult to tune and do not scale well to high-dimensional settings. In this paper we have presented an alternative approach to this problem by directly modelling the log-density of the subposteriors. Using Gaussian-process priors we were able to employ a fully Bayesian strategy towards approximating the full posterior, and unlike competing methods, we were able to account for the uncertainty in the approximation.

Compared to the nonparametric methods, fitting the Gaussian-processes is straightforward using a mixture of marginalisation and maximum likelihood techniques for the hyperparameters. The main drawback of using Gaussian-process approximations is the computational cost. We have reduced the computational cost by, for each subposterior sample, thinning the Markov chain and removing duplicate MCMC samples prior to fitting the GP. We have shown that using only a small number of samples from the subposterior we can accurately approximate the full posterior. Furthermore, the computationally intensive step of fitting the individual GPs to the subposteriors is automatically parallelised, as the subposteriors are independent by definition and the GPs are independent by design.

The algorithms we propose scale well with the number of data points $n$, but fitting a GP when the dimension, $d$, of $\theta$ is high can be computationally expensive as the number of input
points required to produce an accurate approximation grows exponentially with $d$. An extension to this work would be to employ sparse GP approximations to reduce the computational expense for high-dimensional problems.

Acknowledgement

We would like to thank Prof. Paul Fearnhead for helpful discussions. The first author gratefully acknowledges partial support from the EPSRC funded EP/H023151/1 STOR-i centre for doctoral training.

References

Andrieu, C. and Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373. 4

Bardenet, R., Doucet, A., and Holmes, C. (2014). Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. *Proceedings of The 31st International Conference on Machine Learning*, (4):405–413. 2

Beskos, A., Pillai, N., Roberts, G., Sanz-Serna, J.-M., and Stuart, A. (2013). Optimal tuning of the hybrid Monte Carlo algorithm. *Bernoulli*, 19(5A):1501–1534. 5

Chen, T., Fox, E. B., and Guestrin, C. (2014). Stochastic Gradient Hamiltonian Monte Carlo. In *Proceedings of the International Conference on Machine Learning*, volume 32, pages 1–23, Beijing. 2

Csató, L. and Opper, M. (2002). Sparse Online Gaussian Processes. *Neural computation*, 14(2):641–669. 12

Geweke, J. (1989). Bayesian inference in econometric models using Monte Carlo integration. *Econometrica: Journal of the Econometric Society*, 57(6):1317–1339. 5

Girolami, M. and Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214. 5

Hjort, N. L. and Glad, I. K. (1995). Nonparametric Density Estimation with a Parametric Start. *The Annals of Statistics*, 23(3):882–904. 14

Hoffman, M. and Gelman, A. (2014). The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(2008):30. 5
Huang, Z. and Gelman, A. (2005). Sampling for Bayesian Computation with Large Datasets.

Liu, H., Lafferty, J., and Wasserman, L. (2007). Sparse Nonparametric Density Estimation in High Dimensions Using the Rodeo. *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS-07)*, pages 283–290.

Maclaurin, D. and Adams, R. P. (2014). Firefly Monte Carlo: Exact MCMC with Subsets of Data. *Arxiv preprint arXiv:1403.5693*, (Mcmc):1–13.

Neal, R. M. (2010). MCMC Using Hamiltonian Dynamics. In *Handbook of Markov Chain Monte Carlo (Chapman & Hall/CRC Handbooks of Modern Statistical Methods)*, pages 113–162.

Neiswanger, W., Wang, C., and Xing, E. (2013). Asymptotically Exact, Embarrassingly Parallel MCMC. *Arxiv preprint arXiv:1311.4780*.

O’Hagan, A. (1978). Curve Fitting and Optimal Design for Prediction. *Journal of the Royal Statistical Society, Series B*, 40(1):1–42.

Quiñonero-candela, J., Rasmussen, C. E., and Herbrich, R. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1935–1959.

Quiroz, M., Villani, M., and Kohn, R. (2014). Speeding up MCMC by efficient data subsampling. *Arxiv preprint arXiv:1404:4178v1*, pages 1–37.

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Robert, C. and Casella, G. (1999). *Monte Carlo Statistical Methods*. Springer-Verlag, New York, Inc.

Roberts, G. O., Gelman, A., and Gilks, W. (1997). Weak Convergence and Optimal Scaling of the Random Walk Metropolis Algorithms. *The Annals of Applied Probability*, 7(1):110–120.

Roberts, G. O. and Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268.

Scott, S., Blocker, A., Bonassi, F., Chapman, H., George, E., and McCulloch, R. (2013). Bayes and big data: the consensus Monte Carlo algorithm. In *Bayes 250*. 2, 8, 10, 11, 12, 13, 15, 17
Seeger, M., Williams, C., and Lawrence, N. (2003). Fast forward selection to speed up sparse Gaussian process regression. In *Ninth International Workshop on Artificial Intelligence and Statistics. Society for Artificial Intelligence and Statistics*, 2003., page 2003. 12

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. *Neural Information Processing Systems* 18. 12

Teh, Y. W., Thiery, A. H., and Vollemer, S. J. (2014). Consistency and fluctuations for stochastic gradient Langevin dynamics. *Arxiv preprint arXiv:1409.0578v1*. 2

Tierney, L. (1996). Introduction to general state-space markov chain theory. In Gilks, W., Richardson, S., and Spiegelhalter, D., editors, *Markov Chain Monte Carlo in Practice*, pages 59–74. Chapman and Hall, New York. 10

Wang, X. and Dunson, D. B. (2013). Parallelizing MCMC via Weierstrass Sampler. *Arxiv preprint arXiv:1312.4605*. 8, 10, 13, 14, 15, 18

Wang, X., Dunson, D. B., and Heller, K. A. (2015). Parallelizing MCMC with Random Partition Trees. In *NIPS 2015*. 12, 14, 15, 16

Wang, Z., Mohamed, S., and de Freitas, N. (2013). Adaptive Hamiltonian and Riemann Manifold Monte Carlo Samplers. *Proceedings of the 30th International Conference on Machine Learning*. 5

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*. 2

Wilkinson, D. J. (2005). Parallel Bayesian Computation. In *Handbook of Parallel Computing and Statistics*, pages 477–509. 2
Algorithm 3 Hamiltonian Monte Carlo

Input: Initial parameters \( \vartheta^{(0)} \), step-size \( \epsilon \) and trajectory length \( L \).

for \( t = 1 \) to \( T \) do
    Sample momentum \( \varphi \sim \mathcal{N}(0, M) \)
    Set \( \vartheta_1 \leftarrow \vartheta^{t-1} \) and \( \varphi_1 \leftarrow \varphi \)
    for \( l = 1 \) to \( L \) do
        \( \varphi_{l+\frac{1}{2}} \leftarrow \varphi_l + \frac{\epsilon}{2} \nabla \log \pi(\vartheta_l) \)
        \( \vartheta_{l+1} \leftarrow \vartheta_l + \epsilon M^{-1} \varphi_{l+\frac{1}{2}} \)
        \( \varphi_{l+1} \leftarrow \varphi_{l+\frac{1}{2}} + \frac{\epsilon}{2} \nabla \log \pi(\vartheta_{l+1}) \)
    end for
    Set \( \vartheta \leftarrow \vartheta_{L+1} \) and \( \varphi \leftarrow \varphi_{L+1} \)
    With probability \( \min \left\{ 1, \frac{\exp\left( \log \pi(\vartheta) - \frac{1}{2} \varphi M^{-1} \varphi \right)}{\exp\left( \log \pi(\vartheta^{t-1}) - \frac{1}{2} \varphi^{t-1} M^{-1} \varphi^{t-1} \right)} \right\} \)
    set \( \vartheta^t \leftarrow \vartheta \) and \( \varphi^t \leftarrow \varphi \)
end for

Output: Samples \( \{\vartheta^t\}_{t=1}^T \) from \( \pi(\vartheta) \).

A The HMC algorithm