Control Variates for Stochastic Gradient MCMC

Jack Baker[1*] Paul Fearnhead[1] Emily B. Fox[2] Christopher Nemeth[1]

[1] STOR-i Centre for Doctoral Training, Department of Mathematics and Statistics, Lancaster University, Lancaster, UK
[2] Department of Statistics, University of Washington, Seattle, WA

Abstract

It is well known that Markov chain Monte Carlo (MCMC) methods scale poorly with dataset size. We compare the performance of two classes of methods which aim to solve this issue: stochastic gradient MCMC (SGMCMC), and divide and conquer methods. We find an SGMCMC method, stochastic gradient Langevin dynamics (SGLD) to be the most robust in these comparisons. This method makes use of a noisy estimate of the gradient of the log posterior, which significantly reduces the per iteration computational cost of the algorithm. We analyse the algorithm over different dataset sizes and show, despite the per iteration saving, the computational cost is still proportional to the dataset size. We use control variates, a method to reduce the variance in Monte Carlo estimates, to reduce this computational cost to $O(1)$. Next we show that a different control variate technique, known as zero variance control variates can be applied to SGMCMC algorithms for free. This post-processing step improves the inference of the algorithm by reducing the variance of the MCMC output. Zero variance control variates rely on the gradient of the log posterior; we explore how the variance reduction is affected by replacing this with the noisy gradient estimate calculated by SGMCMC.

Keywords: Stochastic gradient MCMC; Langevin dynamics; scalable MCMC; control variates; computational cost; big data

1 Introduction

Markov chain Monte Carlo (MCMC), one of the most popular methods for Bayesian inference, scales poorly with dataset size. This is because standard methods require the whole dataset to be evaluated at each iteration of the MCMC algorithm. Recent innovations in MCMC methodology have produced scalable algorithms that are amenable to large datasets and these new algorithms have been successfully applied to a range of state-of-the-art machine learning problems (e.g. Patterson and Teh, 2013; Wenzhe Li, 2016).

These new scalable algorithms can mostly be divided into two groups: *divide and conquer* and *minibatch*. The divide and conquer methods deliver computational speed-up by splitting the data across separate computer core and running independent parallel MCMC algorithms which are later combined. By comparison, minibatch methods require a single core, but achieve speed-up over standard MCMC by using only a subset of the data at each iteration.

Suppose we have data $\mathbf{x}$, then divide and conquer methods split the data into S disjoint subsets, call these $\mathbf{x}_{B_s}$. MCMC is then run in parallel, targeting each *subposterior* $p(\theta|\mathbf{x}_{B_s})$. The computational speed-up compared to running MCMC on all the data is approximately proportional to the number of available cores. The challenge with using such methods lies in recombining the MCMC output for each subposterior to approximate the full posterior (Scott et al., 2016; Neiswanger et al., 2014; Wang and Dunson, 2013; Srivastava et al., 2015; Nemeth and Sherlock; Scott, 2017).

*email: j.baker1@lancaster.ac.uk

Rather than parallelizing, the computational cost of MCMC can be reduced by leveraging minibatches, where only a subset of the data is used at each MCMC iteration. Some methods alter the Metropolis-Hastings acceptance step to account for the fact that only a subsample of the data is used to calculate the posterior density (Korattikara et al., 2014; Maclaurin and Adams, 2014; Bardenet et al., 2016). Other methods combine efficient proposals based on discretized dynamics that uses gradient information. They reduce the computational cost by using noisy gradient estimates calculated using minibatches of data. They bypass the acceptance step by choosing a small stepsize (Welling and Teh, 2011; Chen et al., 2014; Ding et al., 2014; Dubey et al., 2016). These latter methods, known as stochastic gradient MCMC (SGMCMC), have become popular and are the focus of this paper.

Both the divide and conquer and SGMCMC methods produce algorithms that are computationally more efficient than standard MCMC. However this comes at the expense of them being approximate: recombine steps of the divide and conquer methods are often based on strong assumptions; while the introduction of sub-sampling within most SGMCMC methods produces an MCMC algorithm that no longer has the true posterior as its invariant distribution. Often it is difficult in practice to quantify the error each method introduces. The first contribution of this paper is a systematic comparison of methods that attempts to tease out how different methods perform depending on different features of an application. The conclusion of this simulation study is that divide and conquer methods work well in situations either where the subposteriors are unimodal and have roughly elliptical contours; though there has been promising work which focuses on the 1d case (Minsker et al., 2014). However outside these cases they can be unreliable. As such we generally recommend the use of sub-sampling based methods. These methods are the focus for the rest of the paper. In particular we focus on SGLD, which we found to have the best combination of robustness and simplicity of tuning.

In this paper we show that the computational cost of SGLD is proportional to the dataset size. We do this by comparing SGLD to a coupled true Langevin diffusion that SGLD is trying to approximate. We make this comparison over varied dataset size, and show that in order for the squared error of SGLD to reach a desired level compared with the underlying diffusion, the minibatch size n has to be set proportional to the size of the data. It follows that the computational cost of SGLD is still $O(N)$.

This motivates us to improve the computational cost of SGLD, and we do this by using control variates (Ripley, 2009). We show that these ideas can be easily applied to reduce the Monte Carlo variance of the gradient estimate in stochastic gradient MCMC algorithms. When we compare SGLD with this gradient estimate to the underlying diffusion we find that we can set a constant minibatch size n_0, over any dataset size N, and the squared error of SGLD will still reach the desired level of accuracy. Thus the computational cost has been reduced from $O(N)$ to $O(1)$. We show similar results for the computational cost of SAGA (Dubey et al., 2016), another SGLD algorithm that leverages control variates.

We demonstrate efficiency improvements empirically on a variety of complex models from the machine learning literature. SAGA uses a previous state in the chain in order to produce its control variate, while we use a stochastic optimization step which replaces the burn-in of the chain. We show in our experiments that while SAGA works very well on simpler examples, when it does not have a good starting point on more complex examples, burn-in can be prohibitively slow. The algorithm also appears to get stuck in local modes. This is where our method has an advantage, as the stochastic optimization step is often faster than SGLD burn-in, and does not get stuck. On the other hand, SAGA has the advantage of the control variate calculation being performed within the SGLD algorithm itself.

Not only can control variates be used to speed up stochastic gradient MCMC by enabling smaller minibatches to be used; we show that they can be used to improve the inferences made from the MCMC output. In particular, we can use post-processing control variates (Mira et al., 2013; Papamarkou et al., 2014; Friel et al., 2016) to produce MCMC samples with a reduced variance. The post-processing methods rely on the MCMC output as well as gradient information. Since stochastic gradient MCMC methods already compute estimates of the gradient, we explore replacing the true gradient in the post-processing step with these free estimates. We also show theoretically how this affects the variance reduction factor; and empirically demonstrate the variance reduction that can be achieved from using these post-processing methods.

2 Preliminaries

In this section we outline in more detail the divide and conquer and stochastic gradient MCMC methods that are investigated in the comparisons. The stochastic gradient methods, especially SGLD, were found to be most effective and simple to use in our comparisons. We provide a more detailed explanation of SGLD, as we focus on this method in our control variate analysis.

2.1 Divide-and-conquer methods

Suppose we have data $\mathbf{x}$ realised from a model with density $p(.)$ that depends on some parameter vector $\theta \in \mathbb{R}^d$. The divide and conquer methods make the assumption that subsets of the data are conditionally independent wrt θ. We call these subsets $\mathbf{x}_{B_s}$. The posterior, up to a multiplicative constant, can then be written as

$$p(\theta|\mathbf{x}) \propto \prod_{s=1}^{S} [p(\theta)]^{1/S} p(\mathbf{x}_{B_s}|\theta) = \prod_{s=1}^{S} p(\theta|\mathbf{x}_{B_s}), \tag{1}$$

where we refer to $p(\theta|\mathbf{x}_{B_s})$ as a *subposterior*. So if we know the density of the subposteriors, then their product will be the full posterior. In reality, combining these subposteriors to recover the full posterior is challenging. We look at three methods to recombine the full posterior from the subposterior samples: Consensus Monte Carlo (Scott et al., 2016), KDEMC (Neiswanger et al., 2014) and the Weierstrass sampler (Wang and Dunson, 2013).

2.1.1 Consensus Monte Carlo

The simplest way to recombine the samples from the subposteriors is to approximate each subposterior as a Gaussian distribution. The samples can be used to estimate the mean and variance of each of the subposteriors. Then, conditionally on these estimates we can analytically calculate a Gaussian approximation to the full posterior. This idea was first proposed by Neiswanger et al. (2014). The motivation is that as N gets large the Bernstein-von Mises theorem states that the posterior will be approximately Gaussian (Le Cam, 2012).

The consensus Monte Carlo algorithm of Scott et al. (2016) aims to improve on this. It works by approximating the full posterior as a weighted average of the subposterior samples. The idea behind consensus Monte Carlo is that, if the subposteriors were Gaussian then this method of combining samples would give us draws from the true posterior. Moreover, Scott et al. (2016) argue that consensus Monte Carlo is more robust to deviations from Gaussian subposteriors, as the approximate full posterior that it samples from can inherent some of the properties, such as skewness or heavy tails.

Suppose we have an MCMC sample $\theta_{s1}, \ldots, \theta_{sM}$ from $p(\theta|\mathbf{x}_{B_s})$ for $s = 1, \ldots, S$, Scott et al. (2016) propose estimating the full MCMC chain, call this $\hat{\theta}_i$, as a weighted average of the subposterior samples

$$\hat{\theta}_i = \left(\sum_{s=1}^{S} W_s \right)^{-1} \sum_{s=1}^{S} W_s \theta_{si}, \tag{2}$$

where $W_s \in \mathbb{R}^{d \times d}$ is a weight matrix for subposterior s. Scott et al. (2016) suggest letting $W_s = \Sigma_s^{-1}$, where Σ_s is the subposterior covariance matrix. The consensus approach is very scalable: only a weighted average needs to be calculated after the embarassingly parallel MCMC chains are run. However, the Normality assumptions are questionable, so in the comparison that follows, we test how well the algorithm performs in practice.

2.1.2 KDEMC

Neiswanger et al. (2014) suggest estimating the full posterior by applying kernel density estimation to each subposterior, $\hat{p}(\theta|\mathbf{x}_{B_s})$. Then by (1) we can approximate the full posterior by $\hat{p}(\theta|\mathbf{x}) = \prod_{s=1}^{S} \hat{p}(\theta|\mathbf{x}_{B_s})$.

If Gaussian kernels are used in the approximation, then $\hat{p}(\theta|\mathbf{x})$ becomes a product of Gaussian mixtures. This product can be expanded to give another Gaussian mixture with $O(SM)$ components, where M is the number of iterations of the MCMC chain that are stored, and S is the number of subposteriors. Neiswanger et al. (2014) suggest sampling from this Gaussian mixture using MCMC. We refer to this algorithm as KDEMC. The number of mixture components increases dramatically with the number of subsets and subposterior samples. This means KDEMC can be computationally expensive and inefficient, but the algorithm should target more complex posteriors better. Neiswanger et al. (2014) also suggest a similar method based on *semiparametric* density estimation.

2.1.3 Weierstrass

The Weierstrass method (Wang and Dunson, 2013) is similar to KDEMC, but uses a Weierstrass transform to approximate the subposterior densities rather than a kernel density estimate. Using the Weierstrass approximation rather than a kernel density is associated with a number of better properties, including an improvement when subposteriors do not overlap, and better scalings with dimensionality. The properties of the Weierstrass transform allow the approximation to be sampled using a rejection sampler rather than a Gibbs sampler, which is more efficient.

2.2 Stochastic gradient MCMC

Many MCMC algorithms are based upon discrete-time approximations to continuous-time dynamics, such as the Langevin diffusion, that are known to have the posterior as their invariant distribution. The approximate discrete-time dynamics are then used as a proposal distribution within a Metropolis-Hastings algorithm. The accept-reject step within such an algorithm corrects for any errors in the discrete-time dynamics. Examples of such an approach include the Metropolis-adjusted Langevin algorithm (MALA; see e.g. Roberts and Rosenthal (1998)) and Hamiltonian Monte Carlo (HMC; see Neal (2010)).

2.3 Stochastic gradient Langevin dynamics

SGLD, first introduced by Welling and Teh (2011), is a minibatch version of the Metropolis-adjusted Langevin algorithm. Following the stochastic approximation methods of Robbins and Monro (1951), the SGLD algorithm creates an approximation of the true gradient of the log-posterior by leveraging minibatches of data.

Suppose we aim to make inference on a set of parameters θ, with data $\mathbf{x} = \{x_i\}_{i=1}^N$. We denote the probability density of x_i as $p(x_i|\theta)$ and assign a prior density $p(\theta)$. The resulting posterior is then $p(\theta|\mathbf{x}) \propto p(\theta)\prod_{i=1}^N p(x_i|\theta)$. A Langevin diffusion of this posterior is given by

$$d\theta_t = \frac{1}{2}\nabla \log p(\theta_t|\mathbf{x})dt + dB_t, \tag{3}$$

where B_t is a Wiener process. This equation targets the posterior exactly, but in practice we need to discretize the dynamics to simulate from it, introducing error. A bottleneck for this simulation is that $\nabla \log p(\theta|\mathbf{x})$ is an $O(N)$ operation. So to get around this, Welling and Teh (2011) replace the log-posterior gradient with the following unbiased estimate

$$\nabla \widehat{\log p(\theta_t}|\mathbf{x}) := \nabla \log p(\theta_t) + \frac{N}{n}\sum_{i\in S_t} \nabla \log p(x_i|\theta), \tag{4}$$

for some subsample S_t of $\{1,\ldots,N\}$, with $|S_t| = n$. A single update of SGLD is then

$$\theta_{t+1} = \theta_t + \frac{\epsilon_t}{2}\nabla \widehat{\log p(\theta_t}|\mathbf{x}) + \zeta_t, \tag{5}$$

where $\zeta_t \sim N(0, \epsilon_t)$.

MALA uses a Metropolis-Hastings accept-reject step to correct for the discretization of the Langevin process. Welling and Teh (2011) bypass this acceptance step, which would require calculating $p(\theta|\mathbf{x})$ using the full dataset, and instead use an adaptive rather than fixed stepsize, where $\epsilon_t \to 0$ as $t \to \infty$. The motivation is that the noise in the gradient estimate disappears faster than the process noise, so eventually, the algorithm will sample the posterior approximately. In practice, we found the algorithm does not mix well when the stepsize is decreased to zero, so generally use a fixed small stepsize, as suggested by Vollmer et al. (2016).

2.4 Stochastic gradient Hamiltonian Monte Carlo

The stochastic gradient Hamiltonian Monte Carlo algorithm (SGHMC) (Chen et al., 2014) is similar to SGLD, but instead approximates Hamiltonian dynamics, which underlies Hamiltonian Monte Carlo (HMC) (Neal, 2010). SGHMC augments the parameter space with momentum variables, ν. The algorithm samples approximately from a joint distribution $p(\theta, \nu|\mathbf{x})$, whose marginal distribution for θ is the posterior of interest. The SGHMC algorithms performs the following updates at each iteration

$$\theta_{t+1} = \nu_t, \quad \nu_{t+1} = -\eta \nabla \widehat{\log p(\theta_{t+1}|\mathbf{x})} - \alpha \nu_t + \zeta_t,$$

where $\zeta_t \sim N(0, 2(\alpha - \hat{\beta}_t)\eta)$; η and α are parameters that need to be tuned and $\hat{\beta}_t$ is proportional to an estimate of the Fisher information matrix. Often the dynamics are simulated for a trajectory of length L before the state is stored, at which point ν is resampled. This means the computational cost is therefore L times larger than SGLD.

3 Comparison of MCMC methods for big data

In this Section we aim to provide an extensive comparison of some popular MCMC methods for big data. As far as we know, there has been limited comparison between stochastic gradient and divide and conquer methods, and we aim to bridge this gap in the following comparisons. We use simple examples that focus on important scenarios, and hope to build intuition for where methods should be used. The particular scenarios we focus on are: heavy tails, multimodality and complex geometry. We end the section with a summary of each method based on the comparisons.

We compare each method's performance by measuring the KL divergence between the approximate sample and a HMC sample, taken to be the truth, using the R package *FNN* (Li et al., 2013). The HMC sample is simulated using STAN (Carpenter et al., 2017). The KL divergence is measured over 10 different runs of the algorithm and plotted as boxplots, or in the case of the dimensionality algorithm a line plot with error bars. Contour plots for one simulation are also provided to help develop the reader's intuition. The only method which does not require tuning parameters is the Consensus method, which is a huge advantage as this can take a lot of time. The other methods are tuned by minimizing the KL divergence measure which we use to make the boxplots. The Weierstrass algorithm is implemented using the authors' R package (Wang and Dunson, 2013).

3.1 Heavy tails: multivariate-t

We infer the location θ from data $\mathbf{x}$ simulated from a bivariate-t distribution with known scale Σ and degrees of freedom ν. The density of $\mathbf{x}$ is given by

$$p(\mathbf{x}|\theta) \propto \left[1 + \frac{1}{\nu}(\mathbf{x} - \theta)^T \Sigma^{-1} (\mathbf{x} - \theta)\right]^{-(\nu+2)/2},$$

where we assume an uninformative prior on θ. In order to test the algorithms we use a relatively small dataset size of 800. The number of subposteriors used in the divide and conquer methods is 20. We use a minibatch size of 50 for the stochastic gradient MCMC methods.

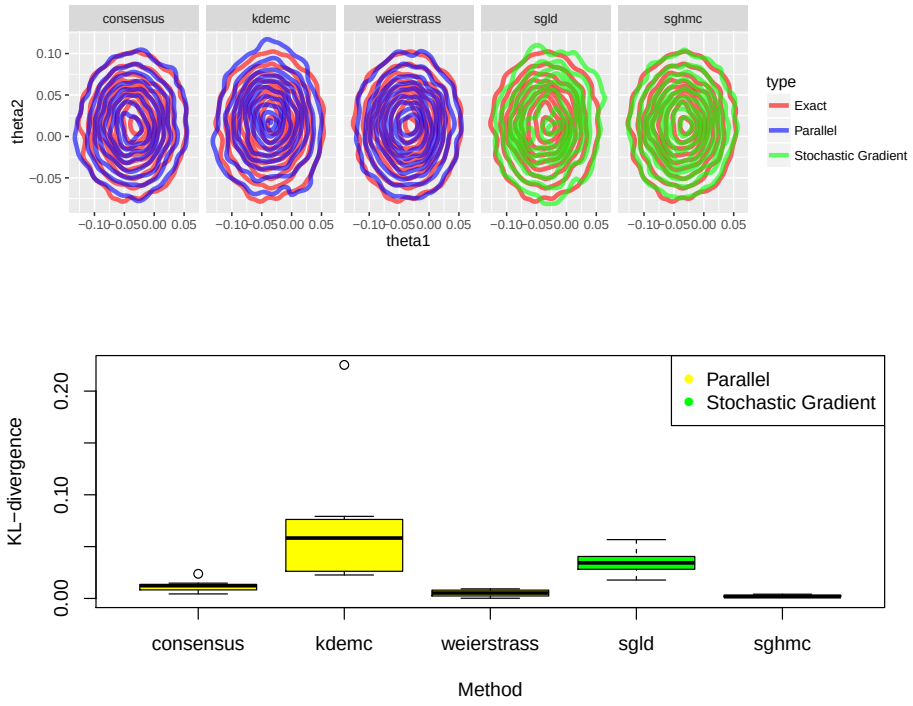

Figure 1: Comparison of method performance for multivariate-t distribution. Contour plots show empirical densities. Box plots show KL-divergence from the truth.

Figure 1 gives an illustrative comparison of the methods discussed in Section 2. The results show that all methods are equally equipped to sufficiently explore this heavy tailed posterior. The KDEMC and SGLD algorithms are the slowest to converge to the posterior. It has been shown in Teh et al. (2016) that the convergence rate of SGLD is $O(T^{-\frac{1}{3}})$, and therefore slower than the standard Monte Carlo rate of $O(T^{-\frac{1}{2}})$. In this scenario SGHMC performs the best in terms of minimizing KL divergence, closely followed by the consensus Monte Carlo algorithm and the Weierstrass sampler. The Weierstrass sampler does a good job of improving the convergence speed of KDEMC. There is an additional advantage in using the consensus Monte Carlo as it does not require tuning, so is arguably the best choice for this problem.

3.2 Multiple modes: Gaussian mixture

We consider a multimodal target where we infer the locations θ_1, θ_2 from data $\mathbf{x}$ simulated from a bimodal, bivariate Gaussian mixture. We assume the mixture has known common scale Σ and unknown allocation probabilities α_1, α_2. We marginalise out the allocation probabilities to obtain the following density of $\mathbf{x}$

$$p(\mathbf{x}|\theta_1, \theta_2) \propto \mathcal{N}(\mathbf{x}|\theta_1, \Sigma) + \mathcal{N}(\mathbf{x}|\theta_2, \Sigma),$$

where $\mathcal{N}(\mathbf{x}|\theta, \Sigma)$ denotes a Gaussian density with mean θ and variance Σ. We assume the priors on θ_i are independent Gaussians with mean 0 and a large variance. We use a larger dataset size of 10000. The number of subposteriors used in the divide and conquer methods is 20. We use a minibatch size of 50 for the stochastic gradient MCMC methods.

Results given in Figure 2 show that the consensus algorithm performs poorly in this setting. The simple weighted average scheme (2) leads to a unimodal posterior approximation which does not account for the bimodality, suggesting that the posterior mass lies between the subposterior modes. KDEMC offers an improvement over the Consensus algorithm with improved posterior coverage, but still does not capture the

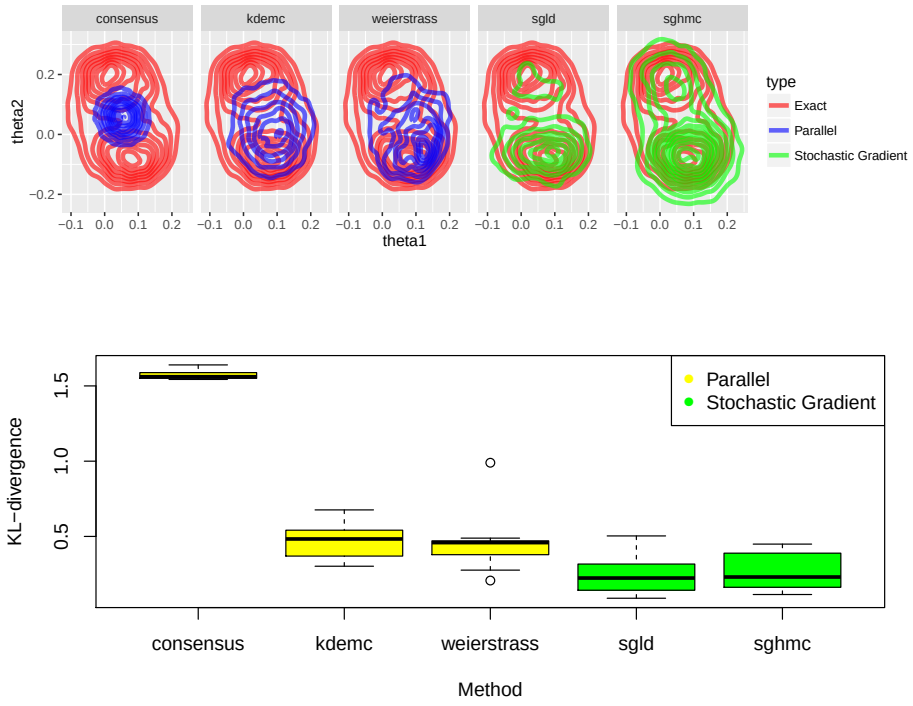

Figure 2: Comparison of method performance for Gaussian mixture. Contour plots show empirical densities. Box plots show KL-divergence from the truth.

bimodality. From investigating the subposteriors it appears that this is a result of the kernel bandwidth being smaller than the width of the posterior, leading to density estimates for each subposterior which tail off rapidly outside of the region where subposterior samples lie. Therefore, the full posterior is only non-zero in the region of intersection of all subposterior samples. The Weierstrass method seems to encounter similar issues to KDEMC, and its performance is not much better.

The stochastic gradient methods perform better than the divide and conquer approaches, and are able to explore both modes. Given a good starting point, SGHMC performs particularly well. When the starting point is further from the posterior mass, the algorithm performs worse than SGLD. The algorithm is also sensitive to the choice of $\hat{\beta}$, as in the experimental setup in Chen et al. (2014), we set $\hat{\beta} = 0$ for all the experiments. As noted in the previous example (Section 3.1), SGLD can be slow to converge and may require further iterations. Contrasting this with the computational cost of SGHMC, which is L times greater than SGLD, we suggest using SGLD for this problem and running the algorithm for more iterations.

3.3 Complex geometry: warped Gaussian

We consider a target with complex geometry known as the warped Gaussian. In this case, locations θ_1 and θ_2 are inferred from data x with density

$$p(x|\theta_1, \theta_2) = \mathcal{N}(x|\theta_1 + \theta_2^2, \sigma_x),$$

where σ_x is a known scale parameter. We assume the prior for each θ_i are independent with density $p(\theta_i) = \mathcal{N}(\theta_i|0, \sigma_\theta)$, where σ_θ is some known scale parameter. In order to test the algorithms we use a relatively small dataset size of 800. The number of subposteriors used in the divide and conquer methods is 20. We use a minibatch size of 50 for the stochastic gradient MCMC methods.

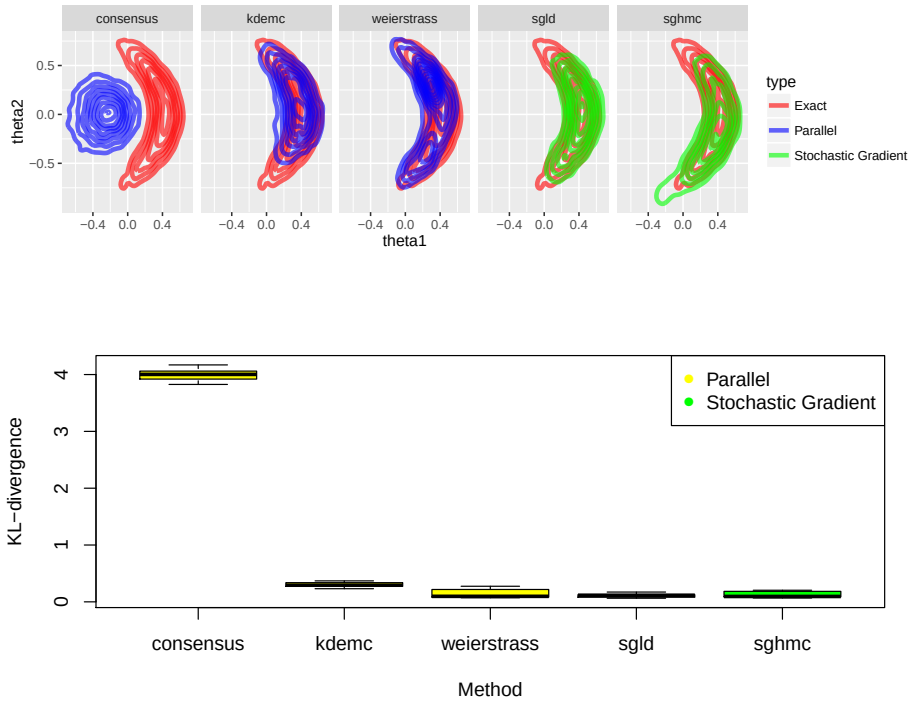

Figure 3: Comparison of method performance for warped Gaussian. Contour plots show empirical densities. Box plots show KL-divergence from the truth.

The results given in Figure 3 show that again the consensus algorithm struggles to approximate the full posterior. The consensus approach uses an average of the subposterior samples, re-scaled by their covariance. One way of understanding why the consensus performs poorly in this example is to consider the situation where there are only two subposteriors, each with approximately the correct warped Gaussian shape and location as the full posterior. Averaging samples from each subposterior would lead to some samples located in the the lower tail of subposterior one being averaged with samples from the upper tail of subposterior two, thus producing an approximation to the full posterior which lies in the centre, as shown in Figure 3. The KDEMC works reasonably well on this example, but underestimates the tails for the same reason as discussed for the mixture example (Section 3.2). The Weierstrass shows some improvement over KDEMC, though does not perform as well as the stochastic gradient methods.

Finally, the stochastic gradient methods perform better than the divide and conquer algorithms and once again SGHMC is more sensitive to the starting point than SGLD.

3.4 Dimensionality: multivariate Gaussian

The examples considered so far have been in low dimensional parameter spaces. In this section we explore how these big data MCMC algorithms scale with increasing the dimension of the posterior. We consider the posterior for θ given $\mathbf{x}$, where $\mathbf{x}$ follows a multivariate Gaussian with known scale Σ. We assume an uninformative prior for θ. In order to test the algorithms we again use a relatively small dataset size of 800. The number of subposteriors used in the divide and conquer methods is 20. We use a minibatch size of 50 for the stochastic gradient MCMC methods.

Figure 4 gives the KL divergence between the full posterior and the approximate posterior resulting from each of the considered algorithms. Kernel density methods are well known to scale poorly with dimension and this is shown here. The Consensus algorithm performs particularly well. This is unsurprising as the

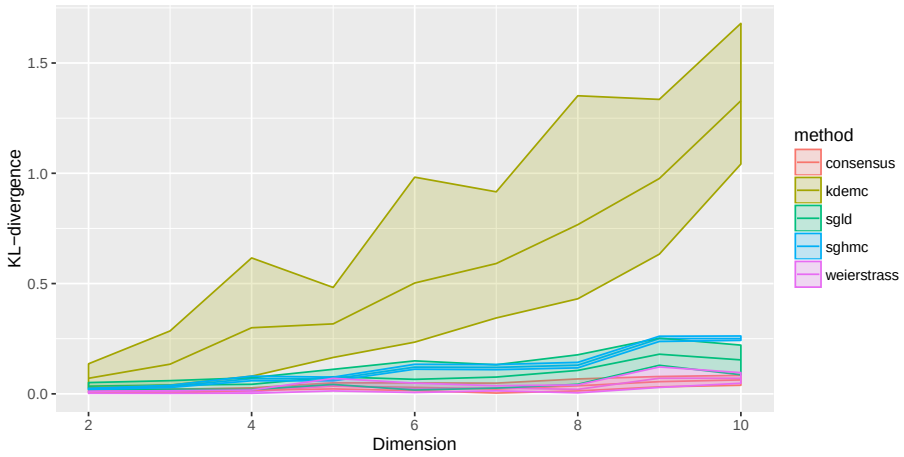

Figure 4: Comparison of method performance for Gaussian. Plot of KL-divergence against dimension for each method.

consensus algorithm is exact when each subposterior is Normally distributed. The Weierstrass method scales much better with dimensionality than its KDEMC counterpart, this is due to its sequential rejection sampling procedure, which ensures that error accumulates linearly in dimensionality, as opposed to exponentially in dimensionality as is the case for KDEMC. Both minibatch methods work well, but the trend in Figure 4 implies that these algorithms may lose some performance in significantly larger posterior spaces. This can, to some extent, be mitigated by pre-conditioning the gradient estimates, for example, using stochastic gradient Riemannian Langevin dynamics (Patterson and Teh, 2013).

3.5 Discussion

When considering unimodal posteriors which do not exhibit complex geometry, the consensus algorithm is arguably preferred; as the algorithm does not require any tuning and scales well in high-dimensional parameter spaces. KDEMC is a natural extension to the consensus algorithm, which merges the subposterior densities rather than the subposterior samples. We found through experimentation that, as with the consensus algorithm, the KDEMC approach tends to underestimate the tails of the full posterior density, which is particularly an issue when the subposterior densities do not overlap. The KDEMC algorithm also scales very poorly with dimension. The Weierstrass sampler, which extends ideas of the KDEMC algorithm, fixes many of these issues. The algorithm scales well with dimensionality; copes better with posteriors that do not overlap and converges faster than KDEMC. However the algorithm still struggles with multimodality, and is not quite up to the standard of the SGMCMC methods when it comes to more complex geometry. The algorithm requires tuning of an acceptance step, but the results are not too sensitive to this choice.

Stochastic gradient methods were found to be robust to the geometry of the posterior as well as handling multimodality. A major disadvantage of these algorithms is how sensitive they are to the choice of stepsize, though some work has been done to improve this (Giles et al., 2016; Gorham et al., 2016). We found in general the extra computational cost of SGHMC did not lead to vastly improved performance over SGLD. This could be due to sensitivity to the Fisher information estimate, or to the choice of starting point.

It was hard to compare computational cost directly between the two methods, but each algorithm was run for the same number of MCMC iterations. For most of the comparisons, the dataset size was chosen to be about 800, so quite small, designed to test the methods. In these cases the computational cost between SGLD and the divide and conquer methods is similar since the divide and conquer methods have a per iteration cost of 40 (as the number of batches is 20), and the SGLD algorithm has a per iteration cost of 50 since that was its minibatch size. So the comparisons certainly demonstrate the slow convergence rate

of SGLD for simple examples. But SGLD demonstrated strength in the more complex geometry of the warped Gaussian. In the Gaussian mixture example, SGLD has the lowest computational cost by far, but still performs the best. It's worth noting that the combining step adds to the computational cost, and for KDEMC this is a particularly slow process.

On the other hand, the trajectory L for SGHMC was chosen to be 3, so that the momentum parameter was not refreshed at every step. This means the per iteration computational cost was about 3 times higher than the other methods. For simple examples, SGHMC did not warrant this extra tuning and computational cost, as it did not perform much better than consensus Monte Carlo. For more complex examples, the method again did not warrant the extra cost over SGLD.

In the mixture example, to test SGLD more, the dataset size is 10000, with minibatch size 50, and the number of batches is still 20, so the computational cost is skewed well in favour of the batch methods. However the SGLD algorithm still outperforms the other methods.

4 Control variates to improve SGLD efficiency

The stochastic gradient versions of MALA and HMC, i.e. SGLD and SGHMC, presented in Section 2, use minibatches of the data to calculate unbiased estimates of the gradient of the log-posterior. We found these methods to be the most robust of those investigated in the comparisons of Section 3, so we focus on them for the rest of the paper, particularly SGLD.

The SGLD algorithm has a reduced per iteration computational cost compared to traditional MCMC algorithms. However, in Section 4.3, we compare SGLD with the true, coupled Langevin diffusion that SGLD is trying to approximate. We show that as we vary the stepsize, in order for the strong error of SGLD to reach a desired level, the minibatch size has to be set proportional to the dataset size N. Thus the algorithm still has $O(N)$ computational cost. As we make clear later, this $O(N)$ cost is due to the variance in the gradient estimate. Therefore, reducing the variance of the gradient estimate should improve the computational cost of the algorithm.

A natural choice for reducing this variance is through control variates (Ripley, 2009). Control variates applied to SGLD have recently been investigated by Dubey et al. (2016), who show that the convergence bound of SGLD is reduced when they are used. They have recently been applied in the continuous-time MCMC setting by Bierkens et al. (2016) and Pollock et al. (2016). Bardenet et al. (2016) introduce control variates to develop a Metropolis-Hastings algorithm which approximates the acceptance step using a subsample of the data.

In Section 4.1, we use control variates to reduce the variance in the gradient estimate used in SGLD. Then in Section 4.3 we compare SGLD with this improved gradient estimate with the true, coupled Langevin diffusion. We show that we can set a constant stepsize n_0 and the strong error will still reach a desired level for any dataset size N. This means the computational cost has been reduced from $O(N)$ to $O(1)$. We focus our theoretical results on SGLD, but the methodology can be applied to any SGMCMC algorithm, since we are only changing the gradient estimate. While earlier results have demonstrated the benefits of control variates applied to SGMCMC for a fixed data set, we believe this is the first result to show the beneficial scaling of control variates for SGLD as the dataset size increases.

4.1 Control variates in the gradient estimate

Let $\hat{\theta}$ be the mode of the posterior $p(\theta|\mathbf{x})$. The log posterior gradient can then be re-written as

$$\nabla \log p(\theta|\mathbf{x}) = \nabla \log p(\hat{\theta}|\mathbf{x}) + [\nabla \log p(\theta|\mathbf{x}) - \nabla \log p(\hat{\theta}|\mathbf{x})],$$

where the first term on the right-hand side is a constant and the bracketed term on the right-hand side can be unbiasedly estimated by

$$\left[\nabla \widehat{\log p(\theta}|\mathbf{x}) - \nabla \widehat{\log p(\hat{\theta}}|\mathbf{x})\right] = \nabla \log p(\theta) - \nabla \log p(\hat{\theta}) + \frac{N}{n}\sum_{i\in S}\left[\nabla \log p(x_i|\theta) - \nabla \log p(x_i|\hat{\theta})\right],$$

Algorithm 1 SGLD-CV

Require: $\hat{\theta}$, $\nabla \log p(\hat{\theta}|\mathbf{x})$, ϵ
  for $t \in 1, \ldots, T$ **do**
    Update $\widetilde{\nabla \log p(\theta_t|\mathbf{x})}$ using (4.1)
    Draw $\zeta_t \sim N(0, \epsilon I)$
    $\theta_{t+1} \leftarrow \theta_t + \frac{\epsilon}{2} \widetilde{\nabla \log p(\theta_t|\mathbf{x})} + \zeta_t$
  end for

where S is a random sample from $\{1, \ldots, N\}$ with $|S| = n$. If the gradient of the likelihood for a single observation is smooth in θ then we will have

$$\nabla \log p(x_i|\theta) \approx \nabla \log p(x_i|\hat{\theta})$$

if $\theta \approx \hat{\theta}$. Hence for $\theta \approx \hat{\theta}$ we would expect the unbiased estimator

$$\widetilde{\nabla \log p(\theta|\mathbf{x})} = \nabla \log p(\hat{\theta}|\mathbf{x}) + \left[\widehat{\nabla \log p(\theta|\mathbf{x})} - \widehat{\nabla \log p(\hat{\theta}|\mathbf{x})}\right]$$

to have a lower variance than the simpler unbiased estimator (4).

The gradient estimate $\widetilde{\nabla \log p(\theta|\mathbf{x})}$ can be substituted into any stochastic gradient MCMC algorithm in place of $\widehat{\nabla \log p(\theta|\mathbf{x})}$. We refer to SGLD using this alternative gradient estimate as SGLD-CV. The full procedure is outlined in Algorithm 1.

Implementing this in practice means finding a suitable $\hat{\theta}$. While we cannot find the true posterior mode, if $\hat{\theta}$ is within distance $O(N^{-\frac{1}{2}})$ of the true posterior mode then, as is shown below, the variance reduction will be by a factor of N. In this case the $O(1)$ computational cost results apply. Under standard asymptotics this is easily achieved. Under more complex asymptotics there should still be efficiency improvements, and we ensure to explore these more complex cases in the experiments (Section 6).

In practice, we find $\hat{\theta}$ using a stochastic optimization algorithm (Robbins and Monro, 1951), and then calculate the full log posterior gradient at this point $\nabla \log p(\hat{\theta}|\mathbf{x})$. So the algorithm requires a one-off $O(N)$ pre-processing step. We can then use control variates with a fixed batch size (i.e. independent of N) to estimate the gradient at each iteration of SGLD. Thus the algorithm has per iteration cost of $O(1)$.

The added computational cost of the optimization procedure is investigated in the simulation studies. We find in practice that the time to find a good $\hat{\theta}$ is often quicker than the time it takes for SGLD to burn-in. If the SGLD-CV is then started from $\hat{\theta}$ then the burn-in is minimal, so the cost of the optimization procedure is offset by the efficiency gain.

4.2 Variance reduction

The improvements of using the control variate gradient estimate (4.1) over the standard (4) become apparent when we calculate the variances of each. First, we assume the following Lipschitz smoothness and bound conditions. These assumptions are discussed in Chen et al. (2015).

Assumption 1. *Bounded gradients: Assume there exists $\sigma \in \mathbb{R}$ such that for all $\theta \in \mathbb{R}^d$ and for all $i \in \{1, \ldots, N\}$, $|\nabla \log p(\theta)| < \sigma$ and $|\nabla \log p(x_i|\theta)| < \sigma$.*
Lipschitz smoothness assumption: Assume there is some $D > 0$ such that

$$|\nabla \log p(x_i|\theta) - \nabla \log p(x_i|\theta')| < D|\theta - \theta'|,$$
$$|\nabla \log p(\theta) - \nabla \log p(\theta')| < D|\theta - \theta'|,$$

for all $i \in \{1, \ldots, N\}$ and for all $\theta \in \mathbb{R}^d$. We assume that D is independent of N.

The Lipschitz smoothness assumption is common for working with these kinds of problems (Dubey et al., 2016; Sato and Nakagawa, 2014). The Lipschitz constant D for the density and prior in Assumption 1 is assumed not to depend on N. But this is very natural, since these quantities are for single observations or independent of the data completely.

Define $\hat{\xi}_t = \nabla \widehat{\log p(\theta_t|\mathbf{x})} - \nabla \log p(\theta_t|\mathbf{x})$ and $\tilde{\xi} = \nabla \widetilde{\log p(\theta_t|\mathbf{x})} - \nabla \log p(\theta_t|\mathbf{x})$. So $\hat{\xi}$ and $\tilde{\xi}$ represent the error in the gradient estimates of SGLD and SGLD-CV, respectively. Under the above assumptions, Dubey et al. (2016) show that

$$\mathrm{Var}[|\hat{\xi}_t|] = \mathrm{E}[|\hat{\xi}_t|^2] \leq \frac{2N^2\sigma^2}{n} \tag{6}$$

$$\mathrm{Var}[|\tilde{\xi}_t|] = \mathrm{E}[|\tilde{\xi}_t|^2] \leq \frac{D^2N^2}{n}\mathrm{E}|\theta_t - \hat{\theta}|^2. \tag{7}$$

In the case of SGLD-CV, the variance in the gradient estimate depends on how close the current state θ_t is to $\hat{\theta}$. Suppose our estimate $\hat{\theta}$ is within $O(N^{-\frac{1}{2}})$ of the truth. Then, provided our MCMC has converged, and standard asymptotics hold, we would expect $\mathrm{E}|\theta_t - \hat{\theta}|$ to also be $O(N^{-\frac{1}{2}})$.

4.3 Computational cost of SGLD and SGLD-CV

In this section we investigate the computational cost of applying control variates within the gradient estimate of SGLD; and how this is affected by varying the size of the data N. Denote $\theta^{(N)}$ to be the parameters of interest when we have N observations. The posterior for $\theta^{(N)}$ contracts at rate $N^{-\frac{1}{2}}$. Suppose we want to run the SGLD algorithm so that it samples from the posterior distribution to a given accuracy. Then the number of iterations, the stepsize ϵ and the minibatch size n needed for the sample to stay below this error will change with varying N.

This motivates us to work with a rescaled parameter $\phi^{(N)} = \sqrt{N}\left(\theta^{(N)} - \theta_0^{(N)}\right)$, where $\theta_0^{(N)}$ is the posterior mode for the dataset of size N. Under standard asymptotics, the limiting posterior distribution for $\phi^{(N)}$ does not depend on N. So the mixing properties of the Langevin diffusion should be similar as we increase N. Also, in order to obtain a given Monte Carlo error, we would want to simulate the Langevin diffusion for a constant length of time under different sizes of N. The Langevin diffusion for $\phi^{(N)}$ can be directly related to the Langevin diffusion for $\theta^{(N)}$, defined in (3), via a linear-transformation of time and space. Hence results on the scalability of SGLD for $\phi^{(N)}$ will directly translate into equivalent results for the SGLD for $\theta^{(N)}$ that we run in practice.

We assume that, using this rescaled parameter $\phi^{(N)}$, we can fix ϵ, and the number of SGLD iterations T, with little effect on mixing. We compare our rescaled SGLD algorithm with the coupled, true underlying Langevin diffusion $\bar{\phi}_t^{(N)}$, which we formally define later. We then define our computational cost as the minibatch size n required for the quantity

$$\mathrm{E}\left|\phi_T^{(N)} - \bar{\phi}_T^{(N)}\right|^2,$$

commonly referred to as the strong error, to stay below an arbitrary level ν, for any fixed T. We investigate how this minibatch size depends on the number of observations N, for both the SGLD and SGLD-CV algorithms. In order for this to be a valid argument, we need to check that rescaling the parameter leads to a valid Langevin diffusion, and that other constants in our strong error bound do not implicitly depend on N.

Define $\beta_N(\phi) = N^{-\frac{1}{2}}\nabla \log p(\theta^{(N)}|\mathbf{x})$, where $\theta^{(N)} = \theta_0^{(N)} + N^{-\frac{1}{2}}\phi^{(N)}$. Our transformation for $\phi^{(N)}$ is independent of t, $\nabla\phi^{(N)}(\theta) = \sqrt{N}\mathbf{1}$ and $\nabla^2\phi^{(N)}(\theta) = 0$, where $\nabla^2 f$ is the Hessian of f. So applying Ito's Lemma to the Langevin diffusion we defined in (3) we have

$$d\phi_t^{(N)} = \frac{\sqrt{N}}{2}\nabla \log p(\theta^{(N)}|\mathbf{x})dt + \sqrt{N}dB_t,$$

We use the well known scaling rule for the Wiener process $\sqrt{\alpha}B_t = B_{\alpha t}$ to rescale time by $1/N$ and get the diffusion we will work with

$$d\phi_t^{(N)} = \frac{1}{2}\beta_N(\phi_t)dt + dB_t. \tag{8}$$

We will use the fact that the Langevin diffusion for phi also satisfies Lipschitz and linear growth conditions

Lemma 1. *Lipschitz Condition for $\beta_N(\phi)$: Under Assumption 1, there is some constant $C > 0$ that does not depend on N such that*

$$|\beta_N(\phi) - \beta_N(\phi')| < C|\phi^{(N)} - \phi'^{(N)}|,$$

for all $N > 0$, for all $\phi, \phi' \in \mathbb{R}^d$.
Linear growth condition: Under Assumption 1, there exists $C > 0$, independent of N, such that

$$|\beta_N(\phi)|^2 < C^2(1 + |\phi|^2),$$

for all $N > 0$, $\phi \in \mathbb{R}^d$

The proof for Lemma 1 is given in the Appendix. A particularly important fact from Lemma 1 is that the constant C does not depend on N in both cases. This, along with Assumption 1, ensures that there is no hidden dependency on N in the result for SGLD-CV. The assumption also ensures that (8) is a valid diffusion. In order for there to be no hidden dependency on N in the result for SGLD, we need to make one more assumption as follows

Assumption 2. *Lower bound on empirical Fisher information: Let $\bar{g}^j(\theta) := \frac{1}{N}\sum_{i=1}^N \partial_{\theta_j} \log p(x_i|\theta)$. Then there exists $W > 0$ such that*

$$\frac{1}{N-1}\sum_{i=1}^{N}\left[\partial_{\theta_j} \log p(x_i|\theta) - \bar{g}^j(\theta)\right]^2 \geq W, \tag{9}$$

for all $N \geq 1$, for all $\theta \in \mathbb{R}^d$.

This is a weak assumption as the RHS of (9) tends to $\mathrm{E}\left[-\partial_{\theta_j} \log p(X|\theta)\right]$, i.e. the j^{th} diagonal element of the Fisher information matrix for 1 observation.

We can see from the results of Dubey et al. (2016) in (7), the computational cost of SGLD-CV will depend on the distance between our simulated output $\phi_t^{(N)}$ and our estimate of the posterior mode $\hat{\phi}^{(N)}$; where we define $\hat{\phi}^{(N)} := \sqrt{N}(\hat{\theta}^{(N)} - \theta_0^{(N)})$. So this last assumption makes some conditions in order to ensure the distance between the states of the SGLD-CV algorithm and $\hat{\phi}$ is $O(1)$.

Assumption 3. *Distance between $\hat{\phi}$ and the true mode: Assume there exists finite E, that does not depend on N, such that*

$$\mathrm{E}\left|\hat{\phi} - \phi_0^{(N)}\right|^2 \leq E,$$

for all $N > 0$, where $\phi_0^{(N)}$ is the true mode.
Distance between true Langevin diffusion and true mode: Let $\bar{\phi}_t^{(N)}$ be the output from the true Langevin diffusion (8) initiated from some fixed point ϕ_0. Then assume there exists finite F, that does not depend on N, such that

$$\mathrm{E}\left|\bar{\phi}_t^{(N)} - \phi_0^{(N)}\right|^2 \leq F$$

for all $N > 0$, $t \in \mathbb{R}^+$.

The first case of Assumption 3 states that in our transformed ϕ space our estimate of the mode will be within constant distance of the true mode. This is equivalent to the estimate $\hat{\theta}$ being $O(N^{-\frac{1}{2}})$ from the true mode. This is a common assumption which holds under standard asymptotics. We choose complex scenarios for our experiments in Section 6, since in these cases the assumption may not be true. This serves to really test some of the underlying theory.

In Theorem 1 that we are about to state, we compare SGLD-CV to the true, coupled Langevin diffusion (8). This assumption states that the true Langevin diffusion is started from some point in our transformed space that is a constant distance away from true the mode. In the standard θ space, this is equivalent to the algorithm being started within $O(N^{-\frac{1}{2}})$ distance from the true posterior mode. Since SGLD-CV is started in the same position in the theorem, we assume the same thing for SGLD-CV.

This means the scaling results do not apply to the burn-in period if we start SGLD-CV arbitrarily far into the tail of the posterior. To get around this issue, we use an optimization step to find $\hat{\theta}$, and then start SGLD-CV at this point. This means that the burn-in step of SGLD is replaced by a single optimization step. Since SGLD-CV begins at a good starting point, the burn-in will be negligible. In our experiments in Section 6, we often find this optimization step is quicker than the SGLD burn-in step. So the algorithm has improved efficiency in both burn-in as well as the simulation step itself. The main cost of these improvements is in tuning, since you have to tune both a stochastic optimization step and an SGLD step.

Now the framework has been set up, we can state our result. As a result of Lemma 1, (8) is a valid diffusion (Sato and Nakagawa, 2014). Then we can use Lemma 1 and Assumption 2 to ensure that we do not have any hidden dependence on N, and our main result follows.

Theorem 1. *Let $\bar{\phi}_t^{(N)}$ be output from the true Langevin diffusion (8) and $\phi_t^{(N)}$ be the output from an equivalent SGLD algorithm with fixed stepsize ϵ and the same driving Brownian motion. Assume for all N, both processes are initiated from the same value ϕ_0. Also assume that minibatches are sampled without replacement from the full observation set. Let $\nu > 0$ be an appropriately chosen error tolerance, which is attainable for the chosen ϵ. Then for any fixed T:*

1. *For standard SGLD, under Assumption 2, we need a minibatch size n of $O(N)$ to obtain $\mathrm{E}|\phi_T^{(N)} - \bar{\phi}_T^{(N)}|^2 \leq \nu$.*
2. *For SGLD-CV, under Assumptions 1 and 3, there exists a fixed minibatch size n_0 of $O(1)$ such that $\mathrm{E}|\phi_T^{(N)} - \bar{\phi}_T^{(N)}|^2 \leq \nu$ for all $N > 0$.*

Theorem 1 shows that while SGLD uses minibatches of size n, meaning that the per iteration cost is much lower than N, the minibatch size n still scales linearly as N changes, if we want to reach a desired level of accuracy (measured in terms of strong error). In comparison we can fix a minibatch size n for SGLD-CV and reach a desired level accuracy no matter what the dataset size is. Before we can begin the SGLD-CV algorithm we need an optimization step to find $\hat{\theta}$, followed by a single $O(N)$ preprocessing step to find $\nabla \log p(\hat{\theta}|\mathbf{x})$, but after this startup cost our algorithm has improved efficiency. We demonstrate the scaling results empirically in the experiments (Section 6) by fitting the same models under different dataset sizes N.

The efficiency gains of using control variates depends on the distance $|\theta_t - \hat{\theta}|^2$. So for SGLD-CV to be $O(1)$ we require $|\theta_t - \hat{\theta}|^2$ to be $O(N^{-1})$, which holds under standard asymptotics. While this may not hold in more complex scenarios, in our simulations we find the efficiency gains are considerable, even for high dimensional, complex examples where these asymptotics may not hold (see the examples in Sections 6.2 and 6.3).

4.4 Other control variate algorithms

Other algorithms that use control variates should benefit from similar computational cost results. The recently introduced SAGA algorithm (Dubey et al., 2016) is an alternative algorithm that uses control variates to reduce the variance in the log posterior gradient estimate of SGLD. In this section we introduce this algorithm, and show that this algorithm has similar efficiency improvements to SGLD-CV. Then in the next section we compare the two algorithms and discuss their merits.

SAGA introduces a new set of variables α_t^i for $i = 1, \ldots, N$ and g_α, initialised at θ_0. At each iteration, α_{t+1}^i is set to θ_t if $i \in S_t$, otherwise, $\alpha_{t+1}^i = \alpha_t^i$. The updates for the algorithm are then as follows

$$\begin{aligned}
\hat{g}_{\alpha,t} &= g_\alpha + \frac{N}{n} \sum_{i \in S_t} \left[\nabla \log p(x_i|\theta_t) - \nabla \log p(x_i|\alpha_t^i) \right] \\
\theta_{t+1} &= \theta_t + \frac{\epsilon}{2} \left[\nabla \log p(\theta_t) + \hat{g}_{\alpha,t} \right] \\
g_\alpha &= g_\alpha + \sum_{i \in S_t} \left[\nabla \log p(x_i|\alpha_{t+1}^i) - \nabla \log p(x_i|\alpha_t^i) \right]
\end{aligned}$$

Theorem 2. *Let $\bar{\phi}_t^{(N)}$ be output from the true Langevin diffusion (8) and $\phi_t^{(N)}$ be the output from an equivalent SAGA algorithm with fixed stepsize ϵ and the same driving Brownian motion. Assume for all N, both processes are initiated from the same value ϕ_0. Also assume that minibatches are sampled without replacement from the full observation set. Let $\nu > 0$ be an appropriately chosen error tolerance, which is attainable for the chosen ϵ. Then for any fixed T:*
Under Assumptions 1 and 3, there exists a fixed minibatch size n_0 of $O(1)$ such that $\mathrm{E}|\phi_T^{(N)} - \bar{\phi}_T^{(N)}|^2 \leq \nu$ for all $N > 0$.

Interestingly the proof of Theorem 2, has a weaker bound than for Theorem 1. So the fixed minibatch size n_0 may need to be up to four times larger for equivalent performance of SAGA compared to SGLD-CV.

4.5 Discussion

In this section we showed that two methods which use control variates to reduce the variance in the gradient estimate of SGLD, SGLD-CV and the previously introduced SAGA (Dubey et al., 2016), reduce the computational cost of SGLD from $O(N)$ to $O(1)$.

SGLD-CV requires an initial optimization step to find an estimate of $\hat{\theta}$. While this adds to the computational time required by the algorithm, it essentially replaces the burn-in of SGLD, as we can start SGLD-CV from this estimate of the mode. We find in our experiments (Section 6) that this initial optimization step plus the negligible burn-in time for SGLD-CV is often quicker than the time taken for SGLD to burn-in, especially for complex scenarios. Though it does make tuning more difficult, as the optimization step must be tuned as well as the stepsize for SGLD itself. SGLD-CV also requires a single $O(N)$ preprocessing step in order to calculate $\nabla \log p(\hat{\theta}|\mathbf{x})$.

In contrast, SAGA does not require an initial optimization step, but the fact that previous states are reused means that minibatch size needed to reach a desired accuracy may need to be larger than SGLD-CV. In our simulations we found that SAGA works very well for simple problems (see the logistic regression example in Section 6.1), but for complicated models the algorithm takes a prohibitively long time to converge (see the probabilistic matrix factorization example in Section 6.2). This is mostly true when the algorithm starts far from the true posterior mode, as is often the case in complex examples. It seemed to be due to the algorithm getting stuck or moving very slowly through local modes.

In this section, we have focused on reducing the computational cost of SGLD using control variates. But the algorithm SGLD-CV only changes the gradient estimate calculation, so the methodology is just as easily applied to other stochastic gradient MCMC methods such as SGHMC and SGNHT (Chen et al., 2014; Ding et al., 2014). Similar bounds on the strong error exist for SGHMC and SGNHT, so extending computational cost results to these algorithms should be possible.

5 Post-processing control variates

Control variates can also been used to improve the inferences made from MCMC by reducing the variance of the output directly. The general aim of MCMC is to estimate expectations of functions, $g(\theta)$, under the

posterior $p(\theta|\mathbf{x})$. Given an MCMC sample $\theta^{(1)}, \ldots, \theta^{(M)}$, from the posterior $p(\theta|\mathbf{x})$, we can estimate $\mathrm{E}[g(\theta)]$ unbiasedly as

$$\mathrm{E}[g(\theta)] \approx \frac{1}{M} \sum_{i=1}^{M} g(\theta^{(i)}).$$

Suppose we introduce a new function $h(\theta)$, which has expectation 0 under the posterior. We can then introduce an alternative function,

$$\tilde{g}(\theta) = g(\theta) + h(\theta),$$

where $\mathrm{E}[\tilde{g}(\theta)] = \mathrm{E}[g(\theta)]$. If $h(.)$ is chosen so that it is negatively correlated with $g(\theta)$, then the variance of $\tilde{g}(\theta)$ will be reduced considerably.

Mira et al. (2013) introduce a way of choosing $h(\theta)$ almost automatically by using the gradient of the log-posterior. Choosing $h(.)$ in this manner is referred to as a zero variance (ZV) control variate. Friel et al. (2016) showed that, under mild conditions, we can replace the log-posterior gradient with an unbiased estimate and still have a valid control variate. SGMCMC methods produce unbiased estimates of the log-posterior gradient, and so it follows that these gradient estimates can be applied as ZV control variates. For the rest of this section, we focus our attention on SGLD, but these ideas are easily extendable to other stochastic gradient MCMC algorithms. We refer to SGLD with these post-processing control variates as SGLD-ZV.

Given the setup outlined above, Mira et al. (2013) propose the following form for $h(\theta)$,

$$h(\theta) = \Delta Q(\theta) + \nabla Q(\theta) \cdot \mathbf{z},$$

here $Q(\theta)$ is a polynomial of θ to be chosen and $\mathbf{z} = -\frac{1}{2}\nabla \log p(\theta|\mathbf{x})$. Δ refers to the *Laplace operator* $\frac{\partial^2}{\partial \theta_1^2} + \cdots + \frac{\partial^2}{\partial \theta_d^2}$. In order to get the best variance reduction, we simply have to optimize the coefficients of the polynomial $Q(.)$. In practice, first or second degree polynomials $Q(\theta)$ often provide good variance reduction (Mira et al., 2013). For the rest of this section we focus on first degree polynomials, so $Q(\theta) = \mathbf{a}^T\theta$, but the ideas are easily extendable to higher orders (Papamarkou et al., 2014).

The SGLD algorithm only calculates an unbiased estimate of $\nabla \log p(\theta|\mathbf{x})$, so we propose replacing $h(\theta)$ with the unbiased estimate

$$\hat{h}(\theta) = \Delta Q(\theta) + \nabla Q(\theta) \cdot \hat{\mathbf{z}}, \tag{10}$$

where $\hat{\mathbf{z}} = -\frac{1}{2}\widehat{\nabla \log p(\theta}|\mathbf{x})$. By identical reasoning to Friel et al. (2016), $\hat{h}(\theta)$ is a valid control variate. Note that $\hat{\mathbf{z}}$ can use any unbiased estimate, and as we will show later, the better the gradient estimate, the better this control variate performs.

We set $Q(\theta)$ to be a linear polynomial $\mathbf{a}^T\theta$, so our SGLD-ZV estimate will take the following form

$$\hat{g}(\theta) = g(\theta) + \mathbf{a}^T\hat{\mathbf{z}}. \tag{11}$$

Similar to standard control variates (Ripley, 2009), we need to find optimal coefficients $\hat{\mathbf{a}}$ in order to minimize the variance of $\tilde{g}(\cdot)$, defined in (11). In our case, the optimal coefficients take the following form (Friel et al., 2016)

$$\hat{\mathbf{a}} = \mathrm{Var}^{-1}(\hat{\mathbf{z}})\,\mathrm{Cov}(\hat{\mathbf{z}}, g(\theta)).$$

This means that SGLD already calculates all the necessary terms for these control variates to be applied for free. So the post-processing step can simply be applied once when the SGLD algorithm has finished, provided the full output plus gradient estimates are stored. With this in place, we can write down the full algorithm in the linear case, which is given in Algorithm 2. For higher order polynomials, the calculations are much the same, but more coefficients need to be estimated (Papamarkou et al., 2014).

The efficiency of ZV control variates in reducing the variance of our MCMC sample is directly affected by using an estimate of the gradient rather than the truth. For the remainder of this section, we investigate how the choice of the gradient estimate, and the minibatch size n, affects the variance reduction.

Assumption 4. $\mathrm{Var}[\phi(\theta)] < \infty$ *and* $\mathrm{Var}[\hat{\psi}(\theta)] < \infty$.

Algorithm 2 SGLD-ZV

Require: $\{\theta_t\}_{t=1}^T$, $\{\widehat{\nabla \log p(\theta_t|\mathbf{x})}\}_{t=1}^T$ ▷ SGLD output

Set $\mathbf{z}_t \leftarrow -\frac{1}{2}\widehat{\nabla \log p(\theta_t|\mathbf{x})}$

Estimate $V_{\mathbf{z}} \leftarrow \text{Var}(\mathbf{z})$, $C_{g,\mathbf{z}} \leftarrow \text{Cov}(g(\theta), \mathbf{z})$

$\hat{\mathbf{a}}_j \leftarrow [V_{\mathbf{z}}]^{-1} C_{g,\mathbf{z}}$

for $t = 1 \dots T$ **do**

$\hat{g}(\theta_t) \leftarrow g(\theta_t) + \hat{\mathbf{a}}^T \mathbf{z}_t$

end for

Theorem 3. *Under Assumption 4, define the optimal variance reduction for ZV control variates using the full gradient estimate to be R, and the optimal variance reduction using SGLD gradient estimates to be $\hat{R}$. Then we have that*

$$\hat{R} \geq \frac{R}{1 + CN^{-1}\text{E}_{\theta|\mathbf{x}}[\text{E}_S(|\xi_S(\theta)|)]}, \tag{12}$$

where $\xi_S(\theta)$ is the noise in the log-posterior gradient estimate, and C is some constant of $O(1)$.

The proof of this result is given in the Appendix. An important consequence of Theorem 3 is that if we use the standard SGLD gradient estimate, then the denominator of (12) is $O(n/N)$, so our variance reduction diminishes as N gets large. However, if we use the SGLD-CV or SAGA estimate instead, then the denominator of (12) is $O(n)$, so the variance reduction does not diminish with increasing dataset size. So for best results, we recommend using the ZV post-processing step after running the SGLD-CV or SAGA algorithm, especially for large N. The ZV post-processing step can be immediately applied in exactly the same way to other stochastic gradient MCMC algorithms, such as SGHMC and SGNHT (Chen et al., 2014; Ding et al., 2014).

It's worth noting that there are some storage constraints for SGLD-ZV. This algorithm requires storing the full MCMC chain, as well as the gradient estimates at each iteration. So the storage cost is twice the storage cost of a standard SGMCMC run, which is fine. However in some high dimensional cases, the required SGMCMC test statistic is estimated on the fly using the most recent output of the chain, as the storage costs are too large. But this is not possible when applying the ZV post-processing step. We suggest that if the dimensionality is not too high and you plan on storing the chain then it is well worth your time applying the ZV post-processing step, as it comes for free. However if the dimensionality is large it may not be worth your while.

6 Experiments

6.1 Logistic regression

We examine our approaches on a Bayesian logistic regression problem. The probability of the i^{th} output $y_i \in \{-1, +1\}$ is given by

$$p(y_i|x_i, \beta) = \frac{1}{1 + \exp(-y_i \beta^T x_i)}.$$

We use a Laplace prior for β with scale 1.

We used the cover type dataset (Blackard and Dean, 1999), which has 581,012 observations, which we split into a training and test set. First we run SGLD, SGLD-CV and SAGA on the dataset, all with minibatch size 500. To empirically support the scalability results of Theorem 1, we fit the model 3 times. In each fit, the dataset size is varied, from about 1% of the full dataset to the full dataset size N. The performance is measured by calculating the log predictive density on a held-out test set every 10 iterations. Some of our examples are high dimensional, so our performance measure aims to reduce dimensionality while still capturing important quantities such as the variance of the chain. We include the burn-in of SGLD

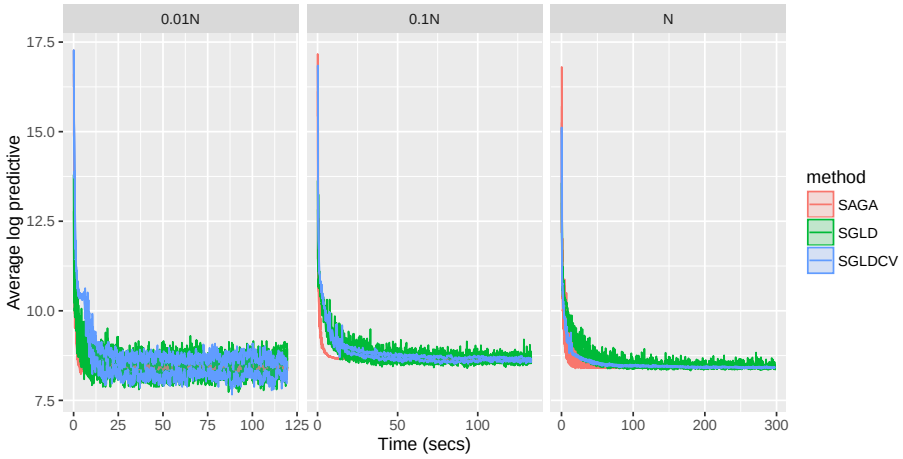

Figure 5: Log predictive density over a test set every 10 iterations of SGLD, SGLD-CV and SAGA fit to a logistic regression model as the data size N is varied.

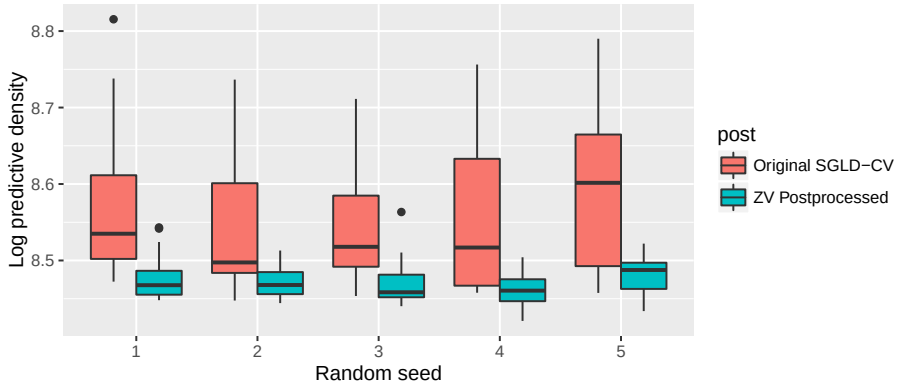

Figure 6: Plots of the log predictive density of an SGLD-CV chain when ZV post-processing is applied versus when it is not, over 5 random runs. Logistic regression model on the cover type dataset (Blackard and Dean, 1999).

and SAGA, to contrast with the optimization step required for SGLD-CV which is included in the total computational time.

The results are plotted against time in Figure 5. The results illustrate the efficiency gains of SGLD-CV and SAGA over SGLD as the dataset size increases, as expected from Theorems 1 and 2. SAGA outperforms SGLD-CV in this example because SGLD converges quickly in this simple setting. In the more complicated examples to follow, we show that SAGA can be slow to converge.

We also compare the log predictive density over a test set for SGLD-CV with and without ZV post-processing, averaged over 5 runs at different seeds. We apply the method to SGLD-CV rather than SGLD due to the favourable scaling results as discussed after Theorem 3. Results are given in Figure 6. The plot shows box-plots of the log predictive density of the SGLD sample before and after post-processing using ZV control variates. The plots show excellent variance reduction of the chain.

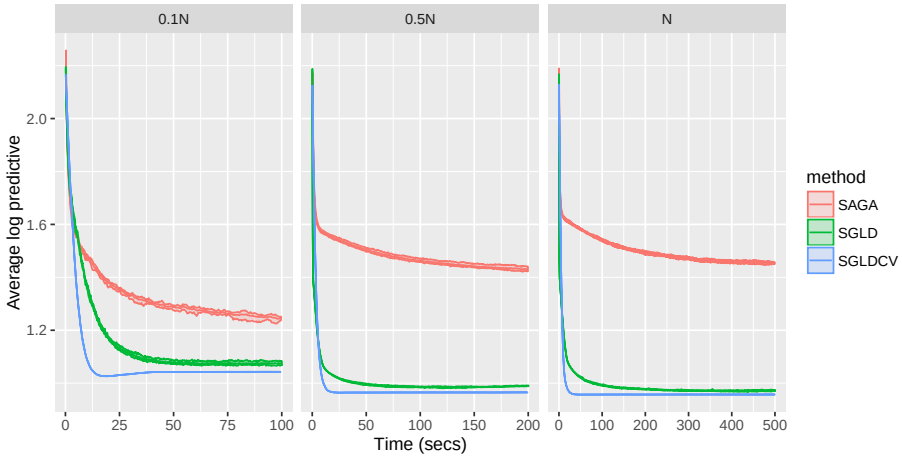

Figure 7: Log predictive density over a test set of SGLD, SGLD-CV and SAGA fit to a Bayesian probabilistic matrix factorization model as the number of users is varied, averaged over 5 runs. We used the Movielens ml-100k dataset.

6.2 Probabilistic matrix factorization

A common recommendation system task is to predict a user's rating of a set of items, given previous ratings and the ratings of other users. The end goal is to recommend new items that the user will rate highly. Probabilistic matrix factorization (PMF) is a popular method to train these models (Mnih and Salakhutdinov, 2008). As the matrix of ratings is sparse, over-fitting is a common issue in these systems, and Bayesian approaches are a way to account for this (Ahn et al., 2015).

In this experiment, we apply SGLD, SGLD-CV and SAGA to a Bayesian PMF problem, using a model similar to Ahn et al. (2015) and Chen et al. (2014). We use the Movielens dataset ml-100k[1], which contains 100,000 ratings from almost 1,000 users and 1,700 movies. We use batch sizes of 5,000, we set a larger minibatch size in this case due to the high-dimensional space. As before, we compare performance by calculating the predictive distribution on a held out dataset every 10 iterations.

We investigate the scaling results of SGLD-CV and SAGA versus SGLD by varying the dataset size. We do this by limiting the number of users in the dataset, ranging from 100 users to the full 943. The results are given in Figure 7. Once again the scaling improvements of SGLD-CV as the dataset size increases are clear.

In this example SAGA converges slowly in comparison even to SGLD. In fact the algorithm converges slowly in all our more complex experiments. The problem is particularly bad for large N. We think this could be a problem when the starting point of SAGA is too far from the posterior mode. Empirically we found that the gradient direction and magnitude can update very slowly in these cases. This is not an issue for simpler examples such as logistic regression. But for more complex examples we believe it could be a sign the algorithm is getting stuck in, or moving slowly through local modes where the gradient is comparatively flatter. The problem appears to be made worse for large N when it takes longer to update g_α. We suggest to alleviate this problem that SAGA needs to begin at a good starting point. This is an example where the optimization step of SGLD-CV is an advantage, as the algorithm is immediately started close to the posterior mode and so the efficiency gains are quick to take effect.

Once again we compare the log predictive density over a test set for SGLD-CV with and without ZV post-processing when applied to the Bayesian PMF problem, averaged over 5 runs at different seeds. Results are given in Figure 6. The plot shows box-plots of the log predictive density of the SGLD sample before and after post-processing using ZV control variates. The plots show excellent variance reduction of the chain.

[1]https://grouplens.org/datasets/movielens/100k/

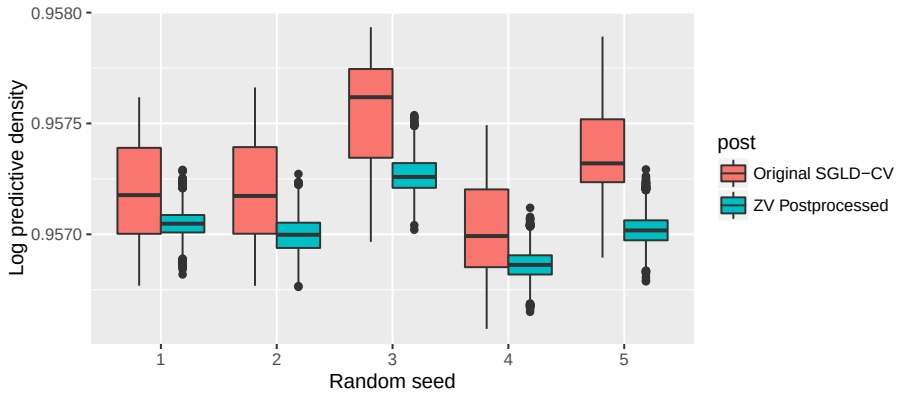

Figure 8: Plots of the log predictive density of an SGLD-CV chain when ZV post-processing is applied versus when it is not, over 5 random runs. SGLD-CV algorithm applied to a Bayesian probablistic matrix factorization problem using the Movielens ml-100k dataset.

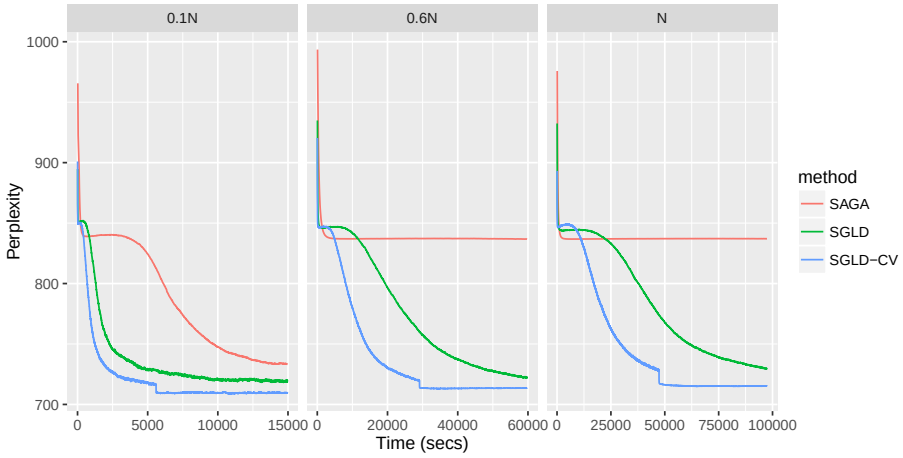

Figure 9: Perplexity of SGLD and SGLD-CV fit to an LDA model as the data size N is varied, averaged over 5 runs. The dataset consists of scraped Wikipedia articles.

6.3 Latent Dirichlet allocation

Latent Dirichlet allocation (LDA) is an important model in document used to describe collections of documents by sets of discovered topics (Blei et al., 2003). The input consists of a matrix of word frequencies in each document, which is very sparse. So again over-fitting is a concern which motivates the use of a Bayesian approach.

Due to storage constraints, it was not feasible to apply SGLD-ZV to this problem, so we focus on SGLD-CV. We scraped approximately 80,000 documents from Wikipedia, and used the 1,000 most common words to form our document-word matrix input. We used a similar formulation to Patterson and Teh (2013), though we did not use a Riemannian sampler.

Once again in our comparison of SGLD, SGLD-CV and SAGA, we vary the dataset size, this time by changing the number of documents used in fitting the model, from 10000 to the full 81538. We use batch sizes of 50 documents. We measure the performance of LDA using the *perplexity* on held out words from each document, a standard performance measure for this model. The results are given in Figure 9. Here the scalability improvements of using SGLD-CV over SGLD are clear as the dataset size increases. This time

the batch size is small compared to the dataset size, which probably makes the scalability improvements more obvious. The sudden drop in perplexity for the SGLD-CV plot occurs at the switch from the stochastic optimization step to SGLD-CV. It is probably due to the algorithm making efficient use of the Gibbs step to simulate the latent topics.

An interesting aspect of this problem is that it appears to have a pronounced local mode where each of the methods get stuck in (this can be seen by the blip in the plot at a perplexity of around 850). SGLD-CV is first to escape followed by SGLD, but SAGA takes a long time to escape. This is probably due to a similar aspect as the one discussed in the previous experiment (Section 6.2). Similar to the previous experiment, we find that while SAGA seems trapped, its gradient estimate changes very little, which could be a sign that the algorithm is moving very slowly through an area with a relatively flat gradient, such as a local mode. A simple solution would be to start SAGA closer to the mode, using a stochastic optimization or SGLD scheme to get it there initially.

7 Discussion

We have provided a comprehensive comparison of popular minibatch and divide and conquer methods on a range of posterior distributions. This motivated our focus towards SGMCMC algorithms, which we found to be more robust. There has been limited comparison across both divide and conquer and stochastic gradient methods, so we hope that these comparisons are illustrative of the challenges of big data MCMC methods; and help direct future research in this area.

Control variates are useful variance reduction techniques for Monte Carlo estimators. We add a significant contribution to the recent work on applying control variates to gradient estimation for SGMCMC algorithms. We have shown that both our proposed SGLD-CV and the previously proposed SAGA (Dubey et al., 2016) algorithms reduce the computational cost of the stochastic gradient Langevin algorithm from $O(N)$ to $O(1)$; providing more efficient simulation of the posterior after burn-in. We have empirically supported these scalability results on a variety of interesting problems from the statistics and machine learning literature using real world datasets; though we found that SAGA could be slow to burn-in on more complex examples. This was probably due to it getting stuck in or moving slowly through local modes, where the gradient is flatter. An interesting future extension would be reducing the startup cost of SGLD-CV, along with making it more automatic.

We showed that stochastic gradient MCMC methods calculate all the information needed to apply zero variance post-processing control variates. This improves the inference of the output by reducing its variance. We explored how the variance reduction is affected by the minibatch size and the gradient estimate, and show using SGLD-CV or SAGA rather than SGLD can achieve a better variance reduction. We demonstrated this variance reduction empirically. A limitation of these post-processing control variates is they require the whole chain, which can lead to high storage costs if the dimensionality of the sample space is high. Future work could explore ways to reduce the storage costs of stochastic gradient MCMC.

8 Acknowledgements

The first author gratefully acknowledges the support of the EPSRC funded EP/L015692/1 STOR-i Centre for Doctoral Training. This work was supported by EPSRC grant EP/K014463/1, ONR Grant N00014-15-1-2380 and NSF CAREER Award IIS-1350133.

References

Ahn, S., Korattikara, A., Liu, N., Rajan, S., and Welling, M. (2015). Large-scale distributed Bayesian matrix factorization using stochastic gradient MCMC. In *In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 9–18. ACM.

Bardenet, R., Doucet, A., and Holmes, C. (2016). On Markov chain Monte Carlo methods for tall data. *Journal of Machine Learning Research*. To appear.

Bierkens, J., Fearnhead, P., and Roberts, G. (2016). The zig-zag process and super-efficient sampling for Bayesian analysis of big data. Available from `https://arxiv.org/abs/1607.03188`.

Blackard, J. A. and Dean, D. J. (1999). Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1).

Chen, C., Ding, N., and Carin, L. (2015). On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *Advances in Neural Information Processing Systems 28*, pages 2278–2286. Curran Associates, Inc.

Chen, T., Fox, E., and Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1683–1691. PMLR.

Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D., and Neven, H. (2014). Bayesian sampling using stochastic gradient thermostats. In *Advances in Neural Information Processing Systems 27*, pages 3203–3211. Curran Associates, Inc.

Dubey, K. A., Reddi, S. J., Williamson, S. A., Poczos, B., Smola, A. J., and Xing, E. P. (2016). Variance reduction in stochastic gradient Langevin dynamics. In *Advances in Neural Information Processing Systems 29*, pages 1154–1162. Curran Associates, Inc.

Friel, N., Mira, A., and Oates, C. (2016). Exploiting multi-core architectures for reduced-variance estimation with intractable likelihoods. *Bayesian Analysis*, 11(1):215–245.

Giles, M., Nagapetyan, T., Szpruch, L., Vollmer, S., and Zygalakis, K. (2016). Multilevel Monte Carlo for scalable Bayesian computations. Available from `https://arxiv.org/abs/1609.06144`.

Gorham, J., Duncan, A. B., Vollmer, S. J., and Mackey, L. (2016). Measuring sample quality with diffusions. Available from `https://arxiv.org/abs/1611.06972`.

Korattikara, A., Chen, Y., and Welling, M. (2014). Austerity in MCMC land: Cutting the Metropolis-Hastings budget. In *Proceedings of The 31st International Conference on Machine Learning*, pages 181–189. PMLR.

Le Cam, L. (2012). *Asymptotic methods in statistical decision theory*. Springer Science & Business Media.

Li, S., Beygelzimer, A., Kakadet, S., Langford, J., Arya, S., and Mount, D. (2013). FNN: fast nearest neighbor search algorithms and applications. R package version 1.1. Available at `https://cran.r-project.org/web/packages/FNN/`.

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

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

Mira, A., Solgi, R., and Imparato, D. (2013). Zero variance Markov chain Monte Carlo for Bayesian estimators. *Statistics and Computing*, 23(5):653–662.

Mnih, A. and Salakhutdinov, R. R. (2008). Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20*, pages 1257–1264. Curran Associates, Inc.

Neal, R. M. (2010). MCMC using Hamiltonian Dynamics. In *Handbook of Markov Chain Monte Carlo*. Chapman & Hall.

Neiswanger, W., Wang, C., and Xing, E. P. (2014). Asymptotically exact, embarrassingly parallel MCMC. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, pages 623–632. AUAI Press.

Nemeth, C. and Sherlock, C. Merging MCMC subposteriors through Gaussian process approximations. Available at `https://arxiv.org/abs/1605.08576`.

Papamarkou, T., Mira, A., Girolami, M., et al. (2014). Zero variance differential geometric Markov chain Monte Carlo algorithms. *Bayesian Analysis*, 9(1):97–128.

Patterson, S. and Teh, Y. W. (2013). Stochastic gradient Riemannian Langevin dynamics on the probability simplex. In *Advances in Neural Information Processing Systems 26*, pages 3102–3110. Curran Associates, Inc.

Pollock, M., Fearnhead, P., Johansen, A. M., and Roberts, G. O. (2016). The scalable Langevin exact algorithm: Bayesian inference for big data. Available at `https://arxiv.org/abs/1609.03436`.

Ripley, B. D. (2009). *Stochastic simulation*. John Wiley & Sons.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.

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.

Sato, I. and Nakagawa, H. (2014). Approximation analysis of stochastic gradient Langevin dynamics by using Fokker-Planck equation and Ito process. In *Proceedings of the 31st International Conference on Machine Learning*, pages 982–990. PMLR.

Scott, S. (2017). Comparing Consensus Monte Carlo strategies for distributed Bayesian computation. *Brazillian Journal of Probability and Statistics*. To appear.

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

Srivastava, S., Cevher, V., Dinh, Q., and Dunson, D. (2015). WASP: Scalable Bayes via barycenters of subset posteriors. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 912–920. PMLR.

Teh, Y. W., Thiéry, A. H., and Vollmer, S. J. (2016). Consistency and fluctuations for stochastic gradient Langevin dynamics. *Journal of Machine Learning Research*, 17(7):1–33.

Vollmer, S. J., Zygalakis, K. C., et al. (2016). (Non-) asymptotic properties of stochastic gradient Langevin dynamics. *Journal of Machine Learning Research*, 17(159):1–48.

Wang, X. and Dunson, D. B. (2013). Parallelizing MCMC via Weierstrass sampler. Available at `https://arxiv.org/abs/1312.4605`.

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

Wenzhe Li, Sungjin Ahn, M. W. (2016). Scalable MCMC for mixed membership stochastic blockmodels. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 723–731. PMLR.

A Computational cost theorems

A.1 Lemmas

A.1.1 Proof of Lemma 1

From Assumption 1 and the triangle inequality

$$\begin{aligned}|\beta_N(\phi) - \beta_N(\phi')| &\leq D(N+1)N^{-\frac{1}{2}} \left|N^{-\frac{1}{2}}\phi^{(N)} - N^{-\frac{1}{2}}\phi'^{(N)}\right| \\ &\leq C\left|\phi^{(N)} - \phi'^{(N)}\right|,\end{aligned}$$

where $C = 2D$. We have chosen the looser bound $C = 2D$, in order to ensure that C does not depend on N, which is important to make our analysis valid.

Now let's look at the linear growth condition. We have proven the Lipschitz condition so that

$$\begin{aligned}|\beta_N(\phi)| &= |\beta_N(\phi) - \beta_N(0) + \beta_N(0)| \\ &\leq |\beta_N(\phi) - \beta_N(0)| + |\beta_N(0)| \\ &\leq C|\phi^{(N)}| + |\beta_N(0)| \\ &\leq C|\phi^{(N)}| + N^{-\frac{1}{2}}\left|\nabla \log p(\theta_0^{(N)}|\mathbf{x})\right| \\ &= C|\phi^{(N)}|\end{aligned}$$

where the first inequality follows from the triangle inequality, the second follows from the Lipschitz condition just proven. The last inequality follows from the fact that $\theta_0^{(N)}$ is the mode of $p(\theta^{(N)}|\mathbf{x})$. It follows that

$$|\beta_N(\phi)|^2 \leq C^2|\phi^{(N)}|^2 \leq C^2(1 + |\phi^{(N)}|^2),$$

as required. □

A.1.2 Distance between Langevin diffusion and estimated mode

Lemma 2. *Distance between Langevin diffusion and estimated mode: Under Assumption 3, there exists $H > 0$ such that*

$$\left|\bar{\phi}_t^{(N)} - \hat{\phi}^{(N)}\right|^2 \leq H \tag{13}$$

for all $N > 0$.

Proof. We can trivially apply Cauchy-Schwarz inequality followed by Assumption 3 to get

$$\begin{aligned}\left|\bar{\phi}_t^{(N)} - \hat{\phi}^{(N)}\right|^2 &= \left|\bar{\phi}_t^{(N)} - \phi_0^{(N)} + \phi_0^{(N)} - \hat{\phi}^{(N)}\right|^2 \\ &\leq 2\left[\left|\bar{\phi}_t^{(N)} - \phi_0^{(N)}\right|^2 + \left|\hat{\phi}^{(N)} - \phi_0^{(N)}\right|^2\right] \\ &\leq 2[F + E].\end{aligned}$$

So set $H = 2[F + E]$ and the result follows. □

A.2 Proof of Theorem 1

Define $\hat{\beta}_N(\phi) = N^{-\frac{1}{2}}\nabla \log \widehat{p(\theta^{(N)}|\mathbf{x})}$ and $\tilde{\beta}_N(\phi) = N^{-\frac{1}{2}}\nabla \log \widetilde{p(\theta^{(N)}|\mathbf{x})}$. Then set $\hat{\delta} = \hat{\beta}_N(\phi) - \beta_N(\phi)$ and $\tilde{\delta} = \tilde{\beta}_N(\phi) - \beta_N(\phi)$, so that $\hat{\delta}$ and $\tilde{\delta}$ represent the gradient estimate noise of SGLD and SGLD-CV respectively in the ϕ space. Let $\bar{\phi}_t$ be a realization from the true underlying Langevin diffusion at time $t\epsilon$ and let ϕ_t be iteration t from an SGLD algorithm.

In order to calculate the computational cost we bound the strong error $\mathrm{E}|\phi_T - \bar{\phi}_T|^2$ for each of the different methods. We then measure the minibatch size n which sets this bound at some acceptable level. We need to bound the strong error in a slightly different way for the two methods SGLD and SGLD-CV, since for SGLD we need a lower bound and for SGLD-CV we need an upper bound.

A.2.1 Scaling result for SGLD

First we'll look at the scaling result for SGLD. Suppose we want to look at the square error between ϕ_t and $\bar{\phi}_t$ at some arbitrary time t. By using the fact that

$$\phi_t = \phi_{t-1} + \frac{\epsilon}{2}\beta_N(\phi_{t-1}) + \zeta_{t-1} + \epsilon\hat{\delta}_{t-1},$$

we can get the upper bound for $\mathrm{E}|\phi_t - \bar{\phi}_t|^2$ to depend completely on the gradient estimate noise. Using the fact that $\hat{\delta}_{t-1}$ is independent of ζ_{t-1} and $\bar{\phi}_t$; $\mathrm{E}(\hat{\delta}_{t-1}) = 0$ and that the other quantities are not random we have

$$\begin{aligned}
\mathrm{E}|\phi_t - \bar{\phi}_t|^2 &= \mathrm{E}\left|\phi_{t-1} + \frac{\epsilon}{2}\beta_N(\phi_{t-1}) + \zeta_{t-1} + \epsilon\hat{\delta}_{t-1} - \bar{\phi}_t\right|^2 \\
&= \mathrm{E}\left|\phi_{t-1} + \frac{\epsilon}{2}\beta_N(\phi_{t-1}) + \zeta_{t-1} - \bar{\phi}_t\right|^2 + \epsilon^2\mathrm{E}|\hat{\delta}_{t-1}|^2 \\
&\geq \epsilon^2\mathrm{E}|\hat{\delta}_{t-1}|^2.
\end{aligned}$$

So that if we prove that $\mathrm{E}|\hat{\delta}_t|^2$ is $O(N)$ for any $t = 1, \ldots, T$ then it follows that the strong error of SGLD will be $O(N)$. Since we are proving the result for any t, we drop the dependence on t for brevity. Suppose the dimension of ϕ is d, then we'll set $\beta_N^j(\phi)$ to be the j^{th} dimensional component of β_N. It follows that

$$\begin{aligned}
\mathrm{E}|\hat{\delta}|^2 &= \mathrm{E}\left[\left(\hat{\beta}_N(\phi) - \beta_N(\phi)\right)^T\left(\hat{\beta}_N(\phi) - \beta_N(\phi)\right)\right] \\
&= \sum_{j=1}^{d}\mathrm{E}\left[\hat{\beta}_N^j(\phi) - \mathrm{E}\hat{\beta}_N^j(\phi)\right]^2 \\
&= \sum_{j=1}^{d}\mathrm{Var}\left[\hat{\beta}_N^j(\phi)\right]
\end{aligned}$$

where we have used that $\mathrm{E}\hat{\beta}_N^j = \beta_N^j$.

Now define $g_0^j(\theta) = \partial_{\theta_j}\log p(\theta)$, $g_i^j(\theta) = \partial_{\theta_j}\log p(x_i|\theta^{(N)})$. We assume that we are sampling without replacement, so we can represent $\widehat{\nabla\log p(\theta|\mathbf{x})} = \nabla\log p(\theta) + \sum_{i=1}^{N}\nabla\log p(x_i|\theta)Z_i$, where Z_i is drawn from

a categorical distribution with N possible outcomes, each with probability n/N.

$$\begin{aligned}
\mathrm{E}|\hat{\delta}|^2 &= \sum_{j=1}^{d} \mathrm{Var}\left[\beta_N^j(\phi)\right] \\
&= \sum_{j=1}^{d} \mathrm{Var}\left[N^{-\frac{1}{2}}\left(g_0^j(\theta) + \frac{N}{n}\sum_{i=1}^{N} g_i^j(\theta) Z_i\right)\right] \\
&= \frac{N}{n^2}\sum_{j=1}^{d} \mathrm{Var}\left[\sum_{i=1}^{N} g_i^j(\theta) Z_i\right] \\
&= \frac{N}{n^2}\sum_{i=1}^{d}\left(\sum_{i=1}^{N} \mathrm{Var}\left[g_i^j(\theta) Z_i\right] + \sum_{i\neq l} \mathrm{Cov}\left[g_i^j(\theta) Z_i, g_l^j(\theta) Z_l\right]\right) \\
&= \frac{N-n}{nN}\sum_{i=1}^{d}\left(\sum_{i=1}^{N}[g_i^j(\theta)]^2 - \frac{1}{N-1}\sum_{i\neq l}[g_i^j(\theta)][g_l^j(\theta)]\right).
\end{aligned}$$

Set $\bar{g}^j(\theta) := \frac{1}{N}\sum_{i=1}^{N} g_i^j(\theta)$ then we have

$$\mathrm{E}|\hat{\delta}|^2 = \frac{(N-n)}{(N-1)n}\sum_{i=1}^{d}\sum_{i=1}^{N}\left(g_i^j(\theta) - \bar{g}^j(\theta)\right)^2.$$

Now we can apply Assumption 2 to get

$$\mathrm{E}|\hat{\delta}|^2 \geq \frac{(N-n)}{n} W d$$

Now suppose we want

$$\mathrm{E}\left|\phi_T^{(N)} - \bar{\phi}_T^{(N)}\right|^2 < \nu,$$

then it follows that we would need

$$\frac{\epsilon^2 W d(N-n)}{n} \leq \epsilon^2 \mathrm{E}|\hat{\delta}|^2 \leq \mathrm{E}\left|\phi_T^{(N)} - \bar{\phi}_T^{(N)}\right|^2 \leq \nu.$$

So that

$$n \geq \frac{\epsilon^2 W d N}{\nu + \epsilon^2 W d}.$$

It follows that, in order to bound the strong error by an acceptable level ν, then we need to set a minibatch size of $O(N)$.

A.2.2 Scaling result for SGLD-CV

We want to prove that we can pick a minibatch size n_0, fixed for all $N > 0$, such that

$$\mathrm{E}\left|\phi_t^{(N)} - \bar{\phi}_t^{(N)}\right|^2 \leq \nu. \tag{14}$$

We will do this by induction. First note that for $t = 0$, SGLD-CV and Langevin diffusion are started at the same point. So trivially the distance between the two will be less than ν for any minibatch size n. Now assume such a minibatch size n_0 can be picked for all time points up to $t-1$, so $\mathrm{E}\left|\phi_s^{(N)} - \bar{\phi}_s^{(N)}\right|^2 < \nu$ for all $s \leq t-1$. We prove the result for t.

We can use the result of Sato and Nakagawa (2014) in their proof of Theorem 5 to write

$$\mathrm{E}[|\phi_t - \bar{\phi}_t|^2] \leq A + B \max_{t \in 1 \ldots t-1} \mathrm{E}[|\tilde{\delta}_t|^2], \tag{15}$$

where A and B are constants depending only on the fixed quantities that we have determined to be independent of N (the Lipschitz constants and the linear growth constant due to Lemma 1, the starting point and the stepsize). Notice that using the result of Dubey et al. (2016) specified in (7), we have that

$$\mathrm{E}|\tilde{\delta}_t|^2 = N^{-1}\mathrm{E}\left|\widetilde{\nabla \log p(\theta_t^{(N)}}|\mathbf{x}) - \nabla \log p(\theta_t^{(N)}|\mathbf{x})\right|^2 \tag{16}$$

$$\leq \frac{D^2 N}{n}\left|\theta_t^{(N)} - \hat{\theta}_t\right| \tag{17}$$

$$= \frac{D^2}{n}\left|\phi_t^{(N)} - \hat{\phi}^{(N)}\right|, \tag{18}$$

where the first inequality follows from (7), the last equality follows from the definition of $\phi^{(N)}$.

Define $\tau := \mathrm{argmax}_t \mathrm{E}[|\hat{\delta}|^2]$, then we can use (7) to obtain

$$\mathrm{E}[|\phi_t - \bar{\phi}_t|^2] \leq A + \frac{BD^2}{n_0}\mathrm{E}\left|\phi_\tau^{(N)} - \hat{\phi}^{(N)}\right|^2 \tag{19}$$

where we have used the Lipschitz condition of Assumption 1, so that D does not depend on N, and we have defined $\hat{\phi}^{(N)} := \sqrt{N}\left(\hat{\theta} - \theta_0^{(N)}\right)$. This bound still relies on the distance between $\phi_\tau^{(N)}$ and $\hat{\phi}$, but we know that $\tau \leq t-1$, so we can apply the Cauchy-Schwarz inequality followed by the inductive hypothesis to get

$$\begin{aligned}
\mathrm{E}[|\phi_t - \bar{\phi}_t|^2] &\leq A + \frac{BD^2}{n_0}\mathrm{E}\left|\phi_\tau^{(N)} - \bar{\phi}_\tau^{(N)} + \bar{\phi}_\tau^{(N)} - \hat{\phi}^{(N)}\right|^2 \\
&\leq A + \frac{2BD^2}{n_0}\left[\mathrm{E}\left|\phi_\tau^{(N)} - \bar{\phi}_\tau^{(N)}\right|^2 + \mathrm{E}\left|\bar{\phi}_\tau^{(N)} - \hat{\phi}^{(N)}\right|^2\right] \\
&\leq A + \frac{2BD^2}{n_0}\left[\nu + H\right],
\end{aligned}$$

where the second inequality follows from the Cauchy-Schwarz inequality, the third inequality follows from the inductive hypothesis and Lemma 2. So define $\nu = A + \frac{2BD^2}{n_0}[\nu + H]$, and the result holds for t, and so the result is proved for all t by induction. Thus pick $n_0 = \frac{2BD^2(\nu+H)}{\nu - A}$, then $\mathrm{E}\left|\phi_t - \bar{\phi}_t\right|^2 \leq \nu$ for all t as required. So specifically, for this choice of n_0, $\mathrm{E}\left|\phi_T - \bar{\phi}_T\right|^2 \leq \nu$. □

A.3 Proof of Theorem 2

We assume that α_t^i is initialized at $\phi_0^{(N)}$, so that it is either at this point or set to $\theta_\tau^{(N)}$ for some $\tau = 1, \ldots, T$. So the parameter does not need to be transformed. For brevity we drop the (N) superscript for α_t^i. Then define $g_{\alpha,t}^{(N)} := g_\alpha + \frac{N}{n}\sum_{i \in S_t}\left[N^{-\frac{1}{2}}\nabla \log p(x_i|\theta_t^{(N)}) - N^{-\frac{1}{2}}\nabla \log p(x_i|\alpha_t^i)\right]$. We define $\hat{\delta}_t$ to be the error in the SAGA log posterior gradient estimate at time t, so that it's given by $\hat{\delta}_t := \nabla \log p(\theta_t^{(N)}) + g_{\alpha,t}^{(N)} - \nabla \log p(\theta_t^{(N)}|\mathbf{x})$.

Similar to the proof of Theorem 1, we proceed by induction. Again since both the true diffusion and the SAGA algorithm are initialised at the same point, the distance between the two is less than ν at $t = 0$, for any minibatch size n. Now for $s \leq t-1$, assume that we can pick a constant minibatch size n_0 such that $\mathrm{E}\left|\phi_s^{(N)} - \bar{\phi}_s^{(N)}\right| \leq \nu$ for all $N > 0$. Now we prove the result for t as follows

Similar to the proof in Theorem 1, we can use the result of Sato and Nakagawa (2014) in their proof of Theorem 5 to write

$$\mathrm{E}[|\phi_t - \bar{\phi}_t|^2] \leq A + B \max_{t \in 1 \ldots t-1} \mathrm{E}[|\hat{\delta}_t|^2], \tag{20}$$

where A and B are constants depending only on the fixed quantities that we have determined to be independent of N (the Lipschitz constants and the linear growth constant due to Lemma 1, the starting point and the stepsize). We can use the result in Dubey et al. (2016) applied in the transformed space, which follows similarly to (18), to find that

$$\mathrm{E}|\hat{\delta}_t|^2 \leq \frac{D^2}{Nn}\sum_{i=1}^{N}\mathrm{E}\left|\phi_t^{(N)}-\alpha_t^i\right|^2. \tag{21}$$

Again the Lipschitz constant D^2 is a result of Assumption 1, so does not depend on N. We replace the minibatch size n by its fixed counterpart n_0. Then subbing this into (20) we get

$$\mathrm{E}[|\phi_t-\bar{\phi}_t|^2] \leq A + B\frac{D^2}{Nn_0}\max_{t\in 1,\ldots,t-1}\sum_{i=1}^{N}\mathrm{E}\left|\phi_t^{(N)}-\alpha_t^i\right|^2.$$

Set $\tau = \mathrm{argmax}_t \sum_{i=1}^{N}\mathrm{E}\left|\phi_t^{(N)}-\alpha_t^i\right|^2$, we can then use Cauchy-Schwarz to obtain the following

$$\begin{aligned}
\mathrm{E}[|\phi_t-\bar{\phi}_t|^2] &\leq A + B\frac{D^2}{Nn_0}\sum_{i=1}^{N}\mathrm{E}\left|\phi_\tau^{(N)}-\alpha_\tau^i\right|^2 \\
&\leq A + B\frac{D^2}{Nn_0}\sum_{i=1}^{N}\mathrm{E}\left|\phi_\tau^{(N)}-\hat{\phi}^{(N)}+\hat{\phi}^{(N)}-\alpha_\tau^i\right|^2 \\
&\leq A + B\frac{2D^2}{Nn_0}\sum_{i=1}^{N}\left[\mathrm{E}\left|\phi_\tau^{(N)}-\hat{\phi}^{(N)}\right|^2+\left|\alpha_\tau^i-\hat{\phi}^{(N)}\right|^2\right].
\end{aligned}$$

But α_t^i is a previous state in the SAGA algorithm, call this previous state $\phi_{\alpha_i}^{(N)}$, then we have

$$\begin{aligned}
\mathrm{E}[|\phi_t-\bar{\phi}_t|^2] &\leq A + B\frac{2D^2}{Nn_0}\sum_{i=1}^{N}\left[\mathrm{E}\left|\phi_\tau^{(N)}-\hat{\phi}^{(N)}\right|^2+\left|\phi_{\alpha_i}^{(N)}-\hat{\phi}^{(N)}\right|^2\right] \\
&= A + B\frac{2D^2}{Nn_0}\sum_{i=1}^{N}\left[\mathrm{E}\left|\phi_\tau^{(N)}-\bar{\phi}_\tau^{(N)}+\bar{\phi}_\tau^{(N)}-\hat{\phi}^{(N)}\right|^2+\left|\phi_{\alpha_i}^{(N)}-\bar{\phi}_{\alpha_i}^{(N)}+\bar{\phi}_{\alpha_i}^{(N)}-\hat{\phi}^{(N)}\right|^2\right] \\
&\leq A + B\frac{4D^2}{Nn_0}\sum_{i=1}^{N}\left[\mathrm{E}\left|\phi_\tau^{(N)}-\bar{\phi}_\tau^{(N)}\right|^2+\left|\hat{\phi}^{(N)}-\bar{\phi}_\tau^{(N)}\right|^2+\left|\phi_{\alpha_i}^{(N)}-\bar{\phi}_{\alpha_i}^{(N)}\right|^2+\left|\hat{\phi}^{(N)}-\bar{\phi}_{\alpha_i}^{(N)}\right|^2\right] \\
&\leq A + B\frac{4D^2}{n_0}\left[\nu+H+\nu+H\right] \\
&= A + B\frac{8D^2}{n_0}\left[\nu+H\right]
\end{aligned}$$

the second inequality follows from the Cauchy-Schwarz inequality and the third inequality follows from the inductive hypothesis and Lemma 2. Thus set $\nu = A + B\frac{8D^2}{n_0}[\nu+H]$ and the result is proved for t, thus by induction proved for all $t>0$. Thus pick $n_0 = \frac{8BD^2(\nu+H)}{\nu-A}$, then $\mathrm{E}\left|\phi_t-\bar{\phi}_t\right|^2 \leq \nu$ for all t as required. So specifically, for this choice of n_0, $\mathrm{E}\left|\phi_T-\bar{\phi}_T\right|^2 \leq \nu$. Notice that the bound is slightly weaker than in Theorem 1 and n_0 will need to be set up to four times as large compared with SGLD-CV to reach the desired accuracy. □

A.4 Postprocessing Theorems

A.5 Lemmas

Lemma 3. *Define* $A = \sum_{i=1}^{d} a_i^2$, *and let* $\xi_S(\theta) = \nabla \widehat{\log p(\theta|\mathbf{x})} - \nabla \log p(\theta|\mathbf{x})$ *be the noise in the gradient estimate. Then*

$$\mathrm{E}_{\theta|\mathbf{x}} \left[\mathbf{a} \cdot \xi_S(\theta)\right]^2 \leq A \mathrm{E}_{\theta|\mathbf{x}} \mathrm{E}_S |\xi_S(\theta)|^2.$$

Proof. We can condition on the gradient noise, and then immediately apply the Cauchy-Schwarz inequality to get

$$\begin{aligned}
\mathrm{E}_{\theta|\mathbf{x}} \left[\mathbf{a} \cdot \xi_S(\theta)\right]^2 &= \mathrm{E}_{\theta|\mathbf{x}} \mathrm{E}_S \left[\mathbf{a} \cdot \xi_S(\theta)\right]^2 \\
&\leq \left(\sum_{i=1}^{d} a_i^2\right) \mathrm{E}_{\theta|\mathbf{x}} \mathrm{E}_S \left|\xi_S(\theta)\right|^2
\end{aligned}$$

□

Lemma 4. $\mathrm{E}_{\theta|x} \left[h(\theta)\right]^2$ *is* $O(N)$.

Proof.

$$\begin{aligned}
\mathrm{E}_{\theta|x} \left[h(\theta)\right]^2 &\leq \frac{1}{4} \left(\sum_{i=1}^{d} a_i^2\right) \mathrm{E}_{\theta|\mathbf{x}} \left|\nabla \log p(\theta|\mathbf{x})\right|^2 \\
&= \frac{A}{4} \sum_{j=1}^{d} \mathrm{Var}_{\theta|\mathbf{x}} \left[\partial_{\theta_j} \log p(\theta|\mathbf{x})\right] \\
&= \frac{AN}{4} \sum_{j=1}^{d} I_{jj}(\theta)
\end{aligned}$$

□

A.6 Proof of Theorem 3

We start from the bound in Theorem 6.1 of Mira et al. (2013), stating for some control variate h, the optimal variance reduction R is given by

$$R = \frac{\left(\mathrm{E}_{\theta|\mathbf{x}} \left[g(\theta) h(\theta)\right]\right)^2}{\mathrm{E}_{\theta|\mathbf{x}} \left[h(\theta)\right]^2},$$

so that in our case we have

$$\begin{aligned}
\hat{R} &= \frac{\left(\mathrm{E}_{\theta|\mathbf{x}} \left[g(\theta) \hat{h}(\theta)\right]\right)^2}{\mathrm{E}_{\theta|\mathbf{x}} \left[\hat{h}(\theta)\right]^2} \\
&= \frac{\left(\mathrm{E}_{\theta|\mathbf{x}} \left[g(\theta) h(\theta)\right]\right)^2}{\mathrm{E}_{\theta|\mathbf{x}} \left[h(\theta)\right]^2 + \frac{1}{4} \mathrm{E}_{\theta|\mathbf{x}} \left[\mathbf{a} \cdot \xi_S(\theta)\right]} \\
&= \frac{R}{1 + \frac{\frac{1}{4} \mathrm{E}_{\theta|\mathbf{x}} \left[\mathbf{a} \cdot \xi_S(\theta)\right]}{\mathrm{E}_{\theta|\mathbf{x}} \left[h(\theta)\right]^2}}.
\end{aligned}$$

Then we can apply Lemmas 3, 4 to immediately get the desired result

$$\hat{R} \geq \frac{R}{1 + CN^{-1}\mathrm{E}_{\theta|\mathbf{x}}\mathrm{E}_S|\xi_S(\theta)|}. \tag{22}$$

□