Stochastic gradient Markov chain Monte Carlo

Christopher Nemeth
Department of Mathematics and Statistics, Lancaster University, Lancaster, UK
and
Paul Fearnhead
Department of Mathematics and Statistics, Lancaster University, Lancaster, UK

October 27, 2020

Abstract

Markov chain Monte Carlo (MCMC) algorithms are generally regarded as the gold standard technique for Bayesian inference. They are theoretically well-understood and conceptually simple to apply in practice. The drawback of MCMC is that performing exact inference generally requires all of the data to be processed at each iteration of the algorithm. For large data sets, the computational cost of MCMC can be prohibitive, which has led to recent developments in scalable Monte Carlo algorithms that have a significantly lower computational cost than standard MCMC. In this paper, we focus on a particular class of scalable Monte Carlo algorithms, stochastic gradient Markov chain Monte Carlo (SGMCMC) which utilises data subsampling techniques to reduce the per-iteration cost of MCMC. We provide an introduction to some popular SGMCMC algorithms and review the supporting theoretical results, as well as comparing the efficiency of SGMCMC algorithms against MCMC on benchmark examples. The supporting R code is available online[1].

Keywords: Bayesian inference, Markov chain Monte Carlo, scalable Monte Carlo, stochastic gradients.

[1]https://github.com/chris-nemeth/sgmcmc-review-paper

1 Introduction

The Bayesian approach to modelling data provides a flexible mathematical framework for incorporating uncertainty of unknown quantities within complex statistical models. The Bayesian posterior distribution encodes the probabilistic uncertainty in the model parameters and can be used, for example, to make predictions for new unobserved data. In general, the posterior distribution cannot be integrated analytically and it is therefore necessary to approximate it. Deterministic approximations, such as the Laplace approximation (Bishop 2006, see Section 4.4), variational Bayes (Blei et al. 2017) and expectation-propagation (Minka 2001), aim to approximate the posterior with a simpler tractable distribution (e.g. a normal distribution). These deterministic approximations are often fit using fast optimisation techniques and trade-off exact posterior inference for computational efficiency.

Markov chain Monte Carlo (MCMC) algorithms (Brooks et al. 2011) approximate the posterior distribution with a discrete set of samples generated from a Markov chain whose invariant distribution is the posterior distribution. Simple MCMC algorithms, such as random-walk Metropolis (Metropolis et al. 1953), are easy to apply and only require that the unnormalised density of the posterior can be evaluated point-wise. More efficient MCMC algorithms, which offer faster exploration of the posterior, utilise gradients of the posterior density within the proposal mechanism (Roberts & Tweedie 1996, Neal 2011, Girolami & Calderhead 2011). Under mild conditions, the samples generated from the Markov chain converge to the posterior distribution (Roberts & Rosenthal 2004) and for many popular MCMC algorithms, rates of convergence based on geometric ergodicity have been established (see Meyn et al. 1994, Roberts & Rosenthal 1997, for details).

Whilst MCMC algorithms have the advantage of providing asymptotically exact posterior samples, this comes at the expense of being computationally slow to apply in practice.

This issue is further exacerbated by the demand to store and analyse large-scale data sets and to fit increasingly sophisticated and complex models to these high-dimensional data. For example, scientific fields, such as population genetics (Raj et al. 2014), brain imaging (Andersen et al. 2018) and natural language processing (Yogatama et al. 2014), commonly use a Bayesian approach to data analysis, but the continual growth in the size of the data sets in these fields prevents the use of traditional MCMC methods. Computational challenges such as these have led to recent research interest in scalable Monte Carlo algorithms. Broadly speaking, these new Monte Carlo techniques achieve computational efficiency by either parallelising the MCMC scheme, or by subsampling the data.

If the data can be split across multiple computer cores then the computational challenge of inference can be parallelised, with an MCMC algorithm run on each core to draw samples from a partial posterior that is conditional on only a subset of the full data. The challenge is then to merge these posterior samples from each computer to generate an approximation to the full posterior distribution. It is possible to construct methods to merge samples that are exact if the partial posteriors are Gaussian (Scott et al. 2016); for example with update rules that just depend on the mean and variance for each partial posterior. However, it is hard to quantify the level of approximation such rules introduce due to non-Gaussianity of the partial posteriors. Alternative merging procedures, that aim to be more robust to non-Gaussianity, have also been proposed (Neiswanger et al. 2014, Rabinovich et al. 2015, Minsker et al. 2017, Srivastava et al. 2018, Nemeth & Sherlock 2018), but it is hard to quantify the level of approximation accuracy such merging procedures have in general. Bespoke methods are also needed when interest in the joint posterior of the parameters relates to subsets of the data, or individual data points, for example when inferring clusters (Zuanetti et al. 2019)

Alternatively, rather than using multiple computer cores, a single MCMC algorithm can be used, where only a subsample of the data is evaluated at each iteration (Bardenet et al. 2017). For example, in the Metropolis-Hastings algorithm, the accept-reject step can be approximated with a subset of the full data (Korattikara et al. 2014, Bardenet et al. 2014, Quiroz et al. 2018). Again these methods introduce a trade-off between computational speed-up and accuracy. For some models, it is possible to use subsamples of the data at each iteration with the guarantee of sampling from the true posterior; e.g., continuous-time MCMC methods (Fearnhead et al. 2018, Bierkens et al. 2019, Bouchard-Côté et al. 2018). These exact methods can only be applied if the posterior satisfies strong conditions, e.g. the derivative of the log-posterior density can be globally bounded. To date, these methods have only been successfully applied to relatively simple models, such as logistic regression.

Perhaps the most general and popular class of scalable, subsampling-based algorithms are stochastic gradient MCMC (SGMCMC) methods. These algorithms are derived from diffusion processes which admit the posterior as their invariant distribution. A discrete-time Euler approximation of the diffusion is used for Monte Carlo sampling. Many such methods have been based on the over-damped Langevin diffusion (Roberts & Tweedie 1996). Simulating from the Euler approximation gives the unadjusted Langevin algorithm. Due to the discretisation error, the invariant distribution of unadjusted Langevin algorithm is only an approximation to the posterior; though adding a Metropolis-type correction produces an MCMC sampler with the correct invariant distribution (Besag 1994). Even without the Metropolis correction, the unadjusted Langevin algorithm can be computationally expensive as it involves calculating the gradient of the log-posterior density at each iteration and this involves a sum over the full data. Inspired by stochastic gradient descent (Robbins & Monro 1951), Welling & Teh (2011) proposed the stochastic gradient Langevin algo-

rithm, where the gradient component of the unadjusted Langevin algorithm is replaced by a stochastic approximation calculated on a subsample of the full data. An advantage of SGMCMC over other subsampling-based MCMC techniques, such as piece-wise deterministic MCMC (Fearnhead et al. 2018), is that it can be applied to a broad class of models and, in the simplest case, only requires that the first-order gradient of the log-posterior density can be evaluated point-wise. A drawback of these algorithms is that, while producing consistent estimates (Teh et al. 2016), they converge at a slower rate than traditional MCMC algorithms. In recent years, SGMCMC algorithms have become a popular tool for scalable Bayesian inference, particularly in the machine learning community, and there have been numerous methodological (Ma et al. 2015, Chen et al. 2014, Dubey et al. 2016, Baker et al. 2019*a*) and theoretical developments (Teh et al. 2016, Vollmer et al. 2016, Dalalyan & Karagulyan 2019, Durmus & Moulines 2017) along with new application areas for these algorithms (Balan et al. 2015, Gan et al. 2015, Wang et al. 2015). This paper presents a review of some of the key developments in SGMCMC and highlights some of the opportunities for future research.

This paper is organised as follows. Section 2 introduces the Langevin diffusion and its discrete-time approximation as the basis for SGMCMC. This section also presents theoretical error bounds on the posterior approximation and an illustrative example of stochastic gradient Langevin dynamics on a tractable Gaussian example. In Section 3, we show how the SGMCMC framework has been extended beyond the Langevin diffusion, with many popular SGMCMC algorithms given as special cases. Like many MCMC algorithms, SGMCMC has tuning parameters which affect the efficiency of the algorithm. Standard diagnostics for tuning traditional MCMC algorithms are not appropriate for SGMCMC and Section 4 introduces the kernel Stein discrepancy as a metric for both tuning and as-

sessing convergence of SGMCMC algorithms. Section 5 reviews some of the recent work on extending SGMCMC to new settings beyond the case where data are independent and the model parameters are continuous on the real space. A simulation study is given in Section 6, where several SGMCMC algorithms are compared against traditional MCMC methods to illustrate the trade-off between speed and accuracy. Finally, Section 7 concludes with a discussion of the main points in the paper and highlights some areas for future research.

2 Langevin-based Stochastic Gradient MCMC

2.1 The Langevin Diffusion

We are interested in sampling from a target density $\pi(\boldsymbol{\theta})$, where we assume $\boldsymbol{\theta} \in \mathbb{R}^d$ and the unnormalised density is of the form,

$$\pi(\boldsymbol{\theta}) \propto \exp\{-U(\boldsymbol{\theta})\}, \tag{1}$$

and defined in terms of a *potential function* $U(\boldsymbol{\theta})$. We will assume that $U(\boldsymbol{\theta})$ is continuous and differentiable almost everywhere, which are necessary requirements for the methods we discuss in this paper. In our motivating applications from Bayesian analysis for big data, the potential will be defined as a sum over data points. For example, if we have independent data, $y_1, \ldots, y_N$ then $\pi(\boldsymbol{\theta}) \propto p(\boldsymbol{\theta}) \prod_{i=1}^N f(y_i|\boldsymbol{\theta})$, where $p(\boldsymbol{\theta})$ is the prior density and $f(y_i|\boldsymbol{\theta})$ is the likelihood for the ith observation. In this setting, we can define $U(\boldsymbol{\theta}) = \sum_{i=1}^N U_i(\boldsymbol{\theta})$, where $U_i(\boldsymbol{\theta}) = -\log f(y_i|\boldsymbol{\theta}) - (1/N) \log p(\boldsymbol{\theta})$.

One way to generate samples from $\pi(\boldsymbol{\theta})$ is to simulate a stochastic process that has π as its stationary distribution. If we sample from such a process for a long time period and throw away the samples we generate during an initial burn-in period, then the remaining

samples will be approximately distributed as π. The quality of the approximation will depend on how fast the stochastic process converges to its stationary distribution from the initial point, relative to the length of the burn-in period. The most common example of such an approach to sampling is MCMC (Hastings 1970, Dunson & Johndrow 2020).

Under mild regularity conditions (Roberts & Tweedie 1996, Pillai et al. 2012), the Langevin diffusion, defined by the stochastic differential equation

$$\mathrm{d}\boldsymbol{\theta}(t) = -\frac{1}{2}\nabla U(\boldsymbol{\theta}(t))\mathrm{d}t + \mathrm{d}B_t, \tag{2}$$

where $\nabla U(\boldsymbol{\theta}(t))$ is the drift term and B_t denotes d-dimensional Brownian motion, has π as its stationary distribution. This equation can be interpreted as defining the dynamics of a continuous-time Markov process over infinitesimally-small time intervals. That is, for a small time-interval $h > 0$, the Langevin diffusion has approximate dynamics given by

$$\boldsymbol{\theta}(t+h) \approx \boldsymbol{\theta}(t) - \frac{h}{2}\nabla U(\boldsymbol{\theta}(t)) + \sqrt{h}\mathbf{Z}, \quad k = 0, \ldots, K \tag{3}$$

where $\mathbf{Z}$ is a vector of d independent standard Gaussian random variables.

The dynamics implied by (3) give a simple recipe to approximately sample from the Langevin diffusion. To do so over a time period of length $T = Kh$, for some integer K, we just set $\boldsymbol{\theta}_0$ to be the initial state of the process and repeatedly simulate from (3) to obtain values of the process at times $h, 2h, \ldots, Kh$. In the following, when using such a scheme we will use the notation $\boldsymbol{\theta}_k$ to denote the state at time kh. If we are interested in sampling from the Langevin diffusion at some fixed time T, then the Euler discretisation will become more accurate as we decrease h; and we can achieve any required degree of accuracy if we choose h small enough. However, it is often difficult in practice to know when h is small enough, see Section 4 for more discussion of this.

2.2 Approximate MCMC using the Langevin Diffusion

As the Langevin diffusion has π as its stationary distribution, it is natural to consider this stochastic process as a basis for an MCMC algorithm. In fact, if it were possible to simulate exactly the dynamics of the Langevin diffusion, then we could use the resulting realisations at a set of discrete time-points as our MCMC output. However, for general $\pi(\boldsymbol{\theta})$ the Langevin dynamics are intractable, and in practice people often resort to using samples generated by the Euler approximation (3).

This is most commonly seen with the Metropolis-adjusted Langevin Algorithm, or MALA (Roberts & Tweedie 1996). This algorithm uses the Euler approximation (3) over an appropriately chosen time-interval, h, to define the proposal distribution of a standard Metropolis-Hastings algorithm. The simulated value is then either accepted or rejected based on the Metropolis-Hastings acceptance probability. Such an algorithm has good theoretical properties, and in particular, can scale better to high-dimensional problems than the simpler random walk MCMC algorithm (Roberts & Rosenthal 1998, 2001).

A simpler algorithm is the unadjusted Langevin algorithm, also known as ULA (Parisi 1981, Ermak 1975), which simulates from the Euler approximation but does not use a Metropolis accept-reject step and so the MCMC output produces a biased approximation of π. Computationally, such an algorithm is quicker per-iteration, but often this saving is small, as the $O(N)$ cost of calculating $\nabla U(\boldsymbol{\theta})$, which is required for one step of the Euler approximation, is often at least as expensive as the cost of the accept-reject step. Furthermore, the optimal step size for MALA is generally large, resulting in a poor Euler approximation to the Langevin dynamics – and so ULA requires a smaller step size, and potentially many more iterations. One advantage that ULA has is that its performance is more robust to poor initialisations; by comparison a well-tuned MALA algorithm often has

a high rejection probability if initialised in the tail of the posterior.

The computational bottleneck for ULA is in calculating $\nabla U(\boldsymbol{\theta})$, particularly if we have a large sample size, N, as $U(\boldsymbol{\theta}) = \sum_{i=1}^{N} U_i(\boldsymbol{\theta})$. A solution to this problem is to use stochastic gradient Langevin dynamics (SGLD Welling & Teh 2011), which avoids calculating $\nabla U(\boldsymbol{\theta})$, and instead uses an unbiased estimate at each iteration. It is trivial to obtain an unbiased estimate using a random subsample of the terms in the sum. The simplest implementation is to choose $n << N$ and estimate $\nabla U(\boldsymbol{\theta})$ with

$$\hat{\nabla} U(\boldsymbol{\theta})^{(n)} = \frac{N}{n} \sum_{i \in \mathcal{S}_n} \nabla U_i(\boldsymbol{\theta}), \tag{4}$$

where $\mathcal{S}_n$ is a random sample, without replacement, from $\{1, \ldots, N\}$. We call this the simple estimator of the gradients, and use the superscript (n) to denote the subsample size used in constructing our estimator. The resulting SGLD is given in Algorithm 1, and allows for the setting where the step size of the Euler discretisation depends on iteration number. Welling & Teh (2011) justified the SGLD algorithm by giving an informal argument that if the step size decreases to zero with iteration number, then it will converge to the true Langevin dynamics, and hence be exact; see Section 2.4 for a formal justification of this.

The advantage of SGLD is that, if $n << N$, the per-iteration cost of the algorithm can be much smaller than either MALA or ULA. For large data applications, SGLD has been empirically shown to perform better than standard MCMC when there is a fixed computational budget (Ahn et al. 2015, Li et al. 2016). In challenging examples, performance has been based on measures of predictive accuracy on held-out test data, rather than based on how accurately the samples approximate the true posterior. Furthermore, the conclusions from such studies will clearly depend on the computational budget, with larger budgets favouring exact methods such as MALA – see the theoretical results in Section 2.4.

Algorithm 1: SGLD

Input: $\boldsymbol{\theta}_0$, $\{h_0, \ldots, h_K\}$.

for $k \in 1, \ldots, K$ **do**

Draw $\mathcal{S}_n \subset \{1, \ldots, N\}$ without replacement

Estimate $\hat{\nabla} U(\boldsymbol{\theta})^{(n)}$ using (4)

Draw $\xi_k \sim N(0, h_k I)$

Update $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k - \frac{h_k}{2} \hat{\nabla} U(\boldsymbol{\theta}_k)^{(n)} + \xi_k$

end

The SGLD algorithm is closely related to stochastic gradient descent (SGD) (Robbins & Monro 1951), an efficient algorithm for finding local maxima of a function. The only difference is the inclusion of the additive Gaussian noise at each iteration of SGLD. Without the noise, but with a suitably decreasing step size, stochastic gradient descent would converge to a local maxima of the density $\pi(\boldsymbol{\theta})$. Again, SGLD has been shown empirically to out-perform stochastic gradient descent (Chen et al. 2014) at least in terms of prediction accuracy – intuitively this is because SGLD will give samples around the estimate obtained by stochastic gradient descent and thus can average over the uncertainty in the parameters. This strong link between SGLD and stochastic gradient descent, and the good peformance the stochastic gradient descent often has for prediction, may also explain why the former performs well when compared to exact MCMC methods in terms of prediction accuracy.

2.3 Estimating the Gradient

A key part of SGLD is replacing the true gradient with an estimate. The more accurate this estimator is, the better we would expect SGLD to perform, and thus it is natural to

consider alternatives to the simple estimator (4).

One way of reducing the variance of a Monte Carlo estimator is to use control variates (Ripley 1987), which in our setting involves choosing a set of simple functions u_i, $i = 1, \ldots, N$, whose sum $\sum_{i=1}^{N} u_i(\boldsymbol{\theta})$ is known for any $\boldsymbol{\theta}$. As

$$\sum_{i=1}^{N} \nabla U_i(\boldsymbol{\theta}) = \sum_{i=1}^{N} u_i(\boldsymbol{\theta}) + \sum_{i=1}^{N} \left(\nabla U_i(\boldsymbol{\theta}) - u_i(\boldsymbol{\theta})\right),$$

we can obtain the unbiased estimator $\sum_{i=1}^{N} u_i(\boldsymbol{\theta}) + (N/n) \sum_{i \in \mathcal{S}_n} (\nabla U_i(\boldsymbol{\theta}) - u_i(\boldsymbol{\theta}))$, where again $\mathcal{S}_n$ is a random sample, without replacement, from $\{1, \ldots, N\}$. The intuition behind this idea is that if each $u_i(\boldsymbol{\theta}) \approx \nabla U_i(\boldsymbol{\theta})$, then this estimator can have a much smaller variance.

Recent works, for example Baker et al. (2019*a*) and Huggins & Zou (2017) (see Bardenet et al. 2017, Pollock et al. 2020, Bierkens et al. 2019, for similar ideas used in different Monte Carlo procedures), have implemented this control variate technique with each $u_i(\boldsymbol{\theta})$ set as a constant. These approaches propose (i) using stochastic gradient descent to find an approximation to the mode of the distribution we are sampling from, which we denote as $\hat{\boldsymbol{\theta}}$; and (ii) set $u_i(\boldsymbol{\theta}) = \nabla U_i(\hat{\boldsymbol{\theta}})$. This leads to the following control variate estimator,

$$\hat{\nabla}_{cv} U(\boldsymbol{\theta})^{(n)} = \sum_{i=1}^{N} \nabla U_i(\hat{\boldsymbol{\theta}}) + \frac{N}{n} \sum_{i \in \mathcal{S}_n} \left(\nabla U_i(\boldsymbol{\theta}) - \nabla U_i(\hat{\boldsymbol{\theta}})\right).$$

Implementing such an estimator involves an up-front of cost of finding a suitable $\hat{\boldsymbol{\theta}}$ and calculating, storing and summing $\nabla U_i(\hat{\boldsymbol{\theta}})$ for $i = 1, \ldots, N$. Of these, the main cost is finding a suitable $\hat{\boldsymbol{\theta}}$. Though we can then use $\hat{\boldsymbol{\theta}}$ as a starting value for the SGLD algorithm, replacing $\boldsymbol{\theta}_0$ with $\hat{\boldsymbol{\theta}}$ in Algorithm 1, which can significantly reduce the burn-in phase (see Figure 2 for an illustration).

The advantage of using this estimator can be seen if we compare bounds on the variance of this and the simple estimator. If $\nabla U_i(\boldsymbol{\theta})$ and its derivatives are bounded for all i and $\boldsymbol{\theta}$, then there are constants C_1 and C_2 such that

$$\text{Var}\left[\hat{\nabla} U(\boldsymbol{\theta})^{(n)}\right] \leq C_1 \frac{N^2}{n}, \quad \text{Var}\left[\hat{\nabla}_{cv} U(\boldsymbol{\theta})^{(n)}\right] \leq C_2 ||\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}||^2 \frac{N^2}{n},$$

where $||\cdot||$ denotes Euclidean distance. Thus, when $\boldsymbol{\theta}$ is close to $\hat{\boldsymbol{\theta}}$, we would expect the latter variance to be smaller. Furthermore, in many settings when N is large we would expect a value of $\boldsymbol{\theta}$ drawn from the target to be of distance $O(N^{-1/2})$, thus using control variates will reduce the variance from $O(N^2/n)$ to $O(N/n)$. This simple argument suggests that, for the same level of accuracy, we can reduce the computational cost of SGLD by $O(N)$ if we use control variates. This is supported by a number of theoretical results (e.g. Nagapetyan et al. 2017, Baker et al. 2019*a*, Brosse et al. 2018) which show that, if we ignore the pre-processing cost of finding $\hat{\boldsymbol{\theta}}$, the computational cost per-effective sample size of SGLD with control variates has a computational cost that is $O(1)$, rather than the $O(N)$ for SGLD with the simple gradient estimator (4).

A further consequence of these bounds on the variance is that they suggest that if $\boldsymbol{\theta}$ is far from $\hat{\boldsymbol{\theta}}$ then the variance of using control variates can be larger, potentially substantially larger, than that of the simple estimator. Two ways have been suggested to deal with this. One is to only use the control variate estimator when $\boldsymbol{\theta}$ is close enough to $\hat{\boldsymbol{\theta}}$ (Fearnhead et al. 2018), though it is up to the user to define what "close enough" is in practice. The second is to update $\hat{\boldsymbol{\theta}}$ during SGLD. This can be done efficiently by using $u_i(\boldsymbol{\theta}) = \nabla U_i(\boldsymbol{\theta}_{k_i})$, where $\boldsymbol{\theta}_{k_i}$ is the value of $\boldsymbol{\theta}$ at the most recent iteration of the SGLD algorithm where $\nabla U_i(\boldsymbol{\theta})$ was evaluated (Dubey et al. 2016). This involves updating the storage of $u_i(\boldsymbol{\theta})$ and its sum at each iteration; importantly the latter can be done with an $O(n)$ calculation. A further possibility, which we are not aware has yet been tried, is to use $u_i(\boldsymbol{\theta})$ that are non-constant,

and thus try to accurately estimate $\nabla U_i(\boldsymbol{\theta})$ for a wide range of $\boldsymbol{\theta}$ values.

Another possibility for reducing the variance of the estimate of $\nabla U(\boldsymbol{\theta})$ is to use preferential sampling. If we generate a sample, $\mathcal{S}_n$, such that the expected number of times i appears is w_i, then we could use the unbiased estimator

$$\hat{\nabla}_w U(\boldsymbol{\theta})^{(n)} = \sum_{i \in \mathcal{S}_n} \frac{\nabla U_i(\boldsymbol{\theta})}{w_i}.$$

The simple estimator (4) is a special case of this estimator where $w_i = n/N$ for all i. This weighted estimator can have a lower variance if we choose larger w_i for $\nabla U_i(\boldsymbol{\theta})$ values that are further from the mean value. A natural situation where such an estimator would make sense would be if we have data from a small number of cases and many more controls, where giving larger weights to the cases is likely to reduce the variance. Similarly, if we have observations that vary in their information about the parameters, then giving larger weights to more informative observations would make sense. Note that using weighted sampling can be combined with the control variate estimator – with a natural choice of weights that are increasing with the size of the derivative of $\nabla U_i(\boldsymbol{\theta})$ at $\hat{\boldsymbol{\theta}}$. We can also use stratified sampling ideas, which try to ensure each subsample is representative of the full data (Sen et al. 2020), or adapt ideas from stochastic optimisation that uses multi-arm bandits to learn a good sampling distribution (Salehi et al. 2017).

Regardless of the choice of gradient estimator, an important question is how large should the subsample size be? A simple intuitive rule, which has some theoretical support (e.g. Vollmer et al. 2016, Nagapetyan et al. 2017), is to choose the subsample size such that if we consider one iteration of SGLD, the variance of the noise from the gradient term is dominated by the variance of the injected noise. As the former scales like h^2 and the latter like h then this suggests that as we reduce the step size, h, smaller subsample sizes could be used – see Section 2.5 for more details.

2.4 Theory for SGLD

As described so far, SGLD is a simple and computationally efficient approach to approximately sample from a stochastic process whose asymptotic distribution is π; but how well do samples from SGLD actually approximate π? In particular, whilst for small step sizes the approximation within one iteration of SGLD may be good, do the errors from these approximations accumulate over many iterations? There is now a body of theory addressing these questions. Here we give a brief, and informal overview of this theory. We stress that all results assume a range of technical conditions on $\pi(\boldsymbol{\theta})$, some of which are strong – see the original references for details. In particular, most results assume that the drift of the underlying Langevin diffusion will push $\boldsymbol{\theta}$ towards the centre of the distribution, an assumption which means that the underlying Langevin diffusion will be geometrically ergodic, and an assumption that is key to avoid the accumulation of error within SGLD.

There are various ways of measuring accuracy of SGLD, but current theory focuses on two approaches. The first considers estimating the expectation of a suitable test function $\phi(\boldsymbol{\theta})$, i.e. $\mathbb{E}_\pi[\phi(\boldsymbol{\theta})] = \int \pi(\boldsymbol{\theta})\phi(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$, using an average over the output from K iterations of SGLD, $(1/K)\sum_{k=1}^{K}\phi(\boldsymbol{\theta}_k)$. In this setting, we can measure the accuracy of the SGLD algorithm through the mean square error of this estimator. Teh et al. (2016) consider this in the case where the SGLD step size h_k decreases with k. The mean square error of the estimator can be partitioned into a square bias term and a variance term. For large K, the bias term increases with the step size, whereas the variance term is decreasing. Teh et al. (2016) show that in terms of minimising the asymptotic mean square error, the optimal choice of step size should decrease as $k^{-1/3}$, with the resulting mean square error of the estimator decaying as $K^{-2/3}$. This is slower than for standard Monte Carlo procedures, where a Monte Carlo average based on K samples will have mean square error that decays

as K^{-1}. The slower rate comes from needing to control the bias as well as the variance, and is similar to rates seen for other Monte Carlo problems where there are biases that need to be controlled (e.g. Section 3.3 of Fearnhead et al. 2008). In practice, SGLD is often implemented with a fixed step size h. Vollmer et al. (2016) give similar results on the bias-variance trade-off for SGLD with a fixed step size, with a mean square error for K iterations and a step size of h being $O(h^2 + 1/(hK))$. The h^2 term comes from the squared bias and $1/hK$ from the variance term. The rate-optimal choice of h as a function of K is $K^{-1/3}$, which again gives an asymptotic mean square error that is $O(K^{-2/3})$; the same asymptotic rate as for the decreasing step size. This result also shows that with larger computational budgets we should use smaller step sizes. Furthermore, if we have a large enough computational resource then we should prefer exact MCMC methods over SGLD: as computing budget increases, exact MCMC methods will eventually be more accurate.

The second class of results consider the distribution that SGLD samples from at iteration K with a given initial distribution and step size. Denoting the density of $\boldsymbol{\theta}_K$ by $\tilde{\pi}_K(\boldsymbol{\theta})$, one can the measure an appropriate distance between $\tilde{\pi}_K(\boldsymbol{\theta})$ and $\pi(\boldsymbol{\theta})$. The most common distance used is the Wasserstein distance (Gibbs & Su 2002), primarily because it is particularly amenable to analysis. Care must be taken when interpreting the Wasserstein distance, as it is not scale invariant – so changing the units of our parameters will result in a corresponding scaling of the Wasserstein distance between the true posterior and the approximation we sample from. Furthermore, as we increase the dimension of the parameters, d, and maintain the same accuracy for the marginal posterior of each component, the Wasserstein distance will scale like $d^{1/2}$.

There are a series of results for both ULA and SGLD in the literature (Dalalyan 2017, Dalalyan & Karagulyan 2019, Durmus & Moulines 2017, Chatterji et al. 2018, Brosse et al.

2018). Most of this theory assumes strong-convexity of the log-target density (see Raginsky et al. 2017, Majka et al. 2020, for similar theory under different assumptions), which means that there exists strictly positive constants, $0 < m \leq M$, such that for all $\boldsymbol{\theta}$, and $\boldsymbol{\theta}'$,

$$||\nabla U(\boldsymbol{\theta}) - \nabla U(\boldsymbol{\theta}')||_2 \leq M||\boldsymbol{\theta} - \boldsymbol{\theta}'||_2, \text{ and } U(\boldsymbol{\theta}) - U(\boldsymbol{\theta}') - \nabla U(\boldsymbol{\theta}')^\top(\boldsymbol{\theta} - \boldsymbol{\theta}') \geq \frac{m}{2}||\boldsymbol{\theta} - \boldsymbol{\theta}'||_2^2, \tag{5}$$

where $||\cdot||_2$ denotes the Euclidean norm. If $U(\boldsymbol{\theta})$ is twice continuously differentiable, these conditions are equivalent to assuming upper and lower bounds on all possible directional derivatives of $U(\boldsymbol{\theta})$. The first bound governs how much the drift of the Langevin diffusion can change, and is important in the theory for specifying appropriate step-lengths, which should be less than $1/M$, to avoid instability of the Euler discretisation; it also ensures that the target density is uni-modal. The second bound ensures that the drift of the Langevin will push $\boldsymbol{\theta}$ towards the centre of the distribution, an assumption which means that the underlying Langevin diffusion will be geometrically ergodic, and consequently is key to avoiding the accumulation of error within SGLD.

For simplicity, we will only informally present results from Dalalyan & Karagulyan (2019), as these convey the main ideas in the literature. These show that, for $h < 1/(M + m)$, the Wasserstein-2 distance between $\tilde{\pi}_K(\boldsymbol{\theta})$ and $\pi(\boldsymbol{\theta})$, denoted by $\mathcal{W}_2(\tilde{\pi}_K, \pi)$ can be bounded as

$$\mathcal{W}_2(\tilde{\pi}_K, \pi) \leq (1 - mh)^K \mathcal{W}_2(\tilde{\pi}_0, \pi) + C_1(hd)^{1/2} + C_2\sigma(hd)^{1/2}, \tag{6}$$

where m, C_1 and C_2 are constants, d is the dimension of $\boldsymbol{\theta}$, and σ^2 is a bound on the variance of the estimate for the gradient. Setting $\sigma^2 = 0$ gives a Wasserstein bound for the ULA approximation. The first term on the right-hand side measures the bias due to starting the SGLD algorithm from a distribution that is not π, and is akin to the bias due to finite

burn-in of the MCMC chain. Providing h is small enough, this will decay exponentially with K. The other two terms are, respectively, the effects of the approximations from using an Euler discretisation of the Langevin diffusion and an unbiased estimate of $\nabla U(\boldsymbol{\theta})$.

A natural question is, what do we learn from results such as (6)? These results give theoretical justification for using SGLD, and show we can sample from an arbitrarily good approximation to our posterior distribution if we choose K large enough, and h small enough. They have also been used to show the benefits of using control variates when estimating the gradient, which results in a computational cost that is $O(1)$, rather than $O(N)$, per effective sample size Baker et al. (2019*a*), Chatterji et al. (2018). Perhaps the main benefit of results such as (6) is that they enable us to compare the properties of the different variants of SGLD that we will introduce in Section 3, and in particular how different algorithms scale with dimension, d (see Section 3 for details). However, they only tell us how these hyper-parameters need to scale with different factors, (e.g., smoothness and dimension), with no specific guidance on the constants in front of those factors.

Perhaps more importantly than having a quantitative measure of approximation error is to have an idea as to the form of the error that the approximations in SGLD induce. Results from Vollmer et al. (2016) and Brosse et al. (2018), either for specific examples or for the limiting case of large N, give insights into this. For an appropriately implemented SGLD algorithm, and for large data size N, these results show that the distribution we sample from will asymptotically have the correct mode but will inflate the variance. We discuss ways to alleviate this in the next section when we consider a specific example.

2.5 A Gaussian Example

To gain insight into the properties of SGLD, it is helpful to consider a simple tractable example where we sample from a Gaussian target. We will consider a 2-dimensional Gaussian, with variance $\mathbf{\Sigma}$ and, without loss of generality, mean zero. The variance matrix can be written as $\mathbf{P}^\top \mathbf{D} \mathbf{P}$ for some rotation matrix $\mathbf{P}$ and diagonal matrix $\mathbf{D}$, whose entries satisfy the condition $\sigma_1^2 \geq \sigma_2^2$. For this model, the drift term of the Langevin diffusion is

$$\nabla U(\boldsymbol{\theta}) = -\mathbf{\Sigma}^{-1}\boldsymbol{\theta} = -\mathbf{P}^\top \mathbf{D}^{-1}\mathbf{P}\boldsymbol{\theta}.$$

The kth iteration of the SGLD algorithm is

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \frac{h}{2}\hat{\nabla} U(\boldsymbol{\theta}_{k-1}) + \sqrt{h}\mathbf{Z} = \boldsymbol{\theta}_{k-1} - \frac{h}{2}\mathbf{P}^\top \mathbf{D}^{-1}\mathbf{P}\boldsymbol{\theta}_{k-1} + h\boldsymbol{\nu}_k + \sqrt{h}\mathbf{Z}_k, \tag{7}$$

where $\mathbf{Z}_k$ is a vector of two independent standard normal random variables and $\boldsymbol{\nu}_k$ is the error in our estimate of $\nabla U(\boldsymbol{\theta}_{k-1})$. The entries of $\mathbf{D}^{-1}$ correspond to the constants that appear in condition (5), with $m = 1/\sigma_1^2$ and $M = 1/\sigma_2^2$.

To simplify the exposition, it is helpful to study the SGLD algorithm for the transformed state $\tilde{\boldsymbol{\theta}} = \mathbf{P}\boldsymbol{\theta}$, for which we have

$$\tilde{\boldsymbol{\theta}}_k = \tilde{\boldsymbol{\theta}}_{k-1} - \frac{h}{2}\mathbf{D}^{-1}\tilde{\boldsymbol{\theta}}_{k-1} + h\mathbf{P}\boldsymbol{\nu} + \sqrt{h}\mathbf{P}\mathbf{Z} = \begin{pmatrix} 1 - h/(2\sigma_1^2) & 0 \\ 0 & 1 - h/(2\sigma_2^2) \end{pmatrix} \tilde{\boldsymbol{\theta}}_{k-1} + h\mathbf{P}\boldsymbol{\nu}_k + \sqrt{h}\mathbf{P}\mathbf{Z}_k.$$

As $\mathbf{P}$ is a rotation matrix, the variance of $\mathbf{P}\mathbf{Z}_k$ is still the identity.

In this case, the SGLD update is a vector auto-regressive process. This process will have a stationary distribution provided $h < 4\sigma_2^2 = 4/M$, otherwise the process will have trajectories that will go to infinity in at least one component. This links to the requirement of a bound on the step size that is required in the theory for convex target distributions described above.

Now assume $h < 2\sigma_2^2$, and write $\lambda_j = h/(2\sigma_j^2) < 1$. We have the following dynamics for each component, $j = 1, 2$

$$\tilde{\boldsymbol{\theta}}_k^{(j)} = (1-\lambda_j)^k \tilde{\boldsymbol{\theta}}_0^{(j)} + \sum_{i=1}^{k} (1-\lambda_j)^{k-i} \left(h(\mathbf{P}\boldsymbol{\nu}_i)^{(j)} + \sqrt{h}(\mathbf{P}\mathbf{Z}_i)^{(j)} \right), \tag{8}$$

where $\tilde{\boldsymbol{\theta}}_k^{(j)}$ is the jth component of $\tilde{\boldsymbol{\theta}}_k$, and similar notation is used for $\boldsymbol{\nu}_i$ and $\mathbf{Z}_i$. From this, we immediately see that SGLD forgets its initial condition exponentially quickly. However, the rate of exponential decay is slower for the component with larger marginal variance, σ_1^2. Furthermore, as the size of h is constrained by the smaller marginal variance σ_2^2, this rate will necessarily be slow if $\sigma_2^2 << \sigma_1^2$; this suggests that there are benefits of re-scaling the target so that marginal variances of different components are roughly equal.

Taking the expectation of (8) with respect to $\boldsymbol{\nu}$ and $\mathbf{Z}$, and letting $k \to \infty$, results in SGLD dynamics that have the correct limiting mean but with an inflated variance. This is most easily seen if we assume that the variance of $\mathbf{P}\boldsymbol{\nu}$ is independent of position, $\mathbf{V}$ say. In this case, the stationary distribution of SGLD will have variance

$$\mathrm{Var}_{\tilde{\pi}}\left[\tilde{\boldsymbol{\theta}}\right] = \begin{pmatrix} (1-(1-\lambda_1)^2)^{-1} & 0 \\ 0 & (1-(1-\lambda_2)^2)^{-1} \end{pmatrix} (h^2\mathbf{V} + h\mathbf{I}),$$

where $\mathbf{I}$ is the identity matrix. The marginal variance for component j is thus

$$\sigma_j^2 \frac{1 + h\mathbf{V}_{jj}}{1 - h/(4\sigma_j^2)} = \sigma_j^2 \left(1 + h\mathbf{V}_{jj}\right) + \frac{h}{4} + O(h^2).$$

The inflation in variance comes both from the noise in the estimate of $\nabla U(\boldsymbol{\theta})$, which is the $h\mathbf{V}_{jj}$ factor, and the Euler approximation, through the additive constant, $h/4$. For more general target distributions, the mean of the stationary distribution of SGLD will not necessarily be correct, but we would expect the mean to be more accurate than the variance,

with the variance of SGLD being greater than that of the true target. The above analysis further suggests that, for targets that are close to Gaussian, it may be possible to perform a better correction to compensate for the inflation of the variance. Vollmer et al. (2016) suggest reducing the driving Brownian noise (see also Chen et al. 2014). That is, we replace $\mathbf{Z}_k$ by Gaussian random variables with a covariance matrix so that the covariance matrix of $h\boldsymbol{\nu}_k + \sqrt{h}\mathbf{Z}$ is the identity. If the variance of $\boldsymbol{\nu}_k$ is known, then Vollmer et al. (2016) show that this can substantially improve the accuracy of SGLD. In practice, however, it is necessary to estimate this variance and it is an open problem as to how one can estimate this accurately enough to make the idea work well in practice (Vollmer et al. 2016). As suggested by a reviewer, an alternative is to estimate the rough size of the variance of $\boldsymbol{\nu}_k$ and use this to guide the choice of h so that the impact of the stochastic gradient would be below some acceptable tolerance.

3 A General Framework for Stochastic Gradient MCMC

So far we have considered SGMCMC based on approximating the dynamics of the Langevin diffusion. However, we can write down other diffusion processes that have π as their stationary distribution, and use similar ideas to approximately simulate from one of these. A general approach to doing this was suggested by Ma et al. (2015) and leads to a much wider class of SGMCMC algorithms, including stochastic gradient versions of popular MCMC algorithms such as Hamiltonian Monte Carlo (Neal 2011, Carpenter et al. 2017).

The class of diffusions we will consider may include a set of auxiliary variables. As such, we let $\boldsymbol{\zeta}$ be a general state, with the assumption that this state contains $\boldsymbol{\theta}$. For example, for the Langevin diffusion $\boldsymbol{\zeta} = \boldsymbol{\theta}$; but we could mimic Hamiltonian MCMC and introduce

an auxiliary velocity component, $\boldsymbol{\rho}$, in which case $\boldsymbol{\zeta} = (\boldsymbol{\theta}, \boldsymbol{\rho})$. We start by considering a general stochastic differential equation for $\boldsymbol{\zeta}$,

$$\mathrm{d}\boldsymbol{\zeta} = \frac{1}{2}\mathbf{b}(\boldsymbol{\zeta})\mathrm{d}t + \sqrt{\mathbf{D}(\boldsymbol{\zeta})}\mathrm{d}B_t, \tag{9}$$

where the vector $\mathbf{b}(\boldsymbol{\zeta})$ is the drift component, $\mathbf{D}(\boldsymbol{\zeta})$ is a positive semi-definite diffusion matrix, and $\sqrt{\mathbf{D}(\boldsymbol{\zeta})}$ is any square-root of $\mathbf{D}(\boldsymbol{\zeta})$. Ma et al. (2015) show how to choose $\mathbf{b}(\boldsymbol{\zeta})$ and $\mathbf{D}(\boldsymbol{\zeta})$ such that (9) has a specific stationary distribution. We define the function $H(\boldsymbol{\zeta})$ such that $\exp\{-H(\boldsymbol{\zeta})\}$ is intergrable and let $\mathbf{Q}(\boldsymbol{\zeta})$ be a skew-symmetric curl matrix, so $\mathbf{Q}^\top = -\mathbf{Q}$. Then the choice

$$\mathbf{b}(\boldsymbol{\zeta}) = -\left[\mathbf{D}(\boldsymbol{\zeta}) + \mathbf{Q}(\boldsymbol{\zeta})\right]\nabla H(\boldsymbol{\zeta}) + \Gamma(\boldsymbol{\zeta}) \quad \text{and} \quad \Gamma_i(\boldsymbol{\zeta}) = \sum_{j=1}^{d} \frac{\partial}{\partial \boldsymbol{\zeta}_j}(\mathbf{D}_{ij}(\boldsymbol{\zeta}) + \mathbf{Q}_{ij}(\boldsymbol{\zeta})), \tag{10}$$

ensures that the stationary distribution of (9) is proportional to $\exp\{-H(\boldsymbol{\zeta})\}$. Ma et al. (2015) show that any diffusion process with a stationary distribution proportional to $\exp\{-H(\boldsymbol{\zeta})\}$ is of the form (9) with the drift and diffusion matrix satisfying (10). To approximately sample from our diffusion, we can employ the same discretisation of the continuous-time dynamics that we used for the Langevin diffusion (3),

$$\boldsymbol{\zeta}_{t+h} \approx \boldsymbol{\zeta}_t - \frac{h}{2}\left[(\mathbf{D}(\boldsymbol{\zeta}_t) + \mathbf{Q}(\boldsymbol{\zeta}_t))\nabla H(\boldsymbol{\zeta}_t) + \Gamma(\boldsymbol{\zeta}_t)\right] + \sqrt{h}\mathbf{Z}, \quad t \geq 0, \tag{11}$$

where $\mathbf{Z} \sim N(0, \mathbf{D}(\boldsymbol{\zeta}_t))$. The diffusions we are interested in have a stationary distribution where the $\boldsymbol{\theta}$-marginal distribution is π. If $\boldsymbol{\zeta} = \boldsymbol{\theta}$ then this requires $H(\boldsymbol{\zeta}) = U(\boldsymbol{\theta})$. If, however, $\boldsymbol{\zeta}$ also includes some auxiliary variables, say $\boldsymbol{\rho}$, then this is most easily satisfied by setting $H(\boldsymbol{\zeta}) = U(\boldsymbol{\theta}) + K(\boldsymbol{\rho})$ for some suitable function $K(\boldsymbol{\rho})$. This choice leads to a stationary distribution under which $\boldsymbol{\theta}$ and $\boldsymbol{\rho}$ are independent.

We can derive a general class of SGMCMC algorithms, where we simply replace the gradient estimate $\nabla H(\boldsymbol{\zeta}_t)$ with an unbiased estimate $\hat{\nabla} H(\boldsymbol{\zeta}_t)$, based on data subsampling.

Ma et al. (2015) suggest that one should also correct for the variance of the estimate of the gradient, as illustrated in the example from Section 2.5, to avoid the inflation of variance in the approximate target distribution. If the variance of our estimator $\hat{\nabla} H(\boldsymbol{\zeta}_t)$ is $\mathbf{V}(\boldsymbol{\theta}_t)$, then this inflates the conditional variance of $\boldsymbol{\zeta}_{t+h}$ given $\boldsymbol{\zeta}_t$ in (11) by $h^2\mathbf{B}(\boldsymbol{\zeta}_t)$ where

$$\mathbf{B}(\boldsymbol{\zeta}_t) = \frac{1}{4}(\mathbf{D}(\boldsymbol{\zeta}_t) + \mathbf{Q}(\boldsymbol{\zeta}_t))\mathbf{V}(\boldsymbol{\theta}_t)(\mathbf{D}(\boldsymbol{\zeta}_t) + \mathbf{Q}(\boldsymbol{\zeta}_t))^\top.$$

Given an estimate $\hat{\mathbf{B}}(\boldsymbol{\zeta}_t)$, we can correct for the inflated variance by simulating $\mathbf{Z} \sim N(0, \mathbf{D}(\boldsymbol{\zeta}_t) - h\hat{\mathbf{B}}(\boldsymbol{\zeta}_t))$. Obviously, this requires that $\mathbf{D}(\boldsymbol{\zeta}_t) - h\hat{\mathbf{B}}(\boldsymbol{\zeta}_t)$ is positive semi-definite. In many cases this can be enforced if h is small enough. If this is not possible, then that suggests the resulting SGMCMC algorithm will be unstable; see below for an example.

The diffusion $\mathbf{D}(\boldsymbol{\zeta})$ and curl $\mathbf{Q}(\boldsymbol{\zeta})$ matrices can take various forms and the choice of matrices will affect the rate of convergence of the MCMC samplers. The diffusion matrix $\mathbf{D}(\boldsymbol{\zeta})$ controls the level of noise introduced into the dynamics of (11). When $||\mathbf{D}(\boldsymbol{\zeta})||$ is large, there is a greater chance that the sampler can escape local modes of the target, and setting $||\mathbf{D}(\boldsymbol{\zeta})||$ to be small increases the accuracy of the sampler within a local mode. Between modes of the target, the remainder of the parameter space is represented by regions of low probability mass where we would want our MCMC sampler to quickly pass through. The curl matrix $\mathbf{Q}(\boldsymbol{\zeta})$ controls the sampler's non-reversible dynamics which allows the sampler to quickly traverse low-probability regions, this is particularly efficient when the curl matrix adapts to the geometry of the target.

In Table 1 we define $H(\boldsymbol{\zeta})$, $\mathbf{D}(\boldsymbol{\zeta})$ and $\mathbf{Q}(\boldsymbol{\zeta})$ for several gradient-based MCMC algorithms. The two most common are SGLD, which we introduced in the previous section, and SGHMC (Chen et al. 2014). This latter process introduces a velocity component that can help improve mixing, as is seen in more standard Hamiltonian MCMC methods. The closest link with the dynamics used in Hamiltonian MCMC is when $\mathbf{D}(\boldsymbol{\zeta})$ is set to be the

zero-matrix. However (Chen et al. 2014) show that this leads to an unstable process that diverges as a result of the accumulation of noise in the estimate of the gradient; a property linked to the fact that $\mathbf{D}(\boldsymbol{\zeta}) - h\hat{\mathbf{B}}(\boldsymbol{\zeta})$ is not positive semi-definite for any h. The choice of $\mathbf{D}(\boldsymbol{\zeta})$ given in Table 1 avoids this problem, with the resulting stochastic differential equation being the so-called under-damped Langevin diffusion.

As discussed in Section 2.5 with regard to SGLD, re-parameterising the target distribution so that the components of $\boldsymbol{\theta}$ are roughly uncorrelated and have similar marginal variances, can improve mixing. An extension of this idea is to adapt the dynamics locally to the curvature of the target distribution – and this is the idea behind Riemannian versions of SGLD and SGHMC, denoted by SGRLD (Patterson & Teh 2013) and SGRHMC (Ma et al. 2015) in Table 1. The challenge with implementing either of these algorithms is obtaining an accurate, yet easy to compute, estimate of the local curvature. A simpler approach is the stochastic gradient Nose-Hoover thermostat (SGNHT) (Ding et al. 2014) algorithm, which introduces state dependence into the curl matrix. This can be viewed as an extension of SGHMC which adaptively controls for the excess noise in the gradients. Naturally, there are many other algorithms that could be derived from this general framework.

[Table 1 about here.]

3.1 Theory for SGHMC

It is natural to ask which of the algorithms presented in Table 1 is most accurate. We will study this question empirically in Section 6, but here we briefly present some theoretical results that compare SGHMC with SGLD for smooth and strongly log-concave target densities. These results are for bounds on the Wasserstein distance between the target distribution and the distribution of the SGMCMC algorithm samples at iteration

k, for an optimally chosen step size (Cheng et al. 2018). The simplest comparison of the efficiencies of the two algorithms is for the case where the gradients are estimated without error. For a given level of accuracy, ϵ, measured in terms of Wasserstein distance, SGLD requires $O(d^2/\epsilon^2)$ iterations, whereas SGHMC requires $O(d/\epsilon)$ iterations. This suggests that SGHMC is to be preferred, and the benefits of SGHMC will be greater in higher dimensions. Similar results are obtained when using noisy estimates of the gradients, providing the variance of the estimates is small enough. However, Cheng et al. (2018) show that there is a phase-transition in the behaviour of SGHMC as the variance of the gradient estimates increases: if it is too large, the SGHMC behaves like SGLD and needs a similar order of iterations to achieve a given level of accuracy.

4 Diagnostic Tests

When using an MCMC algorithm the practitioner wants to know if the algorithm has converged to the stationary distribution, and how to tune the MCMC algorithm to maximise the efficiency of the sampler. In the case of SGMCMC, the target distribution is not the stationary distribution and therefore our posterior samples represent an asymptotically biased approximation of the posterior. Standard MCMC diagnostic tests (Brooks & Gelman 1998) do not account for this bias and therefore are not appropriate for either assessing convergence or tuning SGMCMC algorithms. The design of appropriate diagnostic tests for SGMCMC is a relatively new area of research, and currently methods based on Stein's discrepancy (Gorham & Mackey 2015, Gorham et al. 2019, Gorham & Mackey 2017) are the most popular approach. These methods provide a general way of assessing how accurately a sample of values approximate a distribution.

Assume we have a sample, say from an SGMCMC algorithm, $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K) \in \mathbb{R}^d$, and denote the empirical distribution that this sample defines as $\tilde{\pi}$. We can define a measure of how well this sample approximates our target distribution of interest, π, by comparing how close expectations under $\tilde{\pi}$ are to the expectations under π. If they are close for a broad class of functions, $\mathcal{H}$, then this suggests the approximation error is small. This motivates the following measure of discrepancy,

$$d_{\mathcal{H}}(\tilde{\pi}, \pi) := \sup_{\hbar \in \mathcal{H}} |\mathbb{E}_{\tilde{\pi}} [\hbar(\boldsymbol{\theta})] - \mathbb{E}_{\pi} [\hbar(\boldsymbol{\theta})] |, \tag{12}$$

where $\mathbb{E}_{\tilde{\pi}} [\hbar(\boldsymbol{\theta})] = \frac{1}{K} \sum_{k=1}^{K} \hbar(\boldsymbol{\theta}_k)$ is an approximation of $\mathbb{E}_{\pi} [\hbar(\boldsymbol{\theta})]$. For appropriate choices of $\mathcal{H}$, it can be shown that if we denote the approximation from a sample of size K by $\tilde{\pi}_K$, then $d_{\mathcal{H}}(\tilde{\pi}_K, \pi) \to 0$ if and only if $\tilde{\pi}_K$ converges weakly to π. Moreover, even if this is not the case, if functions of interest are in $\mathcal{H}$ then a small value of $d_{\mathcal{H}}(\tilde{\pi}, \pi)$ would mean that we can accurately estimate posterior expectations of functions of interest.

Unfortunately, (12) is in general intractable as it depends on the unknown $\mathbb{E}_{\pi} [\hbar(\boldsymbol{\theta})]$. The Stein discrepancy approach circumvents this problem by using a class, $\mathcal{H}$, that only contains functions whose expectation under π are zero. We can construct such functions from stochastic processes, such as the Langevin diffusion, whose invariant distribution is π. If the initial distribution of such a process is chosen to be π then the expectation of the state of the process will be constant over time. Moreover, the rate of change of expectations can be written in terms of the expectation of the generator of the process applied to the function: which means that functions that can be written in terms of the generator applied to a function will have expectation zero under π.

In our experience, the computationally most feasible approach, and easiest to implement, is the kernel Stein set approach of Gorham & Mackey (2017), which enables the discrepancy to be calculated as a sum of some kernel evaluated at all pairs of points in the

sample. As with all methods based on Stein discrepancies, it also requires the gradient of the target at each sample point – though we can use unbiased noisy estimates for these (Gorham & Mackey 2017). The kernel Stein discrepancy is defined as

$$KSD(\tilde{\pi}_K, \pi) := \sum_{j=1}^{d} \sqrt{\sum_{k,k'=1}^{K} \frac{k_j^0(\boldsymbol{\theta}_k, \boldsymbol{\theta}_{k'})}{K^2}}, \tag{13}$$

where the Stein kernel for $j \in \{1, \ldots, d\}$ is given by

$$\begin{aligned} k_j^0(\boldsymbol{\theta}, \boldsymbol{\theta}') =& (\nabla_{\boldsymbol{\theta}^{(j)}} U(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}'^{(j)}} U(\boldsymbol{\theta}')) k(\boldsymbol{\theta}, \boldsymbol{\theta}') + \nabla_{\boldsymbol{\theta}^{(j)}} U(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}'^{(j)}} k(\boldsymbol{\theta}, \boldsymbol{\theta}') \\ &+ \nabla_{\boldsymbol{\theta}'^{(j)}} U(\boldsymbol{\theta}') \nabla_{\boldsymbol{\theta}^{(j)}} k(\boldsymbol{\theta}, \boldsymbol{\theta}') + \nabla_{\boldsymbol{\theta}^{(j)}} \nabla_{\boldsymbol{\theta}'^{(j)}} k(\boldsymbol{\theta}, \boldsymbol{\theta}'). \end{aligned}$$

The kernel k has to be carefully chosen, particularly when $d \geq 3$, as some kernel choices, e.g. Gaussian and Matern, result in a kernel Stein discrepancy which does not detect non-convergence to the target distribution. Gorham & Mackey (2017) recommend using the inverse multi-quadratic kernel, $k(\boldsymbol{\theta}, \boldsymbol{\theta}') = (c^2 + ||\boldsymbol{\theta} - \boldsymbol{\theta}'||_2^2)^{\beta}$,which they prove detects non-convergence when $c > 0$ and $\beta \in (-1, 0)$. A drawback of most Stein discrepancy measures, including the kernel Stein method, is that the computational cost scales quadratically with the sample size. This is more computationally expensive than standard MCMC metrics (e.g. effective sample size), however, the computation can be easily parallelised to give faster calculations.

We illustrate the kernel Stein discrepancy on the Gaussian target introduced in Section 2.5, where we choose diagonal and rotation matrices

$$D = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad P = \begin{pmatrix} \cos\frac{\pi}{4} & \sin\frac{\pi}{4} \\ -\sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{pmatrix}.$$

[Figure 1 about here.]

We iterate the Langevin dynamics (7) for 10,000 iterations, starting from $\boldsymbol{\theta} = (0, 0)$ and with noisy gradients simulated as the true gradient plus noise, $\boldsymbol{\nu}_k \sim N(0, 0.01)$. We test the efficiency of the Langevin algorithm in terms of the step size parameter h and use the kernel Stein discrepancy metric (13) to select a step size parameter which produces samples that most closely approximate the target distribution. We consider a range of step size parameters $h = \{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$ which satisfy the requirement that $h < 4\sigma_2^2$ to prevent divergent chains. In Figure 1, we plot the samples generated from the Langevin algorithm for each of the step size parameters. We also calculate the kernel Stein discrepancy (13) and effective sample size for each Markov chain. Visually, it is clear from Figure 1 that $h = 0.1$ produces samples which most closely represent the target distribution. A large value for h leads to over-dispersed samples and a small h prevents the sampler from exploring the whole target space within the fixed number of iterations. Setting $h = 0.1$ also gives the lowest kernel Stein discrepancy, whereas $h = 1$ maximises the effective sample size. This supports the view that effective sample size and other standard MCMC metrics, which do not account for sample bias, are not appropriate diagnostic tools for SGMCMC.

5 Extending the SGMCMC framework

Under the general SGMCMC framework outlined in Section 3, it is possible to extend the SGLD algorithm beyond Langevin dynamics and consider a larger class of MCMC algorithms, which aim to improve the mixing of the Markov chain. In this section, we will focus on ways to extend the applicability of SGMCMC algorithms to a wider class of models. Given our choice of target (1), we have made two key assumptions, i) the parameters exist in $\boldsymbol{\theta} \in \mathbb{R}^d$ and ii) the potential function $U(\boldsymbol{\theta})$ is a summation over independent terms. The

first assumption implies that SGMCMC cannot be used to estimate $\boldsymbol{\theta}$ on a constrained space (e.g. $\boldsymbol{\theta} \in [0, 1]$) and the second assumption that our data $y_1, \ldots, y_N$ are independent or have only certain-types of dependence structure, which means that SGMCMC cannot be applied to many time series or spatial models. We will give a short overview of some of the current research in this area.

SGMCMC sampling from constrained spaces

Many models contain parameters which are constrained, for example, the variance parameter τ^2 in a Gaussian distribution ($\tau \in \mathbb{R}^+$), or the success probability p in a Bernoulli model ($p \in [0, 1]$). Simulating these constrained parameters using the Langevin dynamics (3) will produce samples which violate their constraints, for example, if $\tau_t^2 = \boldsymbol{\theta}_t \gtrapprox 0$, then with high probability, $\tau_{t+1}^2 < 0$. One solution would be to let $h \to 0$ when $\tau^2 \to 0$, however, this would lead to poor mixing of the Markov chain near the boundary of the constrained space. A natural solution to this problem is to transform the Langevin dynamics in such a way that sampling can take place on the unconstrained space, but care is needed as the choice of transformation can impact the mixing of the process near the boundary. Alternatively we can project the Langevin dynamics into a constrained space (Brosse et al. 2017, Bubeck et al. 2018), however, these approaches lead to poorer non-asymptotic convergence rates than in the unconstrained setting. Recently, a mirrored Langevin algorithm (Hsieh et al. 2018) has been proposed, which builds on the mirrored descent algorithm (Beck & Teboulle 2003), to transform the problem of constrained sampling to an unconstrained space via a mirror mapping. Unlike previous works, the mirrored Langevin algorithm has convergence rates comparable with unconstrained SGLD (Dalalyan & Karagulyan 2019).

The structure of some models naturally leads to bespoke sampling strategies. A popu-

lar model in the machine learning literature is the latent Dirichlet allocation (LDA) model (Blei et al. 2003), where the model parameters are constrained to the probability simplex, meaning $\boldsymbol{\theta}^{(j)} \geq 0,\ j = 1, \ldots, d$ and $\sum_{j=1}^{d} \boldsymbol{\theta}^{(j)} = 1$. Patterson & Teh (2013) proposed the first SGLD algorithm for sampling from the probability simplex. Their algorithm, stochastic gradient Riemannian Langevin dynamics (see Table 1) allows for several transformation schemes which transform $\boldsymbol{\theta}$ to $\mathbb{R}^d$. However, this approach can result in asymptotic biases which dominate in the boundary regions of the constrained space. An alternative approach is to use the fact that the posterior for the LDA can be written as a transformation of independent gamma random variables. Using an alternative stochastic process instead of the Langevin diffusion, in this case the Cox-Ingersoll-Ross (CIR) process, we take advantage of the fact that its invariant distribution is a gamma distribution. We can apply this in the large data setting by using data subsampling on the CIR process rather than on the Langevin diffusion (Baker et al. 2018).

SGMCMC sampling with dependent data

Key to developing SGMCMC algorithms is the ability to generate unbiased estimates of $\nabla U(\boldsymbol{\theta})$ using data subsampling, as in (4). Under the assumption that data $y_i,\ i = 1, \ldots, N$ are independent, the potential function $U(\boldsymbol{\theta}) = \sum_{i=1}^{N} U_i(\boldsymbol{\theta})$, and its derivative, are a sum of independent terms (see Section 2.1) and therefore, a random subsample of these terms leads to an unbiased estimate of the potential function, and its derivative. For some dependence structures, we can still write the potential as a sum of terms each of which has an $O(1)$ cost to evaluate. However for many models used for network data, time series and spatial data, using the same random subsampling approach will result in biased estimates for $U(\boldsymbol{\theta})$ and $\nabla U(\boldsymbol{\theta})$. To the best of our knowledge, the challenge of subsampling spatial data, such that

both short and long term dependency is captured, has not been addressed in the SGMCMC setting. For network data, an SGMCMC algorithm has been developed (Li et al. 2016) for the mixed-member stochastic block model, which uses both the block structure of the model, and stratified subsampling techniques, to give unbiased gradient estimates.

In the time series setting, hidden Markov models are challenging for SGMCMC as the temporal dependence in the latent process precludes simple random data subsampling. However, such dependencies are often short range and so data points y_i and y_j will be approximately independent if they are sufficiently distant (i.e. $j >> i$). These properties were used by Ma et al. (2017), who proposed using SGMCMC with gradients estimated using non-overlapping, subsequences of length $2s + 1$, $\mathbf{y}_{i,s} = \{y_{i-s}, \ldots, y_i, \ldots, y_{i+s}\}$. In order to ensure that the subsequences are independent, Ma et al. (2017) extend the length of each subsequence by adding a buffer of size B, to either side, i.e. $\{\mathbf{y}_{LB}, \mathbf{y}_{i,s}, \mathbf{y}_{RB}\}$, where $\mathbf{y}_{LB} = \{y_{i-s-B}, \ldots, y_{i-s-1}\}$ and $\mathbf{y}_{RB} = \{y_{i+s+1}, \ldots, y_{i+s+B}\}$. Non-overlapping buffered subsequences are sampled, but only $\mathbf{y}_{i,s}$ data are used to estimate $\hat{\nabla} U(\boldsymbol{\theta})$. These methods introduce a bias, but one that can be controlled, with the bias often decreasing exponentially with the buffer size. This approach has also been applied to linear (Aicher, Ma, Foti & Fox 2019) and nonlinear (Aicher, Putcha, Nemeth, Fearnhead & Fox 2019) state-space models, where in the case of log-concave models, the bias decays geometrically with buffer size.

6 Simulation Study

We compare empirically the accuracy and efficiency of the SGMCMC algorithms described in Section 3. We consider three popular models. Firstly, a logistic regression model for binary data classification tested on simulated data. Secondly, a Bayesian neural network

(Neal 2012) applied to image classification on a popular data set from the machine learning literature. Finally, we consider the Bayesian probabilistic matrix factorisation model (Salakhutdinov & Mnih 2008) for predicting movie recommendations based on the MovieLens data set. We compare the various SGMCMC algorithms against the STAN software (Carpenter et al. 2017), which by default implements the NUTS algorithm (Hoffman & Gelman 2014) as a method for automatically tuning the Hamiltonian MCMC sampler. We treat the STAN output as the ground truth posterior distribution and assess the accuracy and computational advantages of SGMCMC against this benchmark. Additionally, using STAN, we can sample from a variational approximation to the posterior using the automatic differentiation variational inference (ADVI) algorithm (Kucukelbir et al. 2015), which selects an appropriate variational family and optimises the corresponding variational objective. All of the SGMCMC algorithms are implemented using the R package *sgmcmc* (Baker et al. 2019*b*) with supporting code available online[2].

6.1 Logistic regression model

Consider a binary regression model where $\mathbf{y} = \{y_i\}_{i=1}^N$ is a vector of N binary responses and $\mathbf{X}$ is a $N \times d$ matrix of covariates. If $\boldsymbol{\theta}$ is a $d-$dimensional vector of model parameters, then the likelihood function for the logistic regression model is,

$$p(\mathbf{y}, \mathbf{X} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} \left[\frac{1}{1 + \exp(-\boldsymbol{\theta}^\top \mathbf{x}_i)} \right]^{y_i} \left[1 - \frac{1}{1 + \exp(-\boldsymbol{\theta}^\top \mathbf{x}_i)} \right]^{1-y_i}$$

where $\mathbf{x}_i$ is a $d-$dimensional vector for the ith observation. The prior distribution for $\boldsymbol{\theta}$ is a zero-mean Gaussian with covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = 10\mathbf{I}_d$, where $\mathbf{I}_d$ is a $d \times d$ identity matrix. We can verify that the model satisfies the strong-convexity assumptions from Section 2.4,

[2]https://github.com/***

where $m = \lambda_{\max}^{-1}(\boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ and $M = \frac{1}{4}\sum_{i=1}^{N} \mathbf{x}_i^\top \mathbf{x}_i + \lambda_{\min}^{-1}(\boldsymbol{\Sigma}_{\boldsymbol{\theta}})$, and $\lambda_{\min}(\boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ and $\lambda_{\max}(\boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ are the minimum and maximum eigenvalues of $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$.

We compare the various SGMCMC algorithms where we vary the dimension of $\boldsymbol{\theta}$, $d = \{10, 50, 100\}$. We simulate $N = 10^5$ data points and fix the subsample size $n = 0.01N$ for all test cases. We simulated data under the model described above, with $\mathbf{x}_i \sim N(0, \boldsymbol{\Sigma}_{\mathbf{x}})$ and simulated a matrix with $\boldsymbol{\Sigma}_{\mathbf{x}}^{(i,j)} = \text{Unif}[-\rho, \rho]^{|i-j|}$ and $\rho = 0.4$. We tune the step size h for each algorithm using the kernel Stein discrepancy metric outlined in Section 4 and set the number of leapfrog steps in SGHMC to five. We initialise each sampler by randomly sampling the first iteration $\boldsymbol{\theta}_0 \sim N(0, 1)$.

For our simulations, we ran STAN for $2,000$ iterations and discarded the first $1,000$ iterations as burn-in, as these iterations are part of the algorithms tuning phase. For the SGMCMC algorithms, we ran each algorithm for $20,000$ iterations except in the case of the control variate implementations, where we ran the SGMCMC algorithm for $10,000$ iterations after iterating a stochastic gradient descent algorithm for $10,000$ iterations to find the posterior mode $\hat{\boldsymbol{\theta}}$. Combining the optimisation and sampling steps of the control variate method results in an equal number of iterations for all SGMCMC algorithms. Figure 2 gives the trace plots for MCMC output of each algorithm for the case where $d = 10$ and $N = 10^5$. Each of the SGMCMC algorithms is initialised with the same $\boldsymbol{\theta}_0$ and we see that some components of $\boldsymbol{\theta}$, where the posterior is not concentrated around $\boldsymbol{\theta}_0$, take several thousand iterations to converge. Most notably SGLD, ULA, SGHMC and SGNHT. Of these algorithms, SGHMC and SGNHT converge faster than SGLD, which reflects the theoretical results discussed in Section 3.1, but these algorithms also have a higher computational cost due to the leap frog steps (see Table 2 for computational timings). The ULA algorithm, which uses exact gradients, also converges faster than SGLD in terms of the number of

iterations, but is less efficient in terms of overall computational time. The control variate SGMCMC algorithms, SGLD-CV, SGHMC-CV and SGNHT-CV are all more efficient than their non-control variate counterparts in terms of the number of iterations required for convergence. The control variate algorithms have the advantage that their sampling phase is initialised at a $\boldsymbol{\theta}_0$ that is close to the posterior mode. In essence, the optimisation phase required to find the control variate point $\hat{\boldsymbol{\theta}}$ replaces the burn-in phase of the Markov chain for the SGMCMC algorithm.

[Figure 2 about here.]

The results from Figure 2 are shown for a fixed number of iterations, however, the computational cost per iteration varies between the algorithms. In Figure 3 we run STAN, SGLD, SGLD-CV and ULA for 10 minutes, treating the first minute as the burn-in phase and a longer 1-hour run of STAN as the truth. We can see in this experiment that over short time-periods SGLD performs well, whereas STAN underestimates the posterior variance due to fewer iterations, which results in less time for the chain to mix. SGLD and SGLD-CV produce good estimates of the mean, but as discussed in Section 2.5, SGLD and SGLD-CV over-estimate the variance. Using exact gradients with ULA performs poorly as it does not have the same gains in computational efficiency of SGLD and still has an approximation error.

[Figure 3 about here.]

As well as the visual comparisons (Figure 2), we can compare the algorithms using diagnostic metrics. We use the kernel Stein discrepancy as one of the metrics to assess the quality of the posterior approximation for each of the algorithms. Additionally, the log-loss

is also a popular metric for measuring the predictive accuracy of a classifier on a held-out test data set T_*. In the case of predicted binary responses, the log-loss is

$$l(\boldsymbol{\theta}, T_*) = -\frac{1}{|T_*|} \sum_{(y_*, \mathbf{x}_*) \in T_*} y_* \log p(\mathbf{x}_*, \boldsymbol{\theta}) + (1 - y_*) \log(1 - p(\mathbf{x}_*, \boldsymbol{\theta})),$$

where $p(\mathbf{x}_*, \boldsymbol{\theta}) = (1 + \exp(-\boldsymbol{\theta}^\top \mathbf{x}_*))^{-1}$ is the probability that $y_* = 1$ given covariate $\mathbf{x}_*$.

Table 2 gives the diagnostic metrics for each algorithm, where the log-loss and kernel Stein discrepancy metrics are calculated on the final $1,000$ posterior samples from each algorithm. We also include a variational Bayes approximation using STAN's ADVI algorithm. The variational Bayes approaches are generally faster than MCMC as they use optimisation techniques rather than sampling to approximate the posterior. These variational techniques work particularly well when the posterior is close to its approximating family of distributions, which are usually assumed to be Gaussian. The most notable difference between the algorithms is the computational time. Compared to STAN, all SGMCMC algorithms, and ADVI, are between 10 to 100 times faster when $d = 100$. As expected, given that STAN produces exact posterior samples, it has the lowest log-loss and kernel Stein discrepancy results. However, these results are only slightly better than the SGMCMC results and the computational cost of STAN is significantly higher. All of the SGMCMC results are similar, showing that this class of algorithms can perform well, with significant computational savings, if they are well-tuned. Similarly, we note that the variational approximations produce accuracy results similar to SGMCMC and are significantly computationally cheaper than STAN. One of the advantages of STAN, is that the NUTS algorithm (Hoffman & Gelman 2014) allows the HMC sampler to be automatically tuned, whereas the SGMCMC algorithms have to be tuned using a pilot run over a grid of step size values. As the step size h is a scalar value, the SGMCMC samplers give an equal step size to each dimension.

As discussed in Section 2.5, a scalar step size parameter will mean that the SGMCMC algorithms are constrained by the $\boldsymbol{\theta}$ component with the smallest variance. This could be improved if either the gradients were pre-conditioned (Ahn et al. 2012), or the geometry of the posterior space were accounted for in the sampler (e.g. SGRHMC), which would result in different step sizes for each component of $\boldsymbol{\theta}$, thus improving the overall efficiency of the sampler.

[Table 2 about here.]

6.2 Bayesian neural network

We consider the problem of multi-class classification on the popular MNIST data set (LeCun et al. 2010). The MNIST data set consists of a collection of images of handwritten digits from zero to nine, where each image is represented as 28×28 pixels (a sample of images is shown in Figure 4). We model the data using a two layer Bayesian neural network with 100 hidden variables (using the same setup as Chen et al. (2014)). We fit the neural network to a training data set containing $55,000$ images and the goal is to classify new images as belonging to one of the ten categories. The test set contains $10,000$ handwritten images, with corresponding labels.

[Figure 4 about here.]

Let y_i be the image label taking values $y_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and $\mathbf{x}_i$ is the vector of pixels which has been flattened from a 28×28 image to a one-dimensional vector of length 784. If there are N training images, then $\mathbf{X}$ is a $N \times 784$ matrix representing the full data set of pixels. We model the data as categorical variables with the probability mass

function,

$$p(y_i = k \,|\, \boldsymbol{\theta}, \mathbf{x}_i) = \beta_k(\boldsymbol{\theta}, \mathbf{x}_i), \tag{14}$$

where $\beta_k(\boldsymbol{\theta}, \mathbf{x}_i)$ is the kth element of $\beta(\boldsymbol{\theta}, \mathbf{x}_i) = \sigma\left(\sigma\left(\mathbf{x}_i^\top B + b\right) A + a\right)$ and $\sigma(\mathbf{x}_i) = \exp(\mathbf{x}_i)/(\sum_{j=1}^{N} \exp(\mathbf{x}_i))$ is the softmax function, a generalisation of the logistic link function. The parameters $\boldsymbol{\theta} = (A, B, a, b)$ will be estimated using SGMCMC, where A, B, a and b are matrices of dimension: 100×10, 784×100, 1×10 and 1×100, respectively. We set normal priors for each element of these parameters

$$A_{kl}|\lambda_A \sim N(0, \lambda_A^{-1}), \quad B_{jk}|\lambda_B \sim N(0, \lambda_B^{-1}), \quad a_l|\lambda_a \sim N(0, \lambda_a^{-1}), \quad b_k|\lambda_b \sim N(0, \lambda_b^{-1}),$$

$j = 1, \ldots, 784; \quad k = 1, \ldots, 100; \quad l = 1, \ldots, 10;$ where $\lambda_A, \lambda_B, \lambda_a, \lambda_b \sim \text{Gamma}(1, 1)$ are hyperparameters.

Similar to the logistic regression example (see Section 6.1), we use the log-loss as a test function. We need to update the definition of the log-loss function from a binary classification problem to the multi-class setting. Given a test set T_* of pairs $(y_*, \mathbf{x}_*)$, where now y_* can take values $\{0 - 9\}$. The log-loss function in the multi-class setting is now

$$l(\boldsymbol{\theta}, T_*) = -\frac{1}{|T_*|} \sum_{(y_*, \mathbf{x}_*) \in T_*} \sum_{k=0}^{9} \mathbf{1}_{y_*=k} \log \beta_k(\boldsymbol{\theta}, \mathbf{x}_*), \tag{15}$$

where $\mathbf{1}_A$ is the indicator function, and $\beta_k(\boldsymbol{\theta}, \mathbf{x}_*)$ is the k^{th} element of $\beta(\boldsymbol{\theta}, \mathbf{x}_*)$.

[Figure 5 about here.]

As in Section 6.1, we compare the efficacy of the SGLD, SGHMC and SGNHT algorithms, as well as their control variate counterparts. We ran each of the SGMCMC algorithms for 10^4 iterations and calculated the log-loss (14) for each algorithm. The standard algorithms have 10^4 iterations of burn-in while the control variate algorithms have no

burn-in, but 10^4 iterations in the initial optimisation step. Note that due to the trajectory parameter $L = 5$ of SGHMC and SGHMC-CV, these algorithms will have approximately five times greater computational cost. In order to balance the computational cost, we ran these algorithms for 2,000 iterations in order to produce comparisons with approximately equal computational time. The results are plotted in Figure 5. As with the logistic regression example, we note that there is some indication of improved predictive performance of the control variate methods. Among the standard methods, SGHMC and SGNHT have the best predictive performance, which is to be expected given the apparent trade-off between accuracy and exploration.

6.3 Bayesian probabilistic matrix factorisation

Collaborative filtering is a technique used in recommendation systems to make predictions about a user's interests based on their tastes and preferences. We can represent these preferences with a matrix where the (i, j)th entry is the score that user i gives to item j. This matrix is naturally sparse as not all users provide scores for all items. We can model these data using Bayesian probabilistic matrix factorisation (BPMF) (Salakhutdinov & Mnih 2008), where the preference matrix of user-item ratings is factorised into lower-dimensional matrices representing the users' and items' latent features. A popular application of BPMF is movie recommendations, where the preference matrix contains the ratings for each movie given by each user. This model has been successfully applied to the Netflix data set to extract the latent user-item features from the historical data in order to make movie recommendations for a held-out test set of users. In this example, we will consider the MovieLens data set [3] which contains $100,000$ ratings (taking values $\{1, 2, 3, 4, 5\}$) of $1,682$ movies by

[3] https://grouplens.org/data sets/movielens/100k/

943 users, where each user has provided at least 20 ratings. The data are already split into 5 training and test sets (80%/20% split) for a $5-$fold cross-validation experiment.

Let $\mathbf{R} \in \mathbb{R}^{N \times M}$ be a matrix of observed ratings for N users and M movies where R_{ij} is the rating user i gave to movie j. We introduce matrices $\mathbf{U}$ and $\mathbf{V}$ for users and movies respectively, where $\mathbf{U}_i \in \mathbb{R}^d$ and $\mathbf{V}_j \in \mathbb{R}^d$ are $d-$dimensional latent feature vectors for user i and movie j. The likelihood for the rating matrix is

$$p(\mathbf{R}|\mathbf{U}, \mathbf{V}, \alpha) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[N(R_{ij}|\mathbf{U}_i^\top \mathbf{V}_j, \alpha^{-1}) \right]^{I_{ij}}$$

where I_{ij} is an indicator variable which equals 1 if user i gave a rating for movie j. The prior distributions for the users and movies are

$$p(\mathbf{U}|\mu_{\mathbf{U}}, \Lambda_{\mathbf{U}}) = \prod_{i=1}^{N} N(\mathbf{U}_i|\mu_{\mathbf{U}}, \Lambda_{\mathbf{U}}^{-1}) \quad \text{and} \quad p(\mathbf{V}|\mu_{\mathbf{V}}, \Lambda_{\mathbf{V}}) = \prod_{j=1}^{M} N(\mathbf{V}_j|\mu_{\mathbf{V}}, \Lambda_{\mathbf{V}}^{-1}),$$

with prior distributions on the hyperparameters (where $\mathbf{W} = \mathbf{U}$ or $\mathbf{V}$) given by,

$$\mu_{\mathbf{W}} \sim N(\mu_{\mathbf{W}}|\mu_0, \Lambda_{\mathbf{W}}) \quad \text{and} \quad \Lambda_{\mathbf{W}} \sim \text{Gamma}(a_0, b_0).$$

The parameters of interest in our model are then $\boldsymbol{\theta} = (\mathbf{U}, \mu_{\mathbf{U}}, \Lambda_{\mathbf{U}}, \mathbf{V}, \mu_{\mathbf{V}}, \Lambda_{\mathbf{V}})$ and the hyperparameters for the experiments are $\boldsymbol{\tau} = (\alpha, \mu_0, a_0, b_0) = (3, 0, 1, 5)$. We are free to choose the size of the latent dimension and for these experiments we set $d = 20$.

The predictive distribution for an unknown rating R_{ij}^* given to movie j by user i, is found by marginalising over the latent feature parameters

$$p(R_{ij}^*|\mathbf{R}, \boldsymbol{\tau}) = \int p(R_{ij}^*|\mathbf{U}_i, \mathbf{V}_j, \alpha) \pi(\boldsymbol{\theta}|\mathbf{R}, \boldsymbol{\tau}) \mathrm{d}\boldsymbol{\theta}.$$

We can approximate the predictive density using Monte Carlo integration, where the posterior samples, conditional on the training data, are generated using the SGMCMC algorithms. The held-out test data can be used to assess the predictive accuracy of each of the SGMCMC algorithms, where we use the root mean square error (RMSE) between the predicted and actual rating as an accuracy metric.

[Figure 6 about here.]

We ran each of the SGMCMC algorithms for 10^5 iterations, where for SGLD-CV and SGHMC-CV we applied a stochastic gradient descent algorithm for $50,000$ iterations to find the posterior mode and used this as the fixed point for the control variate, as well as initialising these SGMCMC samplers from the control variate point (i.e. $\boldsymbol{\theta}_0 = \hat{\boldsymbol{\theta}}$). Given the size of the parameter space, we increase the subsample size to $n = 0.1N$ per iteration and tune the step size parameter for each SGMCMC algorithm using diagnostic tests (see Section 4) on a pilot run with 10^4 iterations. As a baseline to assess the accuracy of the SGMCMC algorithms we applied the NUTS sampler from the STAN software to the full data set and ran this for 10^4 iterations, discarding the first half as burn-in. We also tested a fast variational approximation using STAN's ADVI algorithm. Figure 6 gives the RMSE for STAN, ADVI, SGLD and SGHMC along with their control variate versions. The results show that SGHMC produces a lower RMSE than SGLD on the test data with equally improved results for their control variate implementations. ADVI, SGLD and SGHMC quickly converge to a stable RMSE after a few thousand iterations with SGLD-CV and SGHMC-CV producing an overall lower RMSE immediately as they are both initialised from the posterior mode, which removes the burn-in phase. Most notable from these results is that all of the SGMCMC algorithms, and ADVI, outperform the STAN baseline

RMSE. The poorer performance of STAN is attributable to running the algorithm for fewer iterations than the SGMCMC algorithms which could mean that the MCMC sampler has not converged. Running STAN for 10% of the iterations of the SGMCMC algorithms took 3.5 days, whereas SGLD, SGLD-CV, SGHMC and SGHMC-CV took 3.1, 1.9, 16.4 and 14.8 hours, respectively. The ADVI algorithm has a similar computational time to the SGMCMC algorithms, but as it is an optimisation rather than sampling routine, the algorithm stops after it has converged, which in this example occurs after approximately 1,000 iterations. Overall, while SGMCMC algorithms produce biased posterior approximations compared to exact MCMC algorithms, such as STAN's NUTS sampler, they can produce accurate estimates of quantities of interest at significantly reduced computational cost.

7 Discussion

In this paper we have provided a review of the growing literature on SGMCMC algorithms. These algorithms utilise data subsampling to significantly reduce the computational cost of MCMC. As shown in this paper, these algorithms are theoretically well-understood and provide parameter inference at levels of accuracy that are comparable to traditional MCMC algorithms. SGMCMC is still a relatively new class of Monte Carlo algorithms compared to traditional MCMC methods and there remain many open problems and opportunities for further research in this area.

Some key areas for future development in SGMCMC include:

- New algorithms - as discussed in Section 3.1, SGMCMC represents a general class of scalable MCMC algorithms with many popular algorithms given as special cases, therefore it is possible to derive new algorithms from this general setting which may

be more applicable for certain types of target distribution.

- General theoretical results - most of the current theoretical results which bound the error of SGMCMC algorithms assume that the target distribution is log-concave. Relaxing this assumption is non-trivial and may need completely different arguments in order to show similar non-asymptotic error bounds for a broader class of models.

- Tuning techniques - as outlined in Section 4, the efficacy of SGMCMC is dependent on how well the step size parameter is tuned. Standard MCMC tuning rules, such as those based on acceptance rates, are not applicable and new techniques, such as the Stein discrepancy metrics, can be computationally expensive to apply. Developing robust tuning rules, which can be applied in an automated fashion, would make it easier for non-experts to use SGMCMC methods in the same way that adaptive HMC has been applied in the STAN software.

A major success of traditional MCMC algorithms, and their broad appeal in a range of application areas, is partly a result of freely available software, such as WinBUGS (Lunn et al. 2000), JAGS (Plummer 2003), NIMBLE (de Valpine et al. 2017) and STAN (Carpenter et al. 2017). Open-source MCMC software, which may utilise specials features of the target distribution, or provide automatic techniques to adapt the tuning parameters, make MCMC methods more user-friendly to general practitioners. Similar levels of development for SGMCMC, which provide automatic differentiation and adaptive step size parameter tuning, would help lower the entry level for non-experts. Some recent developments in this area include *sgmcmc* in R (Baker et al. 2019*b*) and *Edward* in Python (Tran et al. 2016), but further development is required to fully utilise the general SGMCMC framework.

Acknowledgments

The authors would like to thank Jack Baker for his guidance and help using the `sgmcmc` package. CN gratefully acknowledges the support of EPSRC grants EP/S00159X/1 and EP/R01860X/1. PF is supported by the EPSRC-funded projects Bayes4Health EP/R018561/1 and CoSInES EP/R034710/1.

References

Ahn, S., Korattikara, A., Liu, N., Rajan, S. & Welling, M. (2015), Large-scale distributed Bayesian matrix factorization using stochastic gradient MCMC, *in* 'Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining', ACM, pp. 9–18.

Ahn, S., Korattikara, A. & Welling, M. (2012), Bayesian posterior sampling via stochastic gradient Fisher scoring, *in* 'Proceedings of the 29th International Conference on Machine Learning, ICML 2012', pp. 1591–1598.

Aicher, C., Ma, Y.-A., Foti, N. J. & Fox, E. B. (2019), 'Stochastic gradient MCMC for state space models', *SIAM Journal on Mathematics of Data Science* **1**(3), 555–587.

Aicher, C., Putcha, S., Nemeth, C., Fearnhead, P. & Fox, E. B. (2019), 'Stochastic Gradient MCMC for Nonlinear State Space Models', *arXiv:1901.10568* .

Andersen, M., Winther, O., Hansen, L. K., Poldrack, R. & Koyejo, O. (2018), Bayesian structure learning for dynamic brain connectivity, *in* 'International Conference on Artificial Intelligence and Statistics', pp. 1436–1446.

Baker, J., Fearnhead, P., Fox, E. B. & Nemeth, C. (2019*a*), 'Control variates for stochastic gradient MCMC', *Statistics and Computing* **29**(3), 599–615.

Baker, J., Fearnhead, P., Fox, E. & Nemeth, C. (2018), Large-scale stochastic sampling from the probability simplex, *in* 'Advances in Neural Information Processing Systems', pp. 6721–6731.

Baker, J., Fearnhead, P., Fox, E. & Nemeth, C. (2019*b*), 'sgmcmc: An R Package for Stochastic Gradient Markov Chain Monte Carlo', *Journal of Statistical Software, Articles* **91**(3), 1–27.

Balan, A. K., Rathod, V., Murphy, K. P. & Welling, M. (2015), Bayesian dark knowledge, *in* 'Advances in Neural Information Processing Systems', pp. 3438–3446.

Bardenet, R., Doucet, A. & Holmes, C. (2014), Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach, *in* 'International Conference on Machine Learning (ICML)', pp. 405–413.

Bardenet, R., Doucet, A. & Holmes, C. (2017), 'On Markov chain Monte Carlo methods for tall data', *The Journal of Machine Learning Research* **18**(1), 1515–1557.

Beck, A. & Teboulle, M. (2003), 'Mirror descent and nonlinear projected subgradient methods for convex optimization', *Operations Research Letters* **31**(3), 167–175.

Besag, J. (1994), 'Comments on Representations of knowledge in complex systems by U. Grenander and MI Miller', *Journal of the Royal Statistical Society, Series B* **56**, 591–592.

Bierkens, J., Fearnhead, P. & Roberts, G. O. (2019), 'The zig-zag process and super-efficient sampling for Bayesian analysis of big data', *The Annals of Statistics* **47**(3), 1288–1320.

Bishop, C. M. (2006), *Pattern recognition and machine learning*, Springer.

Blei, D. M., Kucukelbir, A. & McAuliffe, J. D. (2017), 'Variational inference: A review for statisticians', *Journal of the American Statistical Association* **112**(518), 859–877.

Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003), 'Latent dirichlet allocation', *Journal of machine Learning research* **3**, 993–1022.

Bouchard-Côté, A., Vollmer, S. J. & Doucet, A. (2018), 'The bouncy particle sampler: A nonreversible rejection-free Markov chain Monte Carlo method', *Journal of the American Statistical Association* **113**(522), 855–867.

Brooks, S., Gelman, A., Jones, G. & Meng, X.-L. (2011), *Handbook of Markov chain Monte Carlo*, CRC press.

Brooks, S. P. & Gelman, A. (1998), 'General methods for monitoring convergence of iterative simulations', *Journal of Computational and Graphical Statistics* **7**(4), 434–455.

Brosse, N., Durmus, A. & Moulines, É. (2018), The promises and pitfalls of Stochastic Gradient Langevin Dynamics, *in* 'Advances in Neural Information Processing Systems', pp. 8278–8288.

Brosse, N., Durmus, A., Moulines, É. & Pereyra, M. (2017), Sampling from a log-concave distribution with compact support with proximal Langevin Monte Carlo, *in* 'Conference on Learning Theory', pp. 319–342.

Bubeck, S., Eldan, R. & Lehec, J. (2018), 'Sampling from a log-concave distribution with Projected Langevin Monte Carlo', *Discrete & Computational Geometry* **59**(4), 757–783.

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

Chatterji, N., Flammarion, N., Ma, Y., Bartlett, P. & Jordan, M. (2018), On the theory of variance reduction for stochastic gradient monte carlo, Vol. 80 of *Proceedings of Machine Learning Research*, PMLR, pp. 764–773.

Chen, T., Fox, E. & Guestrin, C. (2014), Stochastic gradient Hamiltonian Monte Carlo, *in* 'International Conference on Machine Learning', pp. 1683–1691.

Cheng, X., Chatterji, N. S., Bartlett, P. L. & Jordan, M. I. (2018), Underdamped Langevin

MCMC: A non-asymptotic analysis, *in* S. Bubeck, V. Perchet & P. Rigollet, eds, 'Proceedings of the 31st Conference On Learning Theory', Vol. 75 of *Proceedings of Machine Learning Research*, PMLR, pp. 300–323.

Dalalyan, A. S. (2017), 'Theoretical guarantees for approximate sampling from smooth and log-concave densities', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **79**(3), 651–676.

Dalalyan, A. S. & Karagulyan, A. (2019), 'User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient', *Stochastic Processes and their Applications* **129**(12), 5278–5311.

de Valpine, P., Turek, D., Paciorek, C. J., Anderson-Bergman, C., Lang, D. T. & Bodik, R. (2017), 'Programming with models: writing statistical algorithms for general model structures with nimble', *Journal of Computational and Graphical Statistics* **26**(2), 403–413.

Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D. & Neven, H. (2014), Bayesian sampling using stochastic gradient thermostats, *in* 'Advances in Neural Information Processing Systems', pp. 3203–3211.

Dubey, K. A., Reddi, S. J., Williamson, S. A., Poczos, B., Smola, A. J. & Xing, E. P. (2016), Variance reduction in stochastic gradient Langevin dynamics, *in* 'Advances in Neural Information Processing Systems', pp. 1154–1162.

Dunson, D. B. & Johndrow, J. (2020), 'The Hastings algorithm at fifty', *Biometrika* **107**(1), 1–23.

Durmus, A. & Moulines, E. (2017), 'Nonasymptotic convergence analysis for the unadjusted Langevin algorithm', *The Annals of Applied Probability* **27**(3), 1551–1587.

Ermak, D. L. (1975), 'A computer simulation of charged particles in solution. I. Technique

and equilibrium properties', *The Journal of Chemical Physics* **62**(10), 4189–4196.

Fearnhead, P., Bierkens, J., Pollock, M. & Roberts, G. O. (2018), 'Piecewise deterministic Markov processes for continuous-time Monte Carlo', *Statistical Science* **33**(3), 386–412.

Fearnhead, P., Papaspiliopoulos, O. & Roberts, G. O. (2008), 'Particle filters for partially observed diffusions', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **70**(4), 755–777.

Gan, Z., Chen, C., Henao, R., Carlson, D. & Carin, L. (2015), Scalable deep Poisson factor analysis for topic modeling, *in* 'International Conference on Machine Learning', pp. 1823–1832.

Gibbs, A. L. & Su, F. E. (2002), 'On choosing and bounding probability metrics', *International statistical review* **70**(3), 419–435.

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

Gorham, J., Duncan, A. B., Vollmer, S. J., Mackey, L. et al. (2019), 'Measuring sample quality with diffusions', *The Annals of Applied Probability* **29**(5), 2884–2928.

Gorham, J. & Mackey, L. (2015), Measuring sample quality with Stein's method, *in* 'Advances in Neural Information Processing Systems', pp. 226–234.

Gorham, J. & Mackey, L. (2017), Measuring sample quality with kernels, *in* 'Proceedings of the 34th International Conference on Machine Learning-Volume 70', JMLR. org, pp. 1292–1301.

Hastings, W. K. (1970), 'Monte Carlo sampling methods using Markov chains and their applications', *Biometrika* **57**, 97–109.

Hoffman, M. D. & Gelman, A. (2014), 'The No-U-Turn sampler: adaptively setting

path lengths in Hamiltonian Monte Carlo', *Journal of Machine Learning Research* **15**(1), 1593–1623.

Hsieh, Y.-P., Kavis, A., Rolland, P. & Cevher, V. (2018), Mirrored Langevin Dynamics, *in* 'Advances in Neural Information Processing Systems', pp. 2883–2892.

Huggins, J. & Zou, J. (2017), Quantifying the accuracy of approximate diffusions and Markov chains, *in* 'Artificial Intelligence and Statistics', pp. 382–391.

Korattikara, A., Chen, Y. & Welling, M. (2014), Austerity in MCMC land: Cutting the Metropolis-Hastings budget, *in* 'International Conference on Machine Learning', pp. 181–189.

Kucukelbir, A., Ranganath, R., Gelman, A. & Blei, D. (2015), Automatic variational inference in stan, *in* 'Advances in Neural Information Processing Systems', pp. 568–576.

LeCun, Y., Cortes, C. & Burges, C. (2010), 'MNIST handwritten digit database', AT&T Labs [Online]. Available: `http://yann.lecun.com/exdb/mnist`.

Li, W., Ahn, S. & Welling, M. (2016), Scalable MCMC for mixed membership stochastic blockmodels, *in* 'Artificial Intelligence and Statistics', pp. 723–731.

Lunn, D. J., Thomas, A., Best, N. & Spiegelhalter, D. (2000), 'WinBUGS - a Bayesian modelling framework: concepts, structure, and extensibility', *Statistics and Computing* **10**(4), 325–337.

Ma, Y.-A., Chen, T. & Fox, E. (2015), A complete recipe for stochastic gradient MCMC, *in* 'Advances in Neural Information Processing Systems', pp. 2917–2925.

Ma, Y.-A., Foti, N. J. & Fox, E. B. (2017), 'Stochastic Gradient MCMC Methods for Hidden Markov Models', *arXiv:1706.04632* .

Majka, M. B., Mijatović, A., Szpruch, Ł. et al. (2020), 'Nonasymptotic bounds for sampling algorithms without log-concavity', *Annals of Applied Probability* **30**(4), 1534–1581.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. (1953), 'Equation of state calculations by fast computing machines', *The Journal of Chemical Physics* **21**(6), 1087–1092.

Meyn, S. P., Tweedie, R. L. et al. (1994), 'Computable bounds for geometric convergence rates of Markov chains', *The Annals of Applied Probability* **4**(4), 981–1011.

Minka, T. P. (2001), Expectation propagation for approximate Bayesian inference, *in* 'Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence', Morgan Kaufmann Publishers Inc., pp. 362–369.

Minsker, S., Srivastava, S., Lin, L. & Dunson, D. B. (2017), 'Robust and scalable Bayes via a median of subset posterior measures', *The Journal of Machine Learning Research* **18**(1), 4488–4527.

Nagapetyan, T., Duncan, A. B., Hasenclever, L., Vollmer, S. J., Szpruch, L. & Zygalakis, K. (2017), 'The true cost of stochastic gradient Langevin dynamics', arXiv:1706.02692.

Neal, R. M. (2011), MCMC using Hamiltonian dynamics, *in* S. Brooks, A. Gelman, G. L. Jones & X.-L. Meng, eds, 'Handbook of Markov chain Monte Carlo', CRC Press, pp. 113–162.

Neal, R. M. (2012), *Bayesian learning for neural networks*, Vol. 118, Springer Science & Business Media.

Neiswanger, W., Wang, C. & Xing, E. P. (2014), Asymptotically exact, embarrassingly parallel MCMC, *in* 'Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence', pp. 623–632.

Nemeth, C. & Sherlock, C. (2018), 'Merging MCMC subposteriors through Gaussian-process approximations', *Bayesian Analysis* **13**(2), 507–530.

Parisi, G. (1981), 'Correlation functions and computer simulations', *Nuclear Physics B*

180(3), 378–384.

Patterson, S. & Teh, Y. W. (2013), Stochastic gradient Riemannian Langevin dynamics on the probability simplex, *in* 'Advances in Neural Information Processing Systems', pp. 3102–3110.

Pillai, N. S., Stuart, A. M., Thiéry, A. H. et al. (2012), 'Optimal scaling and diffusion limits for the Langevin algorithm in high dimensions', *The Annals of Applied Probability* **22**(6), 2320–2356.

Plummer, M. (2003), JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling, *in* 'Proceedings of the 3rd International Workshop on Distributed Statistical Computing', Vol. 124, Vienna, Austria., p. 10.

Pollock, M., Fearnhead, P., Johansen, A. M. & Roberts, G. O. (2020), 'Quasi-stationary monte carlo and the scale algorithm', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* .

Quiroz, M., Kohn, R., Villani, M. & Tran, M.-N. (2018), 'Speeding up MCMC by efficient data subsampling', *Journal of the American Statistical Association* pp. 1–13.

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

Raginsky, M., Rakhlin, A. & Telgarsky, M. (2017), Non-convex learning via Stochastic Gradient Langevin Dynamics: a nonasymptotic analysis, *in* 'Conference on Learning Theory', pp. 1674–1703.

Raj, A., Stephens, M. & Pritchard, J. K. (2014), 'fastSTRUCTURE: variational inference of population structure in large SNP data sets', *Genetics* **197**(2), 573–589.

Ripley, B. D. (1987), *Stochastic simulation*, John Wiley & Sons.

Robbins, H. & Monro, S. (1951), 'A stochastic approximation method', *The Annals of*

Mathematical Statistics pp. 400–407.

Roberts, G. O. & 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.

Roberts, G. O. & Rosenthal, J. S. (2001), 'Optimal scaling for various Metropolis-Hastings algorithms', *Statistical Science* **16**(4), 351–367.

Roberts, G. O. & Rosenthal, J. S. (2004), 'General state space markov chains and mcmc algorithms', *Probability Surveys* **1**, 20–71.

Roberts, G. O. & Tweedie, R. L. (1996), 'Exponential convergence of Langevin distributions and their discrete approximations', *Bernoulli* **2**(4), 341–363.

Roberts, G. & Rosenthal, J. (1997), 'Geometric ergodicity and hybrid Markov chains', *Electronic Communications in Probability* **2**, 13–25.

Salakhutdinov, R. & Mnih, A. (2008), Bayesian probabilistic matrix factorization using Markov chain Monte Carlo, *in* 'Proceedings of the 25th international conference on Machine learning', ACM, pp. 880–887.

Salehi, F., Celis, L. E. & Thiran, P. (2017), 'Stochastic optimization with bandit sampling', *arXiv:1708.02544* .

Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I. & 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.

Sen, D., Sachs, M., Lu, J. & Dunson, D. B. (2020), 'Efficient posterior sampling for high-dimensional imbalanced logistic regression', *Biometrika* .

Srivastava, S., Li, C. & Dunson, D. B. (2018), 'Scalable Bayes via barycenter in Wasserstein space', *The Journal of Machine Learning Research* **19**(1), 312–346.

Teh, Y. W., Thiery, A. H. & Vollmer, S. J. (2016), 'Consistency and fluctuations for stochastic gradient Langevin dynamics', *The Journal of Machine Learning Research* **17**(1), 193–225.

Tran, D., Kucukelbir, A., Dieng, A. B., Rudolph, M., Liang, D. & Blei, D. M. (2016), 'Edward: A library for probabilistic modeling, inference, and criticism', *arXiv:1610.09787* .

Vollmer, S. J., Zygalakis, K. C. & Teh, Y. W. (2016), 'Exploration of the (non-) asymptotic bias and variance of stochastic gradient Langevin dynamics', *The Journal of Machine Learning Research* **17**(1), 5504–5548.

Wang, Y.-X., Fienberg, S. & Smola, A. (2015), Privacy for Free: Posterior Sampling and Stochastic Gradient Monte Carlo, *in* 'International Conference on Machine Learning', pp. 2493–2502.

Welling, M. & Teh, Y. W. (2011), Bayesian learning via stochastic gradient Langevin dynamics, *in* 'Proceedings of the 28th International Conference on Machine Learning (ICML)', pp. 681–688.

Yogatama, D., Wang, C., Routledge, B. R., Smith, N. A. & Xing, E. P. (2014), 'Dynamic language models for streaming text', *Transactions of the Association for Computational Linguistics* **2**, 181–192.

Zuanetti, D., Müller, P., Zhu, Y., Yang, S. & Ji, Y. (2019), 'Bayesian nonparametric clustering for large data sets', *Statistics and Computing* **29**, 203–215.

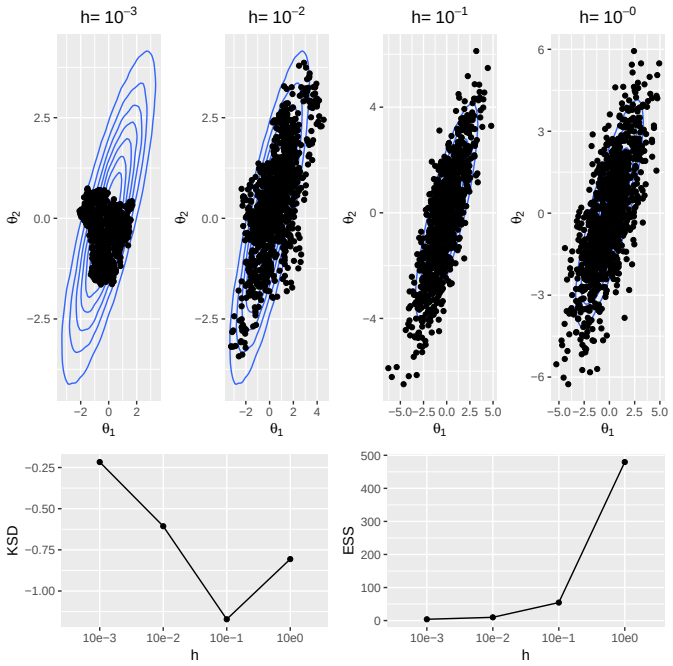

Figure 1: Top: Samples generated from the Langevin dynamics (7) are plotted over the bivariate Gaussian target. The samples are thinned to 1,000 for the ease of visualisation. Bottom: The kernel Stein discrepancy (log10) and effective sample size are calculated for each Markov chain with varying step size parameter h.

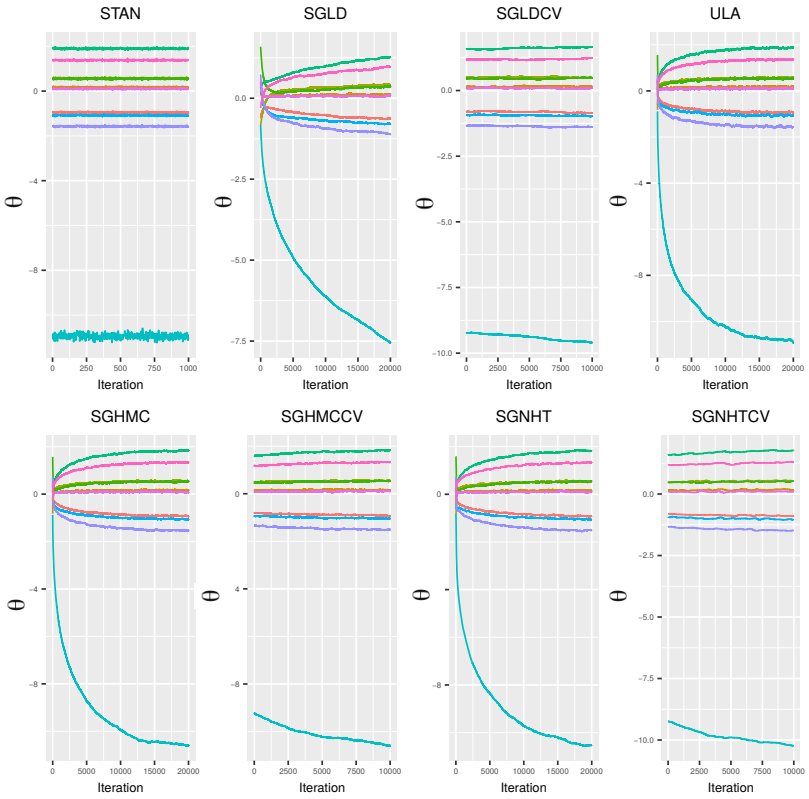

Figure 2: Trace plots for the STAN output and each SGMCMC algorithm with $d = 10$ and $N = 10^5$.

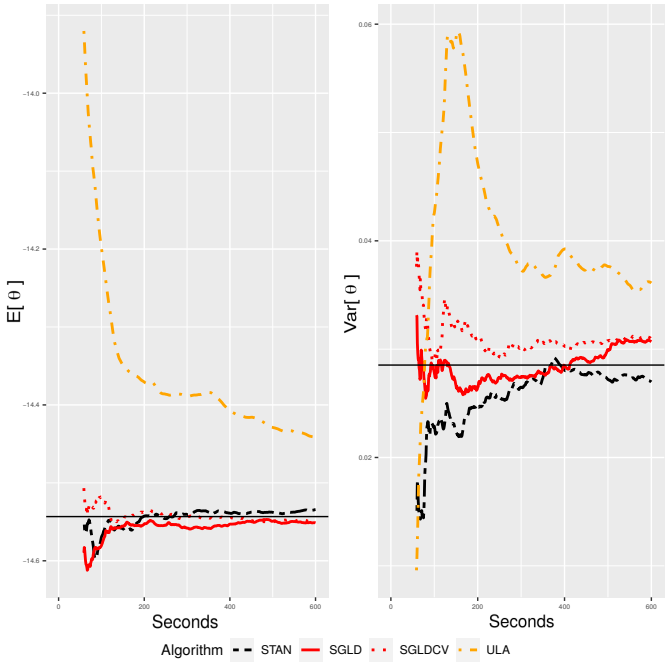

Figure 3: The mean and variance of the first parameter calculated at each second over 600 seconds, where $d = 10$ and $N = 10^5$.

Figure 4: Sample of images from the MNIST data set taken from `https://en.wikipedia.org/wiki/MNIST_database`

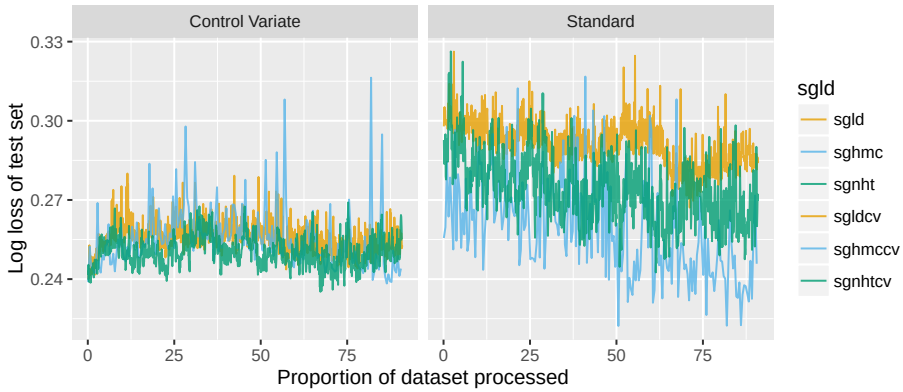

Figure 5: Log-loss calculated on a held-out test data set for each SGMCMC algorithm and its control variate version.

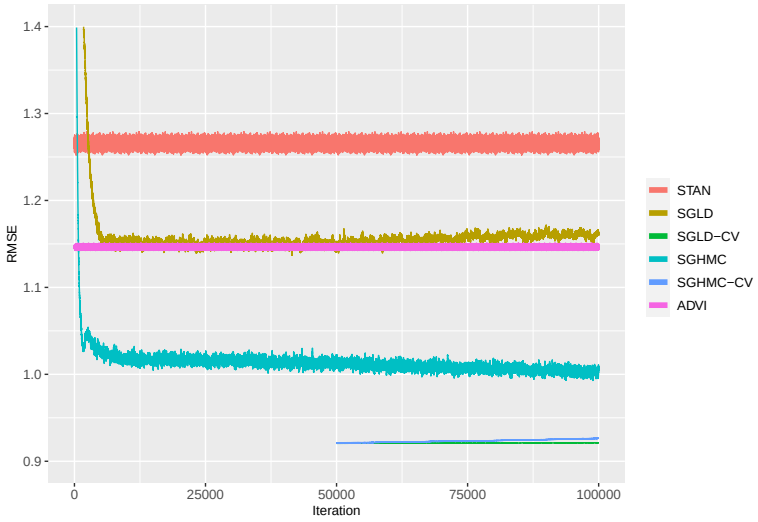

Figure 6: Root mean square error on the predictive performance of each SGMCMC algorithm averaged over five cross-validation experiments.

Table 1: A list of popular SGMCMC algorithms highlighting how they fit within the general stochastic differential equation framework (9)-(10). Most of the terms are defined in the text, except: $\mathbf{C} \succeq h\mathbf{V}(\boldsymbol{\theta})$, which is a positive semi-definite matrix; $G(\boldsymbol{\theta})$ is the Fisher information metric; A is a tuning parameter for SGNHT.

Algorithm	$\boldsymbol{\zeta}$	$H(\boldsymbol{\zeta})$	$\mathbf{D}(\boldsymbol{\zeta})$	$\mathbf{Q}(\boldsymbol{\zeta})$
SGLD	$\boldsymbol{\theta}$	$U(\boldsymbol{\theta})$	$\mathbf{I}$	$\mathbf{0}$
SGRLD	$\boldsymbol{\theta}$	$U(\boldsymbol{\theta})$	$G(\boldsymbol{\theta})^{-1}$	$\mathbf{0}$
SGHMC	$(\boldsymbol{\theta}, \boldsymbol{\rho})$	$U(\boldsymbol{\theta}) + \frac{1}{2}\boldsymbol{\rho}^\top\boldsymbol{\rho}$	$\begin{pmatrix} 0 & 0 \\ 0 & \mathbf{C} \end{pmatrix}$	$\begin{pmatrix} 0 & -\mathbf{I} \\ \mathbf{I} & 0 \end{pmatrix}$
SGRHMC	$(\boldsymbol{\theta}, \boldsymbol{\rho})$	$U(\boldsymbol{\theta}) + \frac{1}{2}\boldsymbol{\rho}^\top\boldsymbol{\rho}$	$\begin{pmatrix} 0 & 0 \\ 0 & G(\boldsymbol{\theta})^{-1} \end{pmatrix}$	$\begin{pmatrix} 0 & -G(\boldsymbol{\theta})^{-1/2} \\ G(\boldsymbol{\theta})^{-1/2} & 0 \end{pmatrix}$
SGNHT	$(\boldsymbol{\theta}, \boldsymbol{\rho}, \eta)$	$U(\boldsymbol{\theta}) + \frac{1}{2}\boldsymbol{\rho}^\top\boldsymbol{\rho}$ $+\frac{1}{2d}(\eta - A)^2$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & A \cdot \mathbf{I} & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & -\mathbf{I} & 0 \\ \mathbf{I} & 0 & \boldsymbol{\rho}^\top/d \\ 0 & -\boldsymbol{\rho}^\top/d & 0 \end{pmatrix}$

Table 2: Diagnostic metrics for each SGMCMC algorithm, plus STAN, with varying dimension of $\boldsymbol{\theta}$ where $N = 10^5$

	d	STAN	SGLD	SGLDCV	SGHMC	SGHMCCV	SGNHT	SGNHTCV	ULA	ADVI
Time (mins)	10	21.64	1.74	1.46	11.24	6.53	2.56	1.54	8.05	0.76
	50	157.24	2.55	2.06	13.43	7.76	3.33	1.93	29.21	2.37
	100	229.76	3.42	2.60	16.01	9.63	4.38	2.36	51.25	4.15
Log-loss	10	0.10	0.11	0.10	0.10	0.10	0.10	0.10	0.10	0.10
	50	0.04	0.06	0.05	0.06	0.05	0.05	0.05	0.05	0.07
	100	0.04	0.06	0.05	0.06	0.05	0.05	0.05	0.06	0.07
KSD	10	6.12	6.26	6.24	6.18	6.25	6.21	6.23	6.19	6.97
	50	9.24	11.73	11.05	11.59	11.11	11.00	11.33	11.30	11.66
	100	11.62	15.70	15.53	15.64	15.07	15.14	15.07	15.97	13.61