# Scalable Bayesian Inference Using Stochastic Gradient Markov Chain Monte Carlo

Srshti Putcha, B.Sc., M.Sc., M.Res


Submitted for the degree of Doctor of Philosophy

at Lancaster University.

March 2024

STOR-i

excellence with impact

*Dedicated to Mallik Tatagaru.*

" *In science, progress is possible. In fact, if one believes in Bayes' theorem, scientific progress is inevitable as predictions are made and as beliefs are tested and refined.*"

– Nate Silver

# Abstract

Bayesian inference offers a flexible framework to account for uncertainty across all unobserved quantities in a model. Markov chain Monte Carlo (MCMC) is a class of sampling algorithms which simulate from the Bayesian posterior distribution. These methods are generally regarded as the go-to computational technique for practical Bayesian modelling.

MCMC is well-understood, offers (asymptotically) exact inference, and can be implemented intuitively. Samplers built upon the Metropolis-Hastings algorithm can benefit from strong theoretical guarantees under reasonable conditions. Derived from discrete-time approximations of Itô diffusions, gradient-based samplers (Roberts and Rosenthal, 1998; Neal, 2011) leverage local gradient information in their proposal, allowing for efficient exploration of the posterior. The most championed of the diffusion processes are the overdamped Langevin diffusion and Hamiltonian dynamics.

In large data settings, standard MCMC can falter. The per-iteration cost of calculating the loglikelihood in the Metropolis-Hastings acceptance step scales with dataset size. Gradient-based samplers are doubly afflicted in this scenario, given that a full-data gradient is computed each iteration. These issues have prompted considerable interest in developing approaches for scalable Bayesian inference.

This thesis proposes novel contributions for stochastic gradient MCMC (Welling and Teh, 2011; Ma et al., 2015; Nemeth and Fearnhead, 2021). Stochastic gradient MCMC utilises data subsampling to construct a noisy, unbiased estimate of the gradient of the log-posterior.

The first two chapters review key background from the literature. Chapter 3 presents our first paper contribution. In this work, we extend stochastic gradient

MCMC to time series, via non-linear, non-Gaussian state space models. Chapter 4 presents the second paper contribution of this thesis. Here, we examine the use of a preferential subsampling distribution to reweight the stochastic gradient and improve variance control. Chapter 5 evaluates the feasibility of using determinantal point processes (Kulesza et al., 2012) for data subsampling in SGLD. We conclude and propose directions for future work in Chapter 6.

# Acknowledgements

I can wholeheartedly say that I wouldn't have gotten to this point without the army of people that surrounds me.

Thank you to my supervisors, Chris and Paul. You have been unfailingly patient with me for the past five-and-a-half years. And I know that I haven't been the easiest (or communicative) student to supervise. You've taught me how to question things from every conceivable angle. I'll never look at a technical problem in quite the same way.

Thank you to the STOR-i leadership team, Jon and Idris. This programme has given me the opportunity to succeed and fail in a safe environment. I hope that I am a better statistician (or rather, data scientist) for it. I'd be remiss if I didn't also thank Jen, Kim, Wendy and Nicky for facilitating everything at STOR-i. The place probably would have fallen apart without you.

To the 2017 STOR-i cohort (the Tigersharks), I'm grateful to have started this process with you. I'll always recall our MRes antics with happiness. I wish you the best with your future endeavours. I hope we keep in touch for many years to come.

To the COVID-19 pandemic, you may have suspended all sense of normalcy and structure for over two years, but you taught me the true meaning of perseverance and the importance of prioritising my own mental health. I'm glad to have finally made it to the finish line at the eleventh hour.

To Hankui, I'm glad to have found a friend in you. Thank you for many delightful evenings in your company, the many pandemic walks, and video calls. To Zak, thank you for commiserating and celebrating in equal measure. Your friendship has been invaluable over the past nine years.

To Anja and Holly, thank you for sitting beside me (physically and spiritually) for the past six years. I could never retreat too far into my shell with the two of you checking up on me. I look forward to going on many more adventures with you.

Lastly, and most importantly, thank you to my wonderful family. Amma, Nanna and Aditya - I don't have enough words to express my gratitude or appreciation. However, I do know that this degree would have been impossible without your unyielding support and love. You make me feel brave.

# Declaration

I declare that the work in this thesis has been done by myself and has not been submitted elsewhere for the award of any other degree.

A version of Chapter 3 is now available as an advanced publication in the journal Bayesian Analysis as,
Aicher, C., Putcha, S., Nemeth, C., Fearnhead, P., and Fox, E. B. (2023). Stochastic Gradient MCMC for Nonlinear State Space Models. In *Bayesian Analysis*, Advance Publication 1-23. DOI: 10.1214/23-BA1395.
The preprint of this work is listed on arXiv under *arXiv preprint arXiv:1901.10568.*

A version of Chapter 4 appears in the conference proceedings of AISTATS 2023 as, Putcha, S., Nemeth, C., and Fearnhead, P (2023). Preferential Subsampling for Stochastic Gradient Langevin Dynamics. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*.

The word count stands at 46,590 words.

Srshti Putcha

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Markov chain Monte Carlo (MCMC) algorithms are a popular class of methods used to conduct inference for Bayesian models. A key drawback of MCMC is that it tends to scale poorly with dataset size. Over time, this has become a practical issue: helped in no part by the growing need in both statistics and machine learning to process larger datasets and to construct increasingly more complex models. This thesis provides methodological contributions for stochastic gradient Markov chain Monte Carlo, a family of approximate samplers that utilises data subsampling techniques to reduce the per-iteration computational cost of MCMC.

In this chapter, we provide a high-level overview of Bayesian inference, its associated computational methods, and the challenges it presents for scalability. This will be built upon further in the literature review presented in Chapter 2. We then proceed to outline the main contributions of this thesis and the contents of the subsequent chapters.

## 1.1 Background

Bayesian inference lays out a framework for fitting a probability model to a dataset and summarises the result in the form of a probability distribution on the model parameters and on other unobserved quantities (e.g., predictions on future observations) (Gelman et al., 2013). Let us consider the unknown model parameters, $\theta$, as random and the

data, $\mathbf{x}$, as fixed. We assume that $\theta \in \Theta \subseteq \mathbb{R}^d$ for the sake of simplicity[1].

In order to formally conduct inference, we need to specify a full probability model over all observable and unobservable quantities. The joint distribution of interest comprises of the prior, $p(\theta)$, and the likelihood, $p(\mathbf{x}|\theta)$. Thus, the full probability model is given by the density function

$$p(\theta, \mathbf{x}) = p(\theta)p(\mathbf{x}|\theta).$$

Our aim is to have a model that encapsulates everything we know about the underlying problem and the data collection process (Gilks et al., 1996; Gelman et al., 2013).

Suppose that the data that we then collect is of size $N$, such that $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$. The density function of the $i$-th datapoint is given by $p(x_i|\theta)$. If we further assume that the data are independent, the likelihood can be written as $p(\mathbf{x}|\theta) = \prod_{i=1}^{N} p(x_i|\theta)$. We can leverage Bayes' theroem to condition on the observed data and obtain the posterior distribution. The posterior density on $\theta$ is defined as

$$\pi(\theta) := p(\theta|\mathbf{x}) = \frac{p(\theta)p(\mathbf{x}|\theta)}{\int_\Theta p(\theta)p(\mathbf{x}|\theta)\, d\theta}.$$

Here, the denominator $m(\mathbf{x}) = \int_\Theta p(\theta)p(\mathbf{x}|\theta)d\theta$ is known as the marginal likelihood or the evidence. The posterior distribution represents our updated beliefs about the model parameters and is the central focus of any Bayesian data analysis (Robert and Casella, 2004; Gelman et al., 2013).

If the marginal likelihood can be calculated, then the posterior density can be evaluated analytically. However, a fundamental issue within Bayesian inference is that the integration required to compute $m(\mathbf{x})$ is rarely analytically tractable. This means that we usually only know the posterior up to the unnormalised density $h(\theta) = p(\theta)p(\mathbf{x}|\theta)$. We can use computational methods like MCMC to simulate from the posterior exactly using the unnormalised density $h$, thereby bypassing the need to calculate $m(\mathbf{x})$ exactly.

Many of the features of the posterior distribution that we are interested in (e.g., moments, quantiles) can be expressed as an expectation of functions of $\theta$ with respect

[1]We loosen this assumption in Chapter 2 to consider a general parameter space, $\Theta$.

to $\pi$. The posterior expectation of a measurable real-valued function, $\psi : \Theta \to \mathbb{R}^d$, is given by,

$$\mathbb{E}_\pi[\psi(\theta)] = \int_\Theta \psi(\theta)\pi(d\theta).$$

As before, it can often be the case that we cannot obtain a closed-form value for expectations of this form. This quantity can be instead estimated using the Monte Carlo estimate

$$\mathbb{E}_\pi[\psi(\theta)] \approx \frac{1}{M}\sum_{m=1}^{M} \psi\big(\theta^{(m)}\big)$$

with a set of $M$ MCMC samples $\{\theta^{(1)}, \ldots, \theta^{(M)}\}$.

## 1.2 Problem Motivation

As the availability of larger datasets increases, using MCMC to evaluate posterior summaries becomes impractical. MCMC requires the calculation of the unnormalised density $h$ at each iteration and this is an $O(N)$ calculation. This issue has led to the development of new methodology that seeks to improve the computational efficiency of MCMC.

Typically, scalable Monte Carlo techniques efficiently process large volume of data by either subsampling the data (Doucet et al., 2015; Maclaurin and Adams, 2015; Bierkens et al., 2019; Quiroz et al., 2019) or by running samplers in parallel across multiple cores (à la Scott et al., 2016; Nemeth and Sherlock, 2018; Vyner et al., 2022). We refer the reader to the Bardenet et al. (2017) review paper on MCMC approaches for tall datasets.

One popular class of algorithms are the stochastic gradient MCMC methods (Nemeth and Fearnhead, 2021). These samplers are developed using discrete-time Euler approximations of Itô diffusion processes that admit the posterior as their invariant distribution. One such well-studied process is the overdamped Langevin diffusion (Roberts and Tweedie, 1996).

The unadjusted Langevin algorithm (ULA) is obtained by simply simulating from the Euler discretisation of the Langevin diffusion. Due to its discretisation error,

ULA merely approximates the posterior at stationarity (adding a Metropolis-Hastings correction gives us the Metropolis-adjusted Langevin algorithm, see e.g., Roberts and Rosenthal, 1998). Furthermore, ULA struggles to scale well to tall datasets as it requires an $O(N)$ log-posterior gradient calculation at each iteration.

Inspired by stochastic optimisation (Robbins and Monro, 1951; Bottou et al., 2018), Welling and Teh (2011) proposed stochastic gradient Langevin dynamics (SGLD). The idea behind SGLD being that the full-data gradient calculation in the ULA sampler could be replaced with an unbiased, noisy estimate that was constructed using a random data subsample. Following this, other samplers using Itô diffusions were also proposed (Ma et al., 2015), including Hamiltonian dynamics (Chen et al., 2014)

Stochastic gradient MCMC can be applied to a broad class of models and, in the simplest case, only require a first-order gradient computation. The main drawback of these samplers is that the inference is no longer exact with respect to $\pi$.

This class of methods have been adapted for various disciplines in machine learning, such as deep generative models (Du et al., 2018), differential privacy (Li et al., 2019) and meta-learning (Gong et al., 2019). Some exciting application areas have been considered, including quantum optimisation (Patti et al., 2022) and additive manufacturing (Chung et al., 2022). The core samplers have also been included in open-source packages for both R and Python (Baker et al., 2019b; Coullon and Nemeth, 2022), with the goal of making the implementation of these methods more accessible to general practitioners.

## 1.3 Thesis Outline

This thesis focuses on two open areas of stochastic gradient MCMC research: improving variance control and extending the class of algorithms to dependent datasets. The main content of this thesis has been divided into four chapters. Chapter 2 contains a literature review of the technical background relevant to stochastic gradient MCMC. We present new research that has been submitted for publication in Chapters 3 and 4. Finally, Chapter 5 assesses the viability of using determinantal point processes in

SGLD to construct informed data subsamples for variance control. Chapter 6 reviews the main contributions of the thesis and then discusses areas for future work. An outline of each thesis chapter is given below.

**Chapter 2: Monte Carlo methods**

This chapter provides an overview of stochastic gradient MCMC. The chapter first covers standard MCMC and summarises the approaches available for scalable Monte Carlo. The next section reviews the fundamentals of stochastic gradient Langevin dynamics, its associated extensions and the broader family of stochastic gradient MCMC samplers. In order to lay the groundwork for Chapter 3, a final section is dedicated to reviewing existing inference approaches for state space models.

**Chapter 3: Stochatic Gradient MCMC for Nonlinear State Space Models**

*A version of this chapter has been released as an Advance Publication in the journal Bayesian Analysis with co-authors Christopher Aicher, Christopher Nemeth, Paul Fearnhead and Emily Fox. The abstract of the article is given below.*

State space models (SSMs) provide a flexible framework for modeling complex time series via a latent stochastic process. Inference for nonlinear, non-Gaussian SSMs is often tackled with particle methods that do not scale well to long time series. The challenge is two-fold: not only do computations scale linearly with time, as in the linear case, but particle filters additionally suffer from increasing particle degeneracy with longer series. Stochastic gradient MCMC methods have been developed to scale Bayesian inference for finite-state hidden Markov models and linear SSMs using buffered stochastic gradient estimates to account for temporal dependencies. We extend these stochastic gradient estimators to nonlinear SSMs using particle methods. We present error bounds that account for both buffering error and particle error in the case of nonlinear SSMs that are log-concave in the latent process. We evaluate our proposed particle buffered stochastic gradient using stochastic gradient MCMC for inference on both long sequential synthetic and minute-resolution financial returns data, demonstrating the importance of this class of methods.

**Chapter 4: Preferential Subsampling for Stochastic Gradient Langevin Dynamics**

*A version of this chapter has been published as part of the conference proceedings of AISTATS 2023. It has been written with co-authors Christopher Nemeth and Paul Fearnhead. The abstract of the paper is given below.*

Stochastic gradient MCMC (SGMCMC) offers a scalable alternative to traditional MCMC, by constructing an unbiased estimate of the gradient of the log-posterior with a small, uniformly-weighted subsample of the data. While efficient to compute, the resulting gradient estimator may exhibit a high variance and impact sampler performance. The problem of variance control has been traditionally addressed by constructing a better stochastic gradient estimator, often using control variates. We propose to use a discrete, non-uniform probability distribution to preferentially subsample data points that have a greater impact on the stochastic gradient. In addition, we present a method of adaptively adjusting the subsample size at each iteration of the algorithm, so that we increase the subsample size in areas of the sample space where the gradient is harder to estimate. We demonstrate that such an approach can maintain the same level of accuracy while substantially reducing the average subsample size that is used.

**Chapter 5: A Feasibility Study - Utilising Determinantal Point Processes for Subsampling**

Inspired by the work of Zhang et al. (2017a), this chapter seeks to evaluate the feasibility of using determinantal point processes (DPPs) to construct low-variance, diverse data subsamples for SGLD. We introduce discuss key DPP concepts and then provide a worked example on synthetic data. We conclude by discussing the practical considerations associated with the proposed approach.

# Chapter 2

# Monte Carlo methods

Many problems in statistics and machine learning rely upon the computation of the expectation of some random quantity with respect to a probability distribution. If these expectations cannot be calculated directly, we must seek out some kind of numerical approximation. Monte Carlo methods construct an unbiased approximation of expectations by simulating draws from the probability distribution of interest.

In Section 2.1, we outline Monte Carlo integration. While perfect Monte Carlo goes some way to helping us to calculate expectations, we need something more powerful when we cannot simulate directly from the underlying probability distribution. Section 2.2 introduces Markov chain Monte Carlo (MCMC). Often considered the gold standard of Bayesian computational methods, MCMC is used to conduct inference on complex and hierarchical models in a variety of domains. Section 2.3 discusses stochastic gradient MCMC and its associated extensions. Finally, we spend some time reviewing existing Bayesian inference approaches for state space models.

We note that Monte Carlo integration is a vast area in computational statistics and so, only the topics necessary for this thesis are reviewed in this chapter. For a more rigorous treatment of traditional MCMC, please refer to either Meyn and Tweedie (1993c) or Robert and Casella (2004). A more practical take on MCMC can be found in Gelman et al. (2013). The more technical underpinnings of Itô diffusions and their numerical approximations can be studied further in Kloeden and Platen (1995); Øksendal (2003) and Khasminskii (2011). We also recommend seeking out Nemeth

and Fearnhead (2021) for a comprehensive review of stochastic gradient MCMC.

## 2.1 Monte Carlo

A central focus of Bayesian computation is the accurate and efficient calculation of integrals. More specifically, we are often concerned with obtaining the expectation of a function. Let $\theta$ be a random variable with distribution $\pi$ over some measurable space, $\Theta$ (such that $\mathbb{P}(\theta \in A) = \pi(A)$). The Borel $\sigma$-field for $\Theta$ is denoted as $\mathcal{B}(\Theta)$.[1]

Suppose that we are interested in integrating a measurable function $\psi : \Theta \to \mathbb{R}^d$ with respect to $\pi$. The expectation of the test function $\psi$ is defined as,

$$\mathbb{E}_\pi[\psi(\theta)] = \int_\Theta \psi(\theta)\pi(d\theta). \tag{2.1.1}$$

The key issue that needs to be overcome is that expectations of the form of (2.1.1) can rarely be calculated analytically. In this case, we typically turn to numerical integration methods to approximate (2.1.1).

Deterministic numerical integration methods (such as Simpson's Rule) evaluate the integrand at $M$ selected points on a grid and compute a weighted version of (2.1.1), with the weights corresponding to the volume of space represented by each of the grid points (Gelman et al., 2013). These techniques can be implemented easily in univariate or bivariate settings, but they scale poorly in dimension $d$ with $O(M^d)$ evaluations required over the grid points. Simulation-based (stochastic) methods, such as Monte Carlo integration, offer us an alternative path forward.

### Perfect Monte Carlo

In classical Monte Carlo, we assume that it is possible to simulate a set of independent and identically distributed (i.i.d) draws $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(M)}\}$ from $\pi$. We can use these samples to construct the empirical average,

$$\hat{\psi}_M = \frac{1}{M}\sum_{m=1}^{M} \psi\big(\theta^{(m)}\big). \tag{2.1.2}$$

[1] We use some light measure-theoretic concepts in Chapter 2 to communicate key ideas, but this thesis will not have a heavy focus on measure theory.

The Monte Carlo estimate, $\hat{\psi}_M$, possesses a number of advantageous statistical properties. By the Strong Law of Large Numbers, we know that as the number of samples $M \to \infty$, $\hat{\psi}_M$ converges almost surely to $\mathbb{E}_\pi[\psi(\theta)]$, i.e.,

$$\hat{\psi}_M \xrightarrow{a.s} \mathbb{E}_\pi[\psi(\theta)], \quad \text{as} \quad M \to \infty.$$

Furthermore, when $\text{Var}[\psi(\theta)] = \sigma^2 < \infty$, an application of the Central Limit theorem tells us that

$$\sqrt{M}(\hat{\psi}_M - \mathbb{E}_\pi[\psi(\theta)]) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \sigma^2), \quad \text{as } M \to \infty.$$

We know that the approximation error of the Monte Carlo estimate given in (2.1.2) is $O(M^{-\frac{1}{2}})$, independent of the dimension of the integrand, $d$. Any other deterministic numerical integration method would have an approximation error that increases as the dimension $d$ grows (Doucet et al., 2001; Robert and Casella, 2004).

It may seem that the method proposed above is sufficient to approximate (2.1.1) in a controlled, yet straightforward manner. However, it will not always be possible to directly simulate from $\pi$, particularly in complex, high-dimensional cases. If the density $\pi(\theta)$ can be evaluated pointwise (at least up to a constant of proportionality), we can instead construct our Monte Carlo estimate by sampling from an alternative proposal distribution, $q$. Importance sampling and rejection sampling are two such methods where this is done.

**Rejection sampling**

The premise of rejection sampling is simple. We require a density function $q(\theta)$ (defined such that $q(\theta) > 0$ whenever $\pi(\theta) > 0$) that possesses the following properties (Gelman et al., 2013):

- (RS1) We can draw from the probability density proportional to $q(\theta)$. It is not mandatory for $q(\theta)$ to integrate to 1, but it must have a finite integral.
- (RS2) There must exist a known constant $C$ for which $\frac{\pi(\theta)}{q(\theta)} \leq C$ for all $\theta$.

At each iteration, we simulate a sample $\theta' \sim q$ and a corresponding uniform random number, $u \sim \mathcal{U}(0, 1)$. The new draw, $\theta'$, is only accepted if $u \leq \frac{\pi(\theta')}{Cq(\theta')}$, where $C$ satisfies

(RS2). In this way, those samples that are more likely to have come from the target density are kept and the others are discarded (von Neumann, 1951; Robert and Casella, 2004). This approach can be quite wasteful, given that only a subset of the proposed draws are kept to construct $\hat{\psi}_M$.

Importance sampling, in contrast, makes use of all generated samples when forming the Monte Carlo estimate. This method also adapts well to recursive estimation problems, as discussed further in Chapter 3.

**Importance sampling**

Importance sampling constructs an estimate of (2.1.1) based on generating a set of draws $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(M)}\}$ from a given distribution $q$ and approximating,

$$\mathbb{E}_\pi[\pi(\theta)] \approx \frac{1}{M} \sum_{m=1}^{M} \frac{\pi\big(\theta^{(m)}\big)}{q\big(\theta^{(m)}\big)} \psi\big(\theta^{(m)}\big). \tag{2.1.3}$$

This method is based on the rewriting (2.1.1) as

$$\mathbb{E}_\pi[\psi(\theta)] = \int_\Theta \psi(\theta) \frac{\pi(\theta)}{q(\theta)} q(d\theta). \tag{2.1.4}$$

As long as $q$ is chosen with appropriate support (i.e., $q(\theta) > 0$ must imply that $\pi(\theta) > 0$), the estimate (2.1.3) will converge almost surely to (2.1.1) (Robert and Casella, 2004). However, the variance of (2.1.3) is finite only when the expectation

$$\mathbb{E}_q\Big[\psi^2(\theta) \frac{\pi^2(\theta)}{q^2(\theta)}\Big] = \mathbb{E}_\pi\Big[\psi^2(\theta) \frac{\pi(\theta)}{q(\theta)}\Big] = \int_\Theta \psi^2(\theta) \frac{\pi(\theta)}{q(\theta)} \pi(d\theta) < \infty.$$

Proposals with tails that are lighter than those of $\pi$ are not well-suited to importance sampling, as then the ratio $\pi/q$ can explode as we move into the tails of $\pi$. In this scenario, the importance weights $\pi\big(\theta^{(m)}\big)/q\big(\theta^{(m)}\big)$ would vary considerably, giving too much significance to a handful of values $\theta^{(m)}$.

## 2.2 Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods offer an alternative approach to the Monte Carlo methods discussed above. By constructing a stochastic process that

converges to $\pi$, we can generate samples from the desired distribution by observing the chain after an initial burn-in period. A key drawback of MCMC is that the draws obtained from the stochastic process are no longer independent. However, alternative asymptotic convergence results exist for these methods (Meyn and Tweedie, 1993c; Robert and Casella, 2004). In this section, we provide a brief survey of Markov chains and stochastic stability, outline the Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970) and discuss the scalability issues for MCMC on tall data.

### 2.2.1 Markov chain preliminaries

A Markov chain is a discrete-time stochastic process, $\left(\theta^{(m)}\right)_{m\geq 0}$, taking values on an arbitrary state space $\Theta$ (Robert and Casella, 2004; Geyer, 2005). Formally, the Markov chain can be described in terms of its conditional distribution,

$$\mathbb{P}\big(\theta^{(m)} \in A \,|\, \theta^{(m-1)}, \ldots, \theta^{(0)}\big) = \mathbb{P}\big(\theta^{(m)} \in A \,|\, \theta^{(m-1)}\big),$$

where $A \in \mathcal{B}(\Theta)$. Here, the Markov property specifies that the conditional distribution of $\theta^{(m)}$ given $\theta^{(0)}, \theta^{(1)}, \ldots, \theta^{(m-1)}$ is the same as the conditional distribution of $\theta^{(m)}$ given $\theta^{(m-1)}$. We assume that all Markov chains are time-homogeneous and do not depend on the current value of $m$.

For notational simplicity, it is cleaner to define a Markov chain in terms of its transition kernel, a function that regulates how the chain evolves over time. The transition kernel is a function $K(\cdot, \cdot)$ on $\Theta \times \mathcal{B}(\Theta)$ such that: (i) $K(\theta, \cdot)$ is a probability measure $\forall \theta \in \Theta$, (ii) $K(\cdot, A)$ is measurable $\forall A \in \mathcal{B}(\Theta)$. If the state space $\Theta$ is discrete, the transition kernel reduces to a (transition) matrix $K$ containing

$$P_{\theta,\theta'} = \mathbb{P}\big(\theta^{(m)} = \theta \,|\, \theta^{(m-1)} = \theta'\big),\ \theta, \theta' \in \Theta.$$

For a general state space, the conditional probability of $\theta^{(m)}$ given $\theta^{(m-1)}$ is given by,

$$\mathbb{P}\big(\theta^{(m)} \in A \,|\, \theta^{(m-1)}\big) = \int_A K\big(\theta^{(m-1)}, d\theta\big).$$

The transition kernel for $m$ transitions ($m > 1$) of the chain is denoted as,

$$K^m(\theta, A) = \mathbb{P}\big(\theta^{(m)} \in A \,|\, \theta^{(0)} = \theta\big) = \int_\Theta K^{m-1}(\theta', A) K(\theta, d\theta'),$$

with $K^1(\theta, A) = K(\theta, A)$.

### Stochastic stability and convergence

We wish to ultimately construct Markov chains that converge to our chosen target $\pi$ over time. For this to be possible, we first need the target distribution to be invariant, such that the marginal distribution of $(\theta_m)_{m\geq 0}$ does not depend on $m$ (Robert and Casella, 2004). We know that an invariant measure $\pi$ exists for the kernel $K(\cdot,\cdot)$ and its associated chain if

$$\pi(B) = \int_{\Theta} K(\theta, B)\pi(d\theta), \;\; \forall B \in \mathcal{B}(\Theta).$$

The invariant distribution is also referred to as stationary if $\pi$ is a probability measure, since $\theta^{(0)} \sim \pi$ implies that $\theta^{(m)} \sim \pi$ for any value of $m$. Furthermore, if the following condition of detailed balance or reversiblity holds,

$$\int_A \pi(d\theta)K(\theta, B) = \int_B \pi(d\theta)K(\theta, A), \tag{2.2.1}$$

for every $A, B \in \mathcal{B}(\Theta)$ then a Markov chain with kernel $K$ has invariant distribution $\pi$ (Meyn and Tweedie, 1993c; Geyer, 2005). Another common shorthand for (2.2.1) is,

$$\pi(d\theta)K(\theta, d\theta') = \pi(d\theta')K(\theta', d\theta). \tag{2.2.2}$$

Even if a Markov chain has a stationary distribution, it may still fail to converge to stationarity as $m \to \infty$. To address this, there are two key things that we need to ensure: (i) the chain is able to converge to stationarity, and (ii) the stationary distribution of the chain is uniquely the target of interest, $\pi$.

Given a nonzero measure $\phi$, a Markov chain is $\phi$-irreducible if for any point $\theta$ and any measurable set $A$ where $\phi(A) > 0$, there exists an integer $m$ such that $K^m(\theta, A) > 0$ (Robert and Casella, 2004; Geyer, 2005). In other words, $\phi$-irreducibility ensures that there is a positive probability that for any starting state the chain will reach any set $A$ having positive measure in finite time.

A Markov chain is Harris recurrent if there exists a nonzero measure $\phi$ such that for any set $A$ with positive measure and any starting point $\theta^{(0)} \in \Theta$, we have that $\sum_{m=1}^{\infty} K^m(\theta^{(0)}, A) > 0$ (Robert and Casella, 2004; Geyer, 2005). Namely, the chain is guaranteed to occupy $A$ and the probability $\theta^{(m)}$ started from $\theta^{(0)}$ occupies $A$ infinitely often is 1.

The state space of any Harris recurrent Markov chain can be partitioned into sets $\{B_i\}_{i=1}^{b}$ and $N$ such that $\phi(N) = 0$ and $K(\theta, B_i) = 1$ when $\theta \in B_j$ for $j = i - 1 \bmod b$. The chain is then said to aperiodic if $b = 1$ and periodic if $b > 1$ (Meyn and Tweedie, 1993c; Geyer, 2005). Aperiodicity removes the possibility of indefinite oscillatory behaviour occurring within the chain over time (Roberts and Rosenthal, 2004).

If all three of these properties - irreducibility, aperiodicity and Harris recurrence - hold, we can prove the existence (and subsequent uniqueness) of the stationary distribution. We can also establish several desirable results (such as a Law of Large Numbers) similar to those presented for Monte Carlo methods in Section 2.1.

At this stage, it is natural to consider a framework for quantifying how quickly a Markov chain converges - namely, its rate of convergence. We first need to outline the total variation norm, which is typically used to calculate the distance between two probability measures. Given two measures $\mu_1$ and $\mu_2$, the total variation norm is given by,

$$\|\mu_1 - \mu_2\|_{TV} = \sup_A |\mu_1(A) - \mu_2(A)|,$$

for all measurable sets $A$ (Robert and Casella, 2004).

An aperiodic, Harris recurrent Markov chain $(\theta_m)_{m \geq 0}$ is said to be ergodic if it converges in total variation to $\pi$ given any initial distribution $\lambda$ (Meyn and Tweedie, 1993c; Geyer, 2005), i.e.,

$$\|\lambda K^m - \pi\|_{TV} = \left\| \int \lambda(d\theta) K^m(\theta, \cdot) - \pi \right\|_{TV} \xrightarrow{m \to \infty} 0.$$

In the case where $\lambda$ is a measure concentrated at the point $\theta$, this reduces to,

$$\|K^m(\theta, \cdot) - \pi\|_{TV} \xrightarrow{m \to \infty} 0. \tag{2.2.3}$$

Geometric ergodicity can be established when the convergence in (2.2.3) occurs at a geometric rate (Geyer, 2005). This occurs when there exists a constant $\rho > 1$ and a bounded function $\beta : \Theta \to \mathbb{R}^+$ such that,

$$\|K^m(\theta, \cdot) - \pi\|_{TV} \leq \beta(\theta) \rho^m.$$

A stronger version of geometric ergodicity can be used for a Markov chain that is Harris recurrent with invariant distribution $\pi$, that is when there is a constant $r > 1$,

$$\sum_{m=1}^{\infty} r^m \|K^m(\theta, \cdot) - \pi\|_{TV} < \infty, \tag{2.2.4}$$

for all $\theta \in \Theta$ (Meyn and Tweedie, 1993c). When the bound in (2.2.4) is uniform, that is if there is a finite constant $R$ such that,

$$\sum_{m=1}^{\infty} r^m \|K^m(\theta, \cdot) - \pi\|_{TV} < R,$$

for all $\theta$, we say that the Markov chain is uniformly ergodic. Therefore, uniform ergodicity implies geometric ergodicity. Furthermore, uniform ergodicity implies that, as $m \to \infty$,

$$\sup_{\theta \in \Theta} \|K^m(\theta, \cdot) - \pi\|_{TV} \to 0.$$

This kind of quantitative analysis is used widely to understand the properties of MCMC algorithms. A large emphasis is placed in the MCMC literature on deriving speed of convergence results with respect to the total variation norm. Often, this is not the most user-friendly or practical distance measure to use. In Section 2.3.2, we shall see convergence results proposed in terms of the Wasserstein distance metric.

### 2.2.2 Metropolis-Hastings algorithm

In many cases, the only thing we know about the target $\pi$ is an unnormalised density. A function $h : \Theta \to \mathbb{R}$ is an unnormalised density with respect to a Lebesgue measure $\mu$, if it is nonnegative, and $0 < \int_\Theta h(\theta) d\mu < \infty$ (Geyer, 2005). In a Bayesian paradigm, the marginal likelihood, $m(\mathbf{x})$, is rarely tractable. This means that we only know the posterior up to an unnormalised density $h(\theta) = p(\theta)p(\mathbf{x} \,|\, \theta)$.

The Metropolis-Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970) navigates this issue by constructing a Markov chain that simulates from the target distribution, $\pi$, and only requires the ability to evaluate the unnormalised density, $h$. It is not necessary to be able to calculate any integrals or know the value of the normalising constant.

The MH update instead leverages an auxiliary proposal density, $q(\cdot, \cdot)$ to propose new moves. For each point $\theta \in \Theta$, $q(\theta, \cdot)$ must be a normalised probability density with respect to the dominating measure $\mu$ with the following two properties: (i) for each $\theta$ we can simulate $\theta' \sim q(\theta, \cdot)$; and (ii) for each $\theta, \theta'$ we can evaluate $q$ pointwise.

Typically, the proposal density is either explicitly defined or symmetric, such that $q(\theta, \theta') = q(\theta', \theta)$ (e.g., a multivariate Normal). Given the current state, a new sample $\theta'$ is drawn from $q(\theta, \cdot)$ and is either accepted with probability

$$\alpha(\theta, \theta') = \min\left\{1, \frac{h(\theta')q(\theta', \theta)}{h(\theta)q(\theta, \theta')}\right\},$$

or rejected. The full method is provided below in Algorithm 1.

**Algorithm 1:** Metropolis-Hastings

1: Input: initialise $\theta^{(0)}$.
2: **for** $m = 1, 2, \ldots, M$ **do**
3: Given $\theta^{(m)}$, generate $\theta' \sim q\big(\theta^{(m)}, \cdot\big)$.
4: Calculate the acceptance probability, $\alpha\big(\theta^{(m)}, \theta'\big)$.
5: Accept $\theta'$ with probability $\alpha$ and set $\theta^{(m+1)} = \theta'$ otherwise.
6: **end for**

Common implementations of the MH algorithm include:

- the independence sampler - where in which proposals are drawn from a fixed proposal, independent of the current state; and,
- the random walk Metropolis - where proposals account for the value previously simulated to generate the following value, i.e.,

  $$\theta' = \theta^{(m)} + \epsilon^{(m)},$$

  where $\epsilon^{(m)}$ is a random perturbation independent of $\theta^{(m)}$.

Furthermore, the Gibbs sampler proposed by Geman and Geman (1984) can be considered to be a special case of the MH algorithm. For the $j$-th component of

the partition of $\Theta$, the Gibbs proposal is set to be proportional to $\pi(\theta_j|\theta_{-j})$ (the conditional density of the $j$-th component, given all other components sampled so far). It can be shown that the corresponding MH update has acceptance probability 1 (Robert and Casella, 2004). See Chapters 7-11 of Robert and Casella (2004) for discussions of other popular MCMC samplers.

In order to prove that the MH algorithm will have $\pi$ as its invariant distribution, it is sufficient to show that its Markov kernel satisfies detailed balance. The MH kernel is given by,

$$K(\theta, A) = \int_A q(\theta, \theta')\alpha(\theta, \theta')\mu(d\theta') + r(x)I(x, A), \tag{2.2.5}$$

where $I(x, A)$ is the identity kernel (Geyer, 2005). We accept proposals $\theta'$ with probability density,

$$p(\theta, \theta') = q(\theta, \theta')\alpha(\theta, \theta').$$

For any measurable set $A$ the first term of (2.2.5), $\int_A q(\theta, \theta')\alpha(\theta, \theta')\mu(d\theta')$, corresponds to the accepted moves in the chain. If this integral were to be evaluated over the whole state space, we would obtain the total probability that the proposed move would be accepted. In that case, the probability that the proposed move is rejected is given by,

$$r(x) = \int q(\theta, \theta')(1 - \alpha(\theta, \theta'))\mu(d\theta').$$

We remain at the original position $\theta$ if the proposal is rejected. It can be verified that detailed balance (2.2.1) holds if and only if,

$$\pi(d\theta)q(\theta, d\theta')\alpha(\theta, \theta') = \pi(d\theta)q(\theta', d\theta)\alpha(\theta', \theta).$$

A simple manipulation of the acceptance probability leads us to the desired result. Theorem 7.2 of Robert and Casella (2004) provides a more formal discussion.

Under further properties on the state space $\Theta$ and the proposal density, it is possible to ensure the Markov chain that results from MH is aperiodic and Harris recurrent (Meyn and Tweedie, 1993c; Robert and Casella, 2004; Geyer, 2005). This then allows for the MH samples to satisfy the Strong Law of Large Numbers. Efficient implementation of the MH algorithm, however, also requires tuning within the chosen

family of proposal densities. Optimal scaling results - such as those for the random walk Metropolis (Roberts et al., 1997) - can be obtained under structured assumptions on the proposal and the target distribution.

In practice, we only simulate from our Markov chain for a finite number of iterations. A good proposal distribution is therefore required to ensure good properties of the chain. Samplers should ideally take into account the local structure of the target distribution. In Section 2.3.1, we discuss how to construct efficient proposals for the MH algorithm that leverage gradient information via continuous-time Markov processes.

### 2.2.3 Scaling up MCMC to tall data

Recall the Bayesian framework introduced in Chapter 1. Let $\mathbf{x} = \{x_1, ..., x_N\} \subset \mathbb{R}^d$ denote a tall dataset with $N \gg 1$. Let $\theta \in \Theta$ denote the model parameters with prior $p(\theta)$. If we assume that we have observed independent data points, the model likelihood would be given by $\prod_{i=1}^{N} p(x_i|\theta)$ and the average loglikelihood can be written as

$$\ell(\theta) = \frac{1}{N} \sum_{i=1}^{N} \log p(x_i|\theta).$$

Using this notation, we can write the posterior density as

$$\pi(\theta) = p(\theta|\mathbf{x}) \propto p(\theta) \exp(N\ell(\theta)). \tag{2.2.6}$$

The Metropolis-Hastings algorithm outlined in Section 2.2.2 can be used to sample from the posterior. In this case, Step 2 of Algorithm 1 is equivalent to setting

$$\log \alpha(\theta, \theta') = \log \left[ \frac{p(\theta')q(\theta', \theta)}{p(\theta)q(\theta, \theta')} \right] + N[\ell(\theta') - \ell(\theta)]. \tag{2.2.7}$$

We see that the MH update requires a full scan of the dataset to calculate the acceptance ratio. When handling tall data, evaluating the loglikelihood ratio in (2.2.7) can be very expensive and this can make it difficult to implement standard MH directly (Bardenet et al., 2017).

There has been a strong push towards developing scalable Monte Carlo methods in the computational statistics literature. These efforts can be broadly classified into five themes.

**Batch (embarrassingly parallel) methods**

A natural mechanism to solve tall data problems is to split the data into tractable batches. Assuming an appropriate independence structure, the posterior can be expressed as,

$$p(\theta|\mathbf{x}) \propto \prod_{s=1}^{S} p(\mathbf{x}_s|\theta)p(\theta)^{\frac{1}{S}},$$

where the prior distribution $p(\theta) = \prod_s p(\theta)^{\frac{1}{S}}$ is decomposed into $S$ components. The data points in each batch, $\mathbf{x}_s$, are assumed to be conditionally independent of all other batches. The idea is to divide and conquer - we can run MH on each batch posterior in parallel and then aggregate the results. While it can be relatively simple to sample from each batch posterior, the key problem to address is how exactly the samples should be reconciled at the end. Huang and Gelman (2005), Neiswanger et al. (2014), Liu (2016), Scott et al. (2016), Nemeth and Sherlock (2018) and Vyner et al. (2022) all propose various means of doing this.

**Exact (pseudo-marginal) subsampling approaches**

Pseudo-marginal MH is a variant of traditional MH which relies on constructing an unbiased, almost surely non-negative estimator of the unnormalised target posterior for any $\theta \in \Theta$ (Andrieu and Roberts, 2009; Bardenet et al., 2017). This estimator can then be substituted into Step 2 of Algorithm 1 and the subsequent Markov chain will attempt to sample exactly from $\pi$. By drawing a random subsample of the data (with or without replacement), it is possible to construct a scalable, unbiased estimator of the unnormalised target, at only a fraction of the computational cost. Although the idea of an exact approximation is appealing, it can be difficult to find suitable general estimators that satisfy our requirements (Jacob and Thiery, 2015). For further discussion, we refer to the comprehensive survey conducted in Quiroz et al. (2018) of pseudo-marginal subsampling approaches for large datasets.

**Approximate subsampling approaches**

This class of methods again considers subsampling approaches where, at each iteration, a random subset of data points are used to approximate the MH acceptance ratio. However, unlike above, the aim here is to sample from an approximation to the target $\pi$. Subsamples are typically selected in an austere manner according to some pre-specified computational budget, with some user-defined tolerance level (Korattikara et al., 2014; Bardenet et al., 2014, 2017).

**Stochastic gradient methods**

In recent years, several scalable MCMC algorithms based on stochastic gradient descent (SGD) have been developed. Here, the idea is to explore the parameter space using SGD updates, replacing the costly gradient calculation over the full dataset with a noisy, unbiased estimate. The class of stochastic gradient MCMC methods (Welling and Teh, 2011; Ma et al., 2015; Nemeth and Fearnhead, 2021) are discussed in further detail in Section 2.3.2. The novel methodology presented in Chapters 3, 4 and 5 of this thesis falls under this family of samplers.

**Continuous time MCMC approaches**

Following the publication of Bardenet et al. (2017) review paper, there has been interesting work that has been conducted on non-reversible MCMC for subsampling applications using continuous time piecewise deterministic Markov processes. See for instance Bouchard-Côté et al. (2018), Fearnhead et al. (2018) and Bierkens et al. (2019).

## 2.3 Stochastic gradient MCMC

Continuous time Markov processes are often used as the basis for Metropolis-Hastings proposals. Two such samplers are the Metropolis-adjusted Langevin algorithm (MALA) (Roberts and Tweedie, 1996; Roberts and Rosenthal, 1998) and Hamiltonian Monte Carlo (HMC) (Neal, 2011). Both methods take into account local gradient information of the target, in order to improve efficiency.

Unfortunately, these approaches do not scale favourably to tall datasets. Gradient-based methods necessitate two passes through the data per-iteration. Specifically, one pass to calculate the gradient of the log-posterior and another whilst computing the MH acceptance ratio. Stochastic gradient MCMC offers a scalable, approximate alternative. They reduce the computational cost by constructing an unbiased, noisy estimate of the gradient, using only a small data subsample. In this section, we first outline the building blocks of gradient-based samplers, before then introducing various stochastic gradient MCMC approaches.

### 2.3.1 Itô processes and MCMC

Most gradient-based samplers - with perhaps the exception of the Barker family of proposals (Livingstone and Zanella, 2022) - are derived from Itô processes, which are Markov processes defined as a solution to a stochastic differential equation (SDE). The overdamped Langevin diffusion and Hamiltonian dynamics are perhaps the most well-known examples of Itô processes in the MCMC literature.

#### Primer on Itô processes

Let $\{\theta(t), t \in \mathbb{R}_+\}$ be a continuous time stochastic process taking values in $\mathbb{R}^d$. The sample path of a stochastic process is the trajectory that corresponds to particular outcome. For convenience, we shorten the random variable $\theta(t)$ to $\theta_t$ for the remainder of this section (Kloeden and Platen, 1995).

If we further define $\mathcal{F}_t$ to be the $\sigma$-field generated by the family of sets $\{\theta_s, 0 \leq s \leq t\}$, $\theta_t$ is Markov process if for all $A \in \mathcal{B}(\mathbb{R}^d)$,

$$\mathbb{P}(\theta_t \in A|\mathcal{F}_s) = \mathbb{P}(\theta_t \in A|\theta_s),$$

for $0 \leq s \leq t$ (Khasminskii, 2011). As in Section 2.2.1, we can establish an analogous framework for defining Harris recurrence and ergodicity in the continuous time case. We refer the reader to Khasminskii (2011) and Meyn and Tweedie (1993a,b) for a formal treatment of these concepts.

We expect the sample paths of a continuous time stochastic process to be well-behaved in most situations. While it may not be reasonable for the paths to be differentiable, we can ensure continuity almost surely. If a continuous time process, $\theta_t$, satisfies Kolmogorov's Continuity Theorem, then sample-path continuity is implied. Moreover, we are interested in the existence of positive constants $\alpha, \beta, C$ and $h$ such that,

$$\mathbb{E}[||\theta_t - \theta_s||^\alpha] \leq C|t-s|^{1+\beta},$$

for $0 \leq s \leq t$ and $|t-s| \leq h$ (Kloeden and Platen, 1995; Øksendal, 2003).

Itô processes are based upon one particular continuous time Markov process known as a Wiener process (also known as Brownian motion). If we consider $\{W_t : t \in \mathbb{R}_+\}$ to be a Wiener process, we know that the following conditions satisfied:

1. $W(0) = 0$ with probability 1,
2. $\mathbb{E}[W(t)] = 0$ for all $t$,
3. $\text{Var}[W(t) - W(s)] = t - s$, for all $0 \leq s \leq t$, and,
4. $W_{t+s} - W_t$ is independent of $W_u$ for $0 < u < t, s \geq 0$ (independent increments),
5. $W_t$ has continuous sample paths (a.s.).

To set up the differential form of an Itô process, we need to be able to integrate with respect to the derivative of a Wiener process. The main difficulty in using traditional integration procedures, such as the Riemann-Stieltjes integral, on the Wiener process is that it is almost surely nowhere differentiable (Kloeden and Platen, 1995).

The Itô integral circumvents this issue by constructing a new integral with respect to the Wiener process (Kloeden and Platen, 1995; Øksendal, 2003). We note that alternatives to the Itô integral - such as the Stratonovich integral - have been developed. However, we focus on the Itô integral at this stage, as it is used more frequently in the literature.

Let $\{\theta_t : t \in \mathbb{R}_+\}$ be a continuous time stochastic process. The Itô integral of $\theta_t$

with respect to $W_t$ is defined as,

$$\int_0^t \theta_s dW_s = \lim_{|P|\to 0} \sum_{k=0}^{m-1} \theta_{t_k}[W_{t_{k+1}} - W_{t_k}],$$

where $P = \{0 = t_0 < t_1 < \ldots < t_m = t\}$ is a partition of $[0, t]$ and $|P| = \sup_k |t_{k+1} - t_k|$, such that the gap between any two consecutive time instants tends to zero (Kloeden and Platen, 1995; Øksendal, 2003). If $W_t$ is a $d$-dimensional Wiener process, then the Itô integral is computed in a coordinate wise manner.

We are now in a position to formally define an Itô process. We denote the functions $b : \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ as the drift and diffusion terms of the SDE respectively. An Itô process is the solution to an SDE of the form,

$$d\theta_t = b(\theta_t)dt + \sigma(\theta_t)dW_t, \tag{2.3.1}$$

where $W_t$ is a $d$-dimensional Wiener process. The equivalent interpretation of (2.3.1) is the stochastic integral equation,

$$\theta_t = \theta_0 + \int_0^t b(\theta_s)ds + \int_0^t \sigma(\theta_s)dW_s.$$

An Itô process is fully specified by its starting point $\theta_0$, the deterministic drift term governed by $b$ and the stochastic diffusion term controlled by $\sigma$. As with deterministic differential equations, a solution to (2.3.1) does not necessarily exist. Conditions are typically placed upon $b$ and $\sigma$ to ensure that a solution exists and that it is unique. Chapter 5 of Øksendal (2003) provides a full discussion of these issues. The following conditions are sufficient for an Itô process to have a unique solution:

- Lipschitz continuity: $||b(\theta) - b(\theta')|| + ||\sigma(\theta) - \sigma(\theta')|| \leq C||\theta - \theta'||$, for some $C \in \mathbb{R}_+$
- Linear growth: $||b(\theta)|| + ||\sigma(\theta)|| \leq C(1 + ||\theta||)$, for $C \in \mathbb{R}_+$.

An Itô diffusion is an Itô process that satisfies the conditions listed above (Øksendal, 2003). It can also be shown that Itô processes satisfy the Markov property and are time homogeneous (Øksendal, 2003; Khasminskii, 2011).

When the drift vector, $b \in \mathbb{R}^d$, and diffusion matrix, $D = \sigma\sigma^T \in \mathbb{R}^{d\times d}$, are moderately regular functions, the transition probabilities of an Itô process will follow the transition density $p = p(s, \theta; t, \vartheta)$, which solves the Fokker-Planck equation (also referred to as the Kolmogorov forward equation),

$$\frac{\partial p}{\partial t} = -\sum_{i=1}^{d} \frac{\partial}{\partial \vartheta_i}\{b^i(\vartheta)p\} + \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2}{\partial \vartheta_i \partial \vartheta_j}\{D^{i,j}(\vartheta)p\}, \tag{2.3.2}$$

for $0 < s \leq t$ and $\theta, \vartheta \in \Theta \subseteq \mathbb{R}^d$. Here, $D^{i,j}$ is the $(i, j)$-th element of the diffusion matrix. If $(s, \theta)$ in $p$ is fixed, the initial condition of (2.3.2) is given by,

$$\lim_{t\downarrow s} p(s, \theta; t, \vartheta) = \delta(\vartheta - \theta),$$

where $\delta(\cdot)$ is the Dirac delta function on $\mathbb{R}^d$. (2.3.2) can be used to describe the evolution of the diffusion over time (Kloeden and Platen, 1995).

The stationary distribution of an Itô process can be found by the letting the right hand side of (2.3.2) be set to zero. This allows us to find solutions of the Fokker-Planck equation that are functionally independent of time. It is sufficient to find a stationary density $p(\vartheta)$ such that,

$$\sum_{i=1}^{d} \frac{\partial}{\partial \vartheta_i}\{b^i(\vartheta)p\} = \frac{1}{2}\sum_{i,j=1}^{d} \frac{\partial^2}{\partial \vartheta_i \partial \vartheta_j}\{d^{i,j}(\vartheta)p\}. \tag{2.3.3}$$

If such a distribution exists, then it will also be invariant and $p(s, \theta; t, \vartheta) \to p(s, \theta; \vartheta) = p(\vartheta)$ as $t \to \infty$.

(2.3.3) is difficult to solve, except for in a handful of cases. The main examples that we will continue to refer back to in this thesis is the overdamped Langevin diffusion and the family of Itô diffusions introduced in Ma et al. (2015).

### Langevin-based samplers

The overdamped Langevin diffusion is an Itô diffusion that is constructed in continuous time so that it converges to a target distribution, $\pi$, under certain regularity conditions (Roberts and Tweedie, 1996). Conventionally, it is assumed that $\pi$ is everywhere non-zero and differentiable, so that the gradient $\nabla \log \pi(\theta)$ is well-defined.

The Langevin diffusion, $\theta(t)$, is defined by the stochastic differential equation,

$$d\theta(t) = \frac{1}{2}\nabla \log \pi(\theta(t))dt + dW_t, \tag{2.3.4}$$

where $\frac{1}{2}\nabla \log \pi(\theta(t))dt$ is a drift term and $W_t$ denotes a $d$-dimensional Wiener process. Under certain regularity conditions, the target distribution of this diffusion is the posterior $\pi$ (Roberts and Tweedie, 1996).

In practice, we need to discretise (2.3.4) in order to simulate from it and this can introduce some approximation error (Kloeden and Platen, 1995). For a small step-size $\epsilon > 0$, the Euler-Maruyama discretisation of (2.3.4) at iteration $m$ is given by

$$\theta^{(m+1)} = \theta^{(m)} + \frac{\epsilon}{2}\nabla \log \pi\big(\theta^{(m)}\big) + \sqrt{\epsilon}\,\eta^{(m)}, \tag{2.3.5}$$

where the noise $\eta^{(m)} \sim \mathcal{N}_d(\mathbf{0}, I_{d\times d})$ is drawn independently at each update .

The dynamics implied by (2.3.5) provide a simple way to approximately sample from the Langevin diffusion. The level of discretisation error in the approximation is controlled by the size of $\epsilon$ and we can achieve any required degree of accuracy if we choose $\epsilon$ small enough.

The unadjusted Langevin algorithm (ULA) (Parisi, 1981; Roberts and Tweedie, 1996) is a simple sampler that simulates from (2.3.5).Roberts and Tweedie (1996) show that the choice of step-size is crucial to maintaining geometric ergodicity in certain cases. If $\epsilon$ is too large, the resulting chain ends up becoming transient and does not admit a stationary distribution.

Given that there is no MH correction, samples obtained from ULA produce a biased approximation of $\pi$. Work conducted in Dalalyan and Karagulyan (2019) and Durmus and Moulines (2019) establishes that with a carefully specified step-size, $\epsilon$, and finite iteration count, $M$, updating the Markov chain (2.3.5) exactly $M$ times can result in an iterate, $\theta_M$, whose distribution is close to that of $\pi$.

Theorem 4 of Dalalyan and Karagulyan (2019) provides non-asymptotic bounds in terms of the Wasserstein distance metric for the rate of convergence of the ULA chain, under the assumption that the target distribution has a smooth and log-concave density $\pi$. The Wasserstein distance of order $\alpha$ (where $\alpha$ is real-valued and greater

than equal to 1), $W_\alpha$, between two measures $\mu$ and $\nu$ on $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$ is given by,

$$W_\alpha(\mu,\nu) = \left[\inf_{\gamma\in\Gamma(\mu,\nu)} \int_{\mathbb{R}^d\times\mathbb{R}^d} \|\theta-\theta'\|^\alpha d\gamma(\theta,\theta')\right]^{\frac{1}{\alpha}},$$

where the infimum is taken over the space $\Gamma(\mu,\nu)$ which consists of all joint distributions $\gamma$ admitting $\mu$ and $\nu$ as marginal distributions.

In their paper, Dalalyan and Karagulyan (2019) argue that the Wasserstein distance measure is generally more suitable for quantifying the convergence error of an approximate sampler over other traditional measures, such as the total-variation norm. For instance, if $\mu$ and $\nu$ are Dirac measures at $x$ and $x'$, the total-variation distance $||\delta_x - \delta_{x'}||_{TV} = 1$ whenever $x \neq x'$. In contrast, the Wasserstein distance between the measures is given by $W_2(\delta_x, \delta_{x'}) = ||x - x'||_2$, which is a smoothly increasing function over $[0,\infty)$. [2]

By introducing a correction step, the Metropolis-adjusted Langevin algorithm (MALA) chain satisfies the detailed balance conditions required to guarantee the existence of a unique stationary distribution, $\pi$. Compared to a simple random walk or independence sampler, MALA uses local gradient information to propose moves in regions of higher density, which are then more likely to be accepted. Roberts and Rosenthal (1998) show us that (in large dimensions) the optimal acceptance rate for MALA is 0.574. In comparison, the optimal acceptance rate of the random walk Metropolis is 0.234 (Roberts et al., 1997).

The per-iteration computational cost of ULA is smaller than that of the Metropolis-adjusted Langevin algorithm. However if the target $\pi$ is the Bayesian posterior, each update of the sampler would be subject to an $O(N)$ full-data gradient calculation. This calculation can be prohibitively expensive if $N$ is too large.

### 2.3.2 Stochastic gradient Langevin dynamics

Welling and Teh (2011) propose a scalable method for Bayesian inference which builds

[2] One issue to take note of with the Wasserstein distance function is that it is not invariant to transformations. For example, if we were to scale a random variable $X$ by a constant $c \in \mathbb{R}$, the Wasserstein distance would increase by a factor of $c^q$.

upon the properties of the ULA sampler. Recall the Bayesian framework outlined in Chapter 1. For convenience, we define $f_i(\theta) = -\log p(x_i|\theta)$ for $i = 1, \ldots, N$, with $f_0(\theta) = -\log p(\theta)$ and $f(\theta) = f_0(\theta) + \sum_{i=1}^{N} f_i(\theta)$. In this setting, the posterior density can be rewritten as, $\pi(\theta) \propto \exp(-f(\theta))$.

The stochastic gradient Langevin dynamics (SGLD) algorithm attempts to improve the per-iteration computational burden of ULA by replacing the full-data gradient with an unbiased estimate (Welling and Teh, 2011). Let the full-data gradient of $f(\theta)$ be given by

$$g^{(m)} = \nabla f\big(\theta^{(m)}\big) = \nabla f_0\big(\theta^{(m)}\big) + \sum_{i=1}^{N} \nabla f_i\big(\theta^{(m)}\big).$$

The unbiased estimate of $g^{(m)}$ proposed by Welling and Teh (2011) takes the form

$$\hat{g}^{(m)} = \nabla f_0\big(\theta^{(m)}\big) + \frac{N}{n} \sum_{i \in \mathcal{S}^m} \nabla f_i\big(\theta^{(m)}\big), \tag{2.3.6}$$

where $\mathcal{S}^m$ is a random subset of $\{1, \ldots, N\}$ and $|\mathcal{S}^m| = n$ ($n \ll N$) is the subsample size. A single update of SGLD is thus given by,

$$\theta^{(m+1)} \leftarrow \theta^{(m)} - \frac{\epsilon^{(m)}}{2} \cdot \hat{g}^{(m)} + \eta^{(m)}, \tag{2.3.7}$$

where $\eta^{(m)} \sim \mathcal{N}_d(0, \epsilon^{(m)} I_{d\times d})$ and $\{\epsilon^{(m)}\}$ corresponds to a schedule of step-sizes which may be fixed (Vollmer et al., 2016) or decreasing (Teh et al., 2016).

Welling and Teh (2011) note that if the step-size $\epsilon^{(m)} \to 0$ as $m \to \infty$, then the Gaussian noise (generated by $\eta^{(m)}$) dominates the noise in the stochastic gradient term. For large $m$, the algorithm approximately samples from the posterior using an increasingly accurate discretisation of the Langevin diffusion. In practice, SGLD does not mix well when the step-size is decreased to zero and so a small fixed step-size $\epsilon$ is typically used instead.

Another issue with the algorithm is that it often explores the state space inefficiently. This is due to the fact that the Langevin diffusion lacks the additional momentum term found in Hamiltonian dynamics possesses (Neal, 2011). Welling and Teh (2011) suggest that this could be remedied by including a preconditioning step to assist with exploration, the thought being that preconditioning would allow the sampler to better take the local features of the posterior in different dimensions into account.

There have been several papers to date that have tried to understand the theoretical underpinnings of the SGLD algorithm. Teh et al. (2016) showed that, under verifiable assumptions, step-size weighted samples are consistent and satisfy a modified Central Limit Theorem. The consistency of the algorithm largely depends on the the use of decreasing step-size schedule, as this asymptotically removes the bias introduced from the Euler discretisation.

There are two competing schools of thought for measuring the accuracy of SGLD convergence. The first considers estimating the expectation of a suitable test function, $\phi(\theta)$, using a set of $M$ SGLD samples, $\frac{1}{M}\sum_{m=1}^{M}\phi\big(\theta^{(m)}\big)$. Teh et al. (2016) show that for a polynomial step-size schedule of the form $\epsilon_m = a(b+m)^{-\alpha}$, setting $\alpha = \frac{1}{3}$ achieves the fastest convergence rate for SGLD. An optimally tuned SGLD chain converges at a rate of $\mathcal{O}(M^{-\frac{1}{3}})$, where $M$ is the total number of iterations run. Note that this convergence rate is slower than the traditional Monte Carlo rate of $\mathcal{O}(M^{-\frac{1}{2}})$ and is due to the decreasing step-size schedule. Vollmer et al. (2016) give similar results for fixed step-size SGLD. The authors derive the asymptotic bias of the constant step-size SGLD explicitly, characterising its dependence on the step-size and the variance of the noisy gradient. Using these results, a modified version of SGLD is proposed, which reduces the impact of the asymptotic bias introduced by the noisy gradient in the original algorithm.

The second class of convergence results compares the distribution that SGLD samples from at some finite iteration $M$ to the target posterior using the Wasserstein distance. See for instance Dalalyan (2017); Durmus and Moulines (2017); Chatterji et al. (2018) and Dalalyan and Karagulyan (2019). Most of these results tend to assume that the log-density of the target posterior is strongly convex.

The results presented in Dalalyan and Karagulyan (2019) also show that the non-asymptotic convergence rates of SGLD depend on the variance of the stochastic gradient estimator. A full analysis of variance reduction in SGLD has been conducted by Chatterji et al. (2018). In that work, the authors compare the theoretical performance of the variance control approaches for SGLD proposed by Dubey et al. (2016), Baker et al. (2019a) and others.

### 2.3.3 Extensions of SGLD

There have been many notable methodological extensions to SGLD in the literature. See Nemeth and Fearnhead (2021) for a full discussion.

Ma et al. (2015) define a general approach to constructing a stochastic gradient MCMC sampler and this lead to a much wider class of stochastic gradient MCMC algorithms. Ma et al. do this by identifying all possible Itô diffusions (including Hamiltonian dynamics, see also Chen et al., 2014) that could be used for posterior sampling, eliminating the need for sampler-specific proofs. Furthermore, the authors show that if the diffusion matrix is positive definite or that the underlying process is ergodic, then the stationary distribution would be unique. Ma et al. (2019) goes further by showing that under the application of one extra assumption, the family of diffusions specified in Ma et al. (2015) is complete.

Patterson and Teh (2013) propose a version of SGLD that is suitable for constrained state spaces, e.g., on the probability simplex (meaning that all components of $\theta$ are strictly positive and sum to 1). Stochastic gradient Riemannian Langevin dynamics (SGRLD) allows for several transformation of the parameters $\theta$ from the probability simplex to $\mathbb{R}^d$. As part of their work to develop a distributed implentation of SGLD, Ahn et al. (2014) adapted SGRLD to a parallelised setting and evaluated its performance on a large-scale Latent Dirichlet Allocation problem. Baker et al. (2018) propose using a discretised version of the Cox-Ingersoll-Ross process instead of the Langevin diffusion to sample on the probability simplex.

There has also been work done to extend to dependent data sources. Li et al. (2016b) propose an SGMCMC algorithm for scalable inference in assortative mixed-membership stochastic block models on a static network. The SGMCMC algorithm proposed by Li et al. builds upon the work of Patterson and Teh (2013). The main idea behind the algorithm is to iteratively update the local parameters and the global parameters of the model on the probability simplex. Given that the global parameters do not change as quickly as the local parameters, a random subset of the global parameters is updated at each iteration.

For a time series setting, Ma et al. (2017) and Aicher et al. (2019) seek to extend

stochastic gradient MCMC to hidden Markov models and linear state space models respectively. These methods scale to large time series data by drawing contiguous subsequences with additional buffers applied on either side. To mitigate the bias, nonoverlapping buffered subsequences are chosen, but the gradient contributions of data points exclusively from the original subsequences are used. Thus far, only analytically tractable state space models have been considered. Our work in Chapter 3 serves as the third paper in this series.

Ahead of Chapter 3, the next section briefly reviews the inference approaches available for parameter estimation in nonlinear, non-Gaussian state space models. For brevity, we restrict our attention to offline Bayesian methods. We note that none of these approaches have been designed with scalability to tall data in mind.

## 2.4 Bayesian parameter estimation for nonlinear state space models

A time series is an ordered sequence of $T$ data points recorded at equally-spaced time intervals. Time series modelling seeks to explain the behaviour of a stochastic process, $X = \{X_t \in \mathbb{R}^{d_x}\}_{t=1}^{T}$, which can be described by parameters $\theta$.

In certain applications, it may be not be possible to directly capture information about the true state process, $X$. We may then need to learn about the state process via a secondary observed process, $Y = \{Y_t \in \mathbb{R}^{d_y}\}_{t=1}^{T}$. A state space model (SSM) allows us to specify the joint distribution of the state process using the observed process. The state process, $X$, is assumed to be a time-homogeneous Markov process such that $Y_t$ depends only on $X_t$ at time $t$ and is conditionally independent of all other observations (Fearnhead and Künsch, 2018).

The posterior distribution over $\theta$ for a general SSM is given by, $\pi(\theta) \propto p(y_{1:T}|\theta)p(\theta)$. We must know this posterior distribution in closed form to be able to conduct inference on $\theta$. However, the marginal likelihood,

$$p(y_{1:T}|\theta) := \prod_{t=1}^{T} p(y_t|y_{1:t-1}, \theta),$$

can only be computed exactly in very specific cases (such as for the linear Gaussian model). For nonlinear, non-Gaussian models, the marginal likelihood is intractable. There are two dominant approaches to tackling this intractable likelihood problem. The first approach uses particle filter methods (Doucet and Johansen, 2009; Kantas et al., 2009) to construct an unbiased approximation of the marginal likelihood. The second approach is known as data augmentation, where the parameters of the SSM are augmented with the state vector and both are inferred within an MCMC algorithm.

### Particle filter approaches

Particle MCMC algorithms target the joint posterior of the parameters and the latent states. The particle marginal Metropolis Hastings (PMMH) algorithm initially proposed by Andrieu et al. (2010) simulates a Markov chain with states, $(\theta, \hat{p}(y_{1:T}|\theta))$. Upon proposing a new parameter vector, $\theta'$, a particle filter is run conditional on the proposal to get $\hat{p}(y_{1:T}|\theta')$. The proposed state is then evaluated with the standard accept-reject step, using the estimated likelihood instead of the true likelihood (Andrieu et al., 2010; Fearnhead and Künsch, 2018).

Andrieu et al. (2010) also propose a particle Gibbs sampler based around using a particle filter to approximate a Gibbs sampler update. Individual particles representing possible states are iteratively updated as one block. Then, a conditional resampling step occurs, where the particle weights are adjusted to reflect the target distribution. A more detailed discussion of the various extensions of PMMH and particle Gibbs can be found in Section 7.2 of Fearnhead and Künsch (2018).

### Data augmentation approaches

Data augmentation treats the unknown latent states as auxiliary variables to be estimated alongside the model parameters $\theta$. This allows for a closed-form complete data likelihood to be specified, which can then be used to construct a joint posterior over the parameters and the states (Tanner and Wong, 1987; van Dyk and Meng, 2001). An MCMC algorithm can then be used to obtain a sample from the complete data likelihood, which we can use to estimate the marginal posterior of $\theta$. Given that

SSMs impose a strong dependence structure on the latent states and parameters, the posterior draws can be highly correlated with poor mixing. Single-update MCMC algorithms tend to perform worse, but the utilisation of block updates can lead to improved results (Borowska and King, 2023).

### Applicability to tall data

Although particle MCMC has been widely celebrated for extending the applicability of MCMC to general SSMs, it comes with some challenges. The performance of PMMH depends heavily on the variability of the acceptance ratio, implying that overestimation of the marginal likelihood can lead to poor performance. It is recommended that the number of particles, $N$, should scale linearly with the length of the time series, $T$, so as to maintain reasonable performance (Kantas et al., 2009; Yıldırım et al., 2018). Moreover, each particle MCMC iteration requires a separate particle filter update and practioners typically may find that they need to run several thousands of iterations to obtain reasonable results. In general, applying particle MCMC to conduct inference on longer time series data can be computationally prohibitive. Mingas et al. (2017), Yıldırım et al. (2018), and Hirt and Dellaportas (2019) all propose novel approaches to tackle this issue.

On the other hand, data augmentation MCMC approaches suffer from the many of the same scaling issues as standard MCMC. If a Metropolis-Hastings step is required, the per-iteration cost would scale linearly with the length of the time series, $T$. Practitioners may fare slightly better if they opt for a Gibbs approach with well-specified block updates, provided that conditional posteriors could be obtained.

With the incorporation of random subsampling, we are able to tackle these scalability issues head on. Chapter 3 does exactly this by taking drawing contiguous, buffered subsequences of the observed time series at each iteration. Furthermore, we outline (both empirically and theoretically) the tradeoffs that must be made between the subsequence size, the additional buffer size and the number of particles used when implementing the proposed approach.

# Chapter 3

# Stochastic Gradient MCMC for Nonlinear State Space Models

## 3.1 Introduction

Nonlinear *state space models* (SSMs) are widely used in many scientific domains for modeling time series. For example, nonlinear SSMs can be applied in engineering (e.g., target tracking, Gordon et al. 1993), in epidemiology (e.g., compartmental disease models, Dukic et al. 2012), and to financial time series (e.g., stochastic volatility models, Shephard 2005). To capture complex dynamical structure, nonlinear SSMs augment the observed time series with a latent state sequence, inducing a Markov chain dependence structure. Parameter inference for nonlinear SSMs requires us to handle this latent state sequence. This is typically achieved using *particle filtering* methods.

Particle filtering algorithms are a set of flexible Monte Carlo simulation-based methods, which use a set of samples, also known as *particles*, to approximate the posterior distribution over the latent states. Unfortunately, inference in nonlinear SSMs does not scale well to long sequences: (i) the cost of each update requires full passes through the data that scales linearly with the length of the sequence, and (ii) the number of particles (and hence the computation per data point) required to control the bias of the particle filter scales linearly with the length of the sequence Kantas

et al. (2015).

Stochastic gradient Markov chain Monte Carlo (SG-MCMC) is a popular method for scaling Bayesian inference to large data sets, replacing full data gradients with stochastic gradient estimates based on subsets of data (Welling and Teh, 2011; Ma et al., 2015). In the context of SSMs, naive stochastic gradients are biased because subsampling breaks temporal dependencies in the data (Ma et al., 2017; Aicher et al., 2019). To correct for this, Ma et al. (2017) and Aicher et al. (2019) have developed *buffered* stochastic gradient estimators that control the bias. The latent state sequence is marginalized in a buffer around each subsequence, which reduces the effect that breaking dependencies has on the estimate of the gradient. However, the work so far has been limited to SSMs where analytic marginalization is possible (e.g., finite-state HMMs and linear dynamical systems).

In this work, we propose *particle buffered* gradient estimators that generalize the buffered gradient estimators to nonlinear SSMs. Although straightforward in concept, a number of unique challenges arise in this setting. First, we show how buffering in nonlinear SSMs can be approximated with a modified particle filter. Second, we provide an error analysis of our proposed estimators by decomposing the error into subsequence error, buffering error, and particle filter error and analyze how this error propagates to estimating posterior means with SGMCMC. Third, we extend the buffering error bounds of Aicher et al. (2019) to nonlinear SSMs with log-concave likelihoods and show that buffer error decays geometrically in buffer size, ensuring that a small buffer size can be used in practice.

The theory we present highlights the importance of controlling bias in the estimate of the gradient – as whilst the impact of a high variance estimator on the accuracy of the SG-MCMC algorithm can be controlled by increasing the number of steps and reducing the step size, it is not possible to change the implementation of the SG-MCMC algorithm to reduce the impact of the bias. We then show theoretically that introducing buffering enables us to control the bias of the estimates of the gradient – with the bias decaying geometrically in the size of the buffer. We investigate the accuracy of our new approach on a range of models with both synthetic and real data

– and show that for fixed computational cost we have obtained substantial gains in accuracy over alternatives. This is due to the reduced bias relative to unbuffered versions of SG-MCMC and through the fact that using stochastic gradient methods allows for more iterations of the MCMC algorithm when compared to approaches that estimate gradients using all observations.

Python code for our Algorithm and for replicating our numerical studies is available at `https://github.com/aicherc/sgmcmc_ssm_code`.

## 3.2 Background

### 3.2.1 Nonlinear State Space Models for Time Series

State space models are a class of discrete-time bivariate stochastic processes consisting of a latent state process $X = \{X_t \in \mathbb{R}^{d_x}\}_{t=1}^T$ and a second observed process, $Y = \{Y_t \in \mathbb{R}^{d_y}\}_{t=1}^T$. The evolution of the state variables is typically assumed to be a time-homogeneous Markov process, such that the latent state at time $t$, $X_t$, is determined only by the latent state at time $t-1$, $X_{t-1}$. The observed states are conditionally independent given the latent states. Given the prior $X_0 \sim \nu(x_0|\theta)$ and parameters $\theta \in \Theta$, the generative model for $X, Y$ is thus

$$\begin{aligned} X_t|(X_{t-1} = x_{t-1}, \theta) &\sim p(x_t \,|\, x_{t-1}, \theta), \\ Y_t|(X_t = x_t, \theta) &\sim p(y_t \,|\, x_t, \theta), \end{aligned} \tag{3.2.1}$$

where we call $p(x_t \,|\, x_{t-1}, \theta)$ the *transition density* and $p(y_t \,|\, x_t, \theta)$ the *emission density*.

For an arbitrary sequence $\{z_i\}$, we use $z_{i:j}$ to denote the sequence $(z_i, z_{i+1}, \ldots, z_j)$. To infer the model parameters $\theta$, a quantity of interest is the *score function*, the gradient of the marginal loglikelihood, $\nabla_\theta \log p(y_{1:T}|\theta)$. Using the score function, the loglikelihood can be maximized iteratively via a (batch) *gradient ascent* algorithm (Robbins and Monro, 1951), given the observations, $y_{1:T}$.

If the latent state posterior $p(x_{1:T}|y_{1:T}, \theta)$ can be expressed analytically, we can

calculate the score using *Fisher's identity* (Cappé et al., 2005),

$$\begin{aligned}\nabla_\theta \log p(y_{1:T} \mid \theta) &= \mathbb{E}_{X|Y,\theta}[\nabla_\theta \log p(X_{1:T}, y_{1:T} \mid \theta)] \\ &= \sum_{t=1}^{T} \mathbb{E}_{X|Y,\theta}[\nabla_\theta \log p(X_t, y_t \mid x_{t-1}, \theta)]. \end{aligned} \tag{3.2.2}$$

If the latent state posterior, $p(x_{1:T}|y_{1:T}, \theta)$, is not available in closed-form, we can approximate the expectations of the latent state posterior. One popular approach is via *particle filtering* methods.

### Particle Filtering and Smoothing

*Particle filtering* algorithms (see e.g., Doucet and Johansen, 2009; Fearnhead and Künsch, 2018) can be used to create an empirical approximation of the expectation of a function $H(X_{1:T})$ with respect to the posterior density, $p(x_{1:T}|y_{1:T}, \theta)$. This is done by generating a collection of $N$ random samples or *particles*, $\{x_t^{(i)}\}_{i=1}^N$ and calculating their associated importance weights, $\{w_t^{(i)}\}_{i=1}^N$, recursively over time. We update the particles and weights with *sequential importance resampling* (Doucet and Johansen, 2009) in the following manner.

(i) *Resample* auxiliary ancestor indices $\{a_1, \ldots, a_N\}$ with probabilities proportional to the importance weights, i.e., $a_i \sim \text{Categorical}(w_{t-1}^{(i)})$.

(ii) *Propagate* particles $x_t^{(i)} \sim q(\cdot|x_{t-1}^{(a_i)}, y_t, \theta)$, using a proposal distribution $q(\cdot|\cdot)$.

(iii) *Update* and normalize the weight of each particle,

$$w_t^{(i)} \propto \frac{p(y_t|x_t^{(i)}, \theta)p(x_t^{(i)}|x_{t-1}^{(a_i)}, \theta)}{q(x_t^{(i)}|x_{t-1}^{(a_i)}, y_t, \theta)} \ , \ \sum_i w_t^{(i)} = 1 \ . \tag{3.2.3}$$

The auxiliary variables, $\{a_i\}_{i=1}^N$, represent the indices of the *ancestors* of the particles, $\{x_t^{(i)}\}_{i=1}^N$, sampled at time $t$. The introduction of ancestor indices allows us to keep track of the lineage of particles over time (Andrieu et al., 2010). The *multinomial resampling* scheme given in (i) describes the procedure by which *offspring* particles are produced.

Resampling at each iteration is used to mitigate against the problem of *weight degeneracy*. This phenomenon occurs when the variance of the importance weights grows, causing more and more particles to have negligible weight. Aside from the multinomial resampling scheme described above, there are various other resampling schemes outlined in the particle filtering literature, such as stratified sampling (Kitagawa, 1996) and residual sampling (Liu and Chen, 1998).

If the proposal density $q(x_t|x_{t-1}, y_t, \theta)$ is the transition density $p(x_t|x_{t-1}, \theta)$ we obtain the *bootstrap particle filter* (Gordon et al., 1993). By using the transition density for proposals, the importance weight recursion in (3.2.3) simplifies to $w_t^{(i)} \propto p(y_t|x_t^{(i)}, \theta)$.

When our target function decomposes into a pairwise sum $H(x_{1:T}) = \sum_{t=1}^{T} h_t(x_t, x_{t-1})$ – such as for Fisher's identity $h_t(x_t, x_{t-1}) = \nabla_\theta \log p(y_t, x_t \mid x_{t-1}, \theta)$ – then we only need to keep track of the partial sum $H_t = \sum_{s=1}^{t} h_s(x_s, x_{s-1})$ in the filter Doucet and Johansen (2009): see Algorithm 2.

**Algorithm 2:** Particle Filter

1: **Input:** number of particles, $N$, pairwise statistics, $h_{1:T}$, observations $y_{1:T}$, proposal density $q$,
2: Draw $x_0^{(i)} \sim \nu(x_0|\theta)$, set $w_0^{(i)} = \frac{1}{N}$, and $H_0^{(i)} = 0\ \forall i$.
3: **for** $t = 1, \ldots, T$ **do**
4: Resample ancestor indices $\{a_1, \ldots, a_N\}$.
5: Propagate particles $x_t^{(i)} \sim q(\cdot|x_{t-1}^{(a_i)}, y_t, \theta)$.
6: Update each $w_t^{(i)}$ according to (3.2.3).
7: Update statistics $H_t^{(i)} = H_{t-1}^{(a_i)} + h_t(x_t^{(i)}, x_{t-1}^{(a_i)})$.
8: **end for**
9: Return $H = \sum_{i=1}^{N} w_T^{(i)} H_T^{(i)}$.

A key challenge for particle filters is handling large $T$. Not only do long sequences require $\mathcal{O}(T)$ computation, but particle filters require a large number of particles, $N$, to avoid *particle degeneracy*: the use of resampling in the particle filter causes path-dependence over time, depleting the number of distinct particles available overall. For Algorithm 2, the variance in $H$ scales as $\mathcal{O}(T^2/N)$ (Poyiadjis et al., 2011).

Therefore to maintain a constant variance, the number of particles would need to increase quadratically with $T$, which is computationally infeasible for long sequences. Poyiadjis et al. (2011); Nemeth et al. (2016) and Olsson and Westerborn (2017) propose alternatives to Step 7 of Algorithm 2 that trade additional computation or bias to decrease the variance in $H$ to $\mathcal{O}(T/N)$. Fixed-lag particle smoothers provide another approach to avoid particle degeneracy, where sample paths are not updated after a fixed lag (Kitagawa and Sato, 2001; Dahlin et al., 2015). All of these methods perform a full pass over the data $y_{1:T}$, which requires $\mathcal{O}(T)$ computation.

### 3.2.2 Stochastic Gradient MCMC

One popular method to conduct scalable Bayesian inference for large data sets is *stochastic gradient* Markov chain Monte Carlo (SGMCMC). Given a prior $p(\theta)$, to draw a sample $\theta$ from the posterior $p(\theta|y) \propto p(y|\theta)p(\theta)$, gradient-based MCMC methods simulate a stochastic differential equation (SDE) based on the gradient of the loglikelihood $g_\theta = \nabla_\theta \log p(y|\theta)$, such that the posterior is the stationary distribution of the SDE. SGMCMC methods replace the full-data gradients with stochastic gradients, $\widehat{g}_\theta$, using subsamples of the data to avoid costly computation.

The most common method of the SGMCMC family is the *stochastic gradient Langevin dynamics* (SGLD) algorithm (Welling and Teh, 2011; Nemeth and Fearnhead, 2021):

$$\theta^{(k+1)} \leftarrow \theta^{(k)} + \epsilon^{(k)} \cdot (\widehat{g}_\theta + \nabla \log p(\theta)) + \mathcal{N}(0, 2\epsilon^{(k)}), \tag{3.2.4}$$

where $\epsilon^{(k)}$ is the stepsize and $\theta_1$ is an initialization of the chain. When $\widehat{g}_\theta$ is unbiased and with an appropriate decreasing stepsize, the distribution of $\theta^{(k)}$ asymptotically converges to the posterior distribution (Teh et al., 2016). Dalalyan and Karagulyan (2019) provide non-asymptotic bounds on the Wasserstein distance between the posterior and the output of SGLD after $K$ steps for fixed $\epsilon^{(k)} = \epsilon$ and possibly biased $\widehat{g}_\theta$.

Many extensions of SGLD exist in the literature, including using control variates to reduce the variance of $\widehat{g}_\theta$ (Baker et al., 2019a; Nagapetyan et al., 2017; Chatterji

et al., 2018) and augmented dynamics to improve mixing (Ma et al., 2015) such as stochastic gradient Hamiltonian Monte Carlo (Chen et al., 2014), stochastic gradient Nosé-Hoover thermostat (Ding et al., 2014), and stochastic gradient Riemannian Langevin dynamics (Girolami and Calderhead, 2011; Patterson and Teh, 2013).

### Stochastic Gradients for SSMs

An additional challenge when applying SGMCMC to SSMs is handling the temporal dependence between observations. Based on a subset $\mathcal{S}$ of size $S$, an unbiased stochastic gradient estimate of (3.2.2) is

$$\sum_{t\in\mathcal{S}} \Pr(t \in \mathcal{S})^{-1} \cdot \mathbb{E}_{X|y_{1:T},\theta}[\nabla_\theta \log p(X_t, y_t \mid X_{t-1}, \theta)]. \tag{3.2.5}$$

Although (3.2.5) is a sum over $S$ terms, it requires taking expectations with respect to $p(x|y_{1:T}, \theta)$, which requires processing the full sequence $y_{1:T}$. One approach to reduce computation is to randomly sample $\mathcal{S}$ as a contiguous subsequence $\mathcal{S} = \{s+1, \ldots, s+S\}$ and approximate (3.2.5) using only $y_{\mathcal{S}}$

$$\sum_{t\in\mathcal{S}} \Pr(t \in \mathcal{S})^{-1} \cdot \mathbb{E}_{X|y_{\mathcal{S}},\theta}[\nabla_\theta \log p(X_t, y_t \mid X_{t-1}, \theta)]. \tag{3.2.6}$$

However, (3.2.6) is *biased* because the expectation over the latent states $x_{\mathcal{S}}$ is conditioned only on $y_{\mathcal{S}}$ rather than $y_{1:T}$.

To control the bias in stochastic gradients while also avoiding accessing the full sequence, previous work on SGMCMC for SSMs proposed *buffered* stochastic gradients (Ma et al., 2017; Aicher et al., 2019).

$$\widehat{g}_\theta(S, B) = \sum_{t\in\mathcal{S}} \frac{\mathbb{E}_{X|y_{\mathcal{S}^*},\theta}[\nabla_\theta \log p(X_t, y_t \mid X_{t-1}, \theta)]}{\Pr(t \in \mathcal{S})}, \tag{3.2.7}$$

where $\mathcal{S}^* = \{s + 1 - B, \ldots, s + S + B\}$ is the *buffered* subsequence such that $\mathcal{S} \subseteq \mathcal{S}^* \subseteq \{1, \ldots, T\}$ (see Figure 3.2.1). When the "buffer" extends outside of the original subsequence (e.g., $s + 1 - B < 1$ or $s + S + B > T$), then we can extend the model to $\{1 - B, \ldots, T + B\}$ and assume the observations $y_t$ outside of $\{1, \ldots, T\}$ are missing. In practice, we will truncate $\mathcal{S}^*$ by intersecting it with $\{1, \ldots, T\}$.

The unbiased gradient estimate, which conditions on all data (3.2.5), is $\widehat{g}(S,T)$ and the estimator with no buffering (3.2.6) is $\widehat{g}(S,0)$. As $B$ increases from 0 to $T$, the estimator $\widehat{g}_\theta(S,B)$ trades computation for reduced bias. In particular, when the

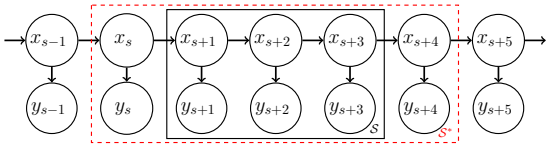



Figure 3.2.1: Graphical model of $\mathcal{S}^*$ with $S = 3$ and $B = 1$.

model and gradient both satisfy a Lipschitz property, the error decays geometrically in buffer size $B$, see Theorem 4.1 of Aicher et al. (2019). Specifically, for all $\mathcal{S}$

$$\|\widehat{g}_\theta(S,B) - \widehat{g}_\theta(S,T)\|_2 = \mathcal{O}(L_\theta^B \cdot T/S), \tag{3.2.8}$$

where $L_\theta$ is a bound for the Lipschitz constants of the *forward and backward smoothing kernels*[1]

$$\begin{aligned}\vec{\Psi}_t(x_{t+1}, x_t) &= p(x_{t+1} \,|\, x_t, y_{1:T}, \theta),\\ \overleftarrow{\Psi}_t(x_{t-1}, x_t) &= p(x_{t-1} \,|\, x_t, y_{1:T}, \theta).\end{aligned} \tag{3.2.9}$$

The bound provided in (3.2.8) ensures that only a modest buffer size $B$ is required (e.g., $\mathcal{O}(\log \delta^{-1})$ for an accuracy of $\delta$). Unfortunately, neither the buffered stochastic gradient $\widehat{g}_\theta(S,B)$ nor the smoothing kernels $\{\vec{\Psi}_t, \overleftarrow{\Psi}_t\}$ have a closed form for nonlinear SSMs.

## 3.3 Method

In this section, we propose a particle buffered stochastic gradient for nonlinear SSMs, by applying the particle approximations of Section 3.2.1 to (3.2.7).

[1]We follow Aicher et al. (2019) and consider Lipschitz constants for a kernel $\Psi$ measured in terms of the $p$-Wasserstein distance between distributions of $x, x'$ and $\Psi(x), \Psi(x')$.

### 3.3.1 Buffered Stochastic Gradient Estimates for Nonlinear SSMs

Let $g_\theta^{\mathrm{PF}}(S, B, N)$ denote the particle approximation of $\widehat{g}_\theta(S, B)$ with $N$ particles. We approximate the expectation over $p(x|y_{\mathcal{S}^*}, \theta)$ in (3.2.7) using Algorithm 2 run over $\mathcal{S}^*$. In the following we will use $\nu_0$ as the prior distribution for $X_{s+1-B}$, which is a natural choice if the state process is stationary and $\nu_0$ is its stationary distribution; for other cases better choices for the prior distribution of $X_{s+1-B}$ may be possible.

The complete data loglikelihood, $\log p(y_{\mathcal{S}}, x_{\mathcal{S}}, \theta)$, in (3.2.7) decomposes into a sum of pairwise statistics

$$H = \sum_{t \in \mathcal{S}^*} h_t(x_t, x_{t-1}) \ , \tag{3.3.1}$$

where

$$h_t(x_t, x_{t-1}) = \begin{cases} \dfrac{\nabla_\theta \log p(x_t, y_t \mid x_{t-1}, \theta)}{\Pr(t \in \mathcal{S})} & \text{if } t \in \mathcal{S}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3.2}$$

We highlight that the statistic is zero for $t$ in the left and right buffers $\mathcal{S}^* \backslash \mathcal{S}$. Although $H_t$ is not updated by $h_t$ for $t$ in $\mathcal{S}^* \backslash \mathcal{S}$, running the particle filter over the buffers is *crucial* to reduce the bias of $g_\theta^{\mathrm{PF}}(S, B, N)$.

Note that $g_\theta^{\mathrm{PF}}(S, B, N)$ allows us to approximate the non-analytic expectation in (3.2.7) with a modest number of particles $N$, by avoiding the particle degeneracy and full sequence runtime bottlenecks, as the particle filter is only run over $\mathcal{S}^*$, which has length $S + 2B \ll T$.

### 3.3.2 SGMCMC Algorithm

Using $g_\theta^{\mathrm{PF}}(S, B, N)$ as our stochastic gradient estimate in SGLD, (3.2.4), gives us Algorithm 3.

Algorithm 3 can be extended by (i) averaging over multiple sequences or varying the subsequence sampling method (Schmidt et al., 2015; Ou et al., 2018), (ii) using different particle filters such as those listed in Section 3.2.1, and (iii) using more advanced SGMCMC schemes such as those listed in Section 3.2.2.

**Algorithm 3:** Buffered PF-SGLD

Input: data $y_{1:T}$, initial $\theta^{(0)}$, stepsize $\epsilon$, subsequence size $S$, buffer size $B$, particle size $N$
**for** $k = 1, 2, \ldots, K$ **do**
  Sample $\mathcal{S} = \{s+1, \ldots, s+S\}$
  Set $\mathcal{S}^* = \{s+1-B, \ldots, s+S+B\}$.
  Calculate $g_\theta^{\mathrm{PF}}$ over $\mathcal{S}^*$ using Alg. 2 on (3.3.2).
  Set $\theta^{(k+1)} \leftarrow \theta^{(k)} + \epsilon \cdot (g_\theta^{\mathrm{PF}} + \nabla \log p(\theta)) + \mathcal{N}(0, 2\epsilon)$
**end for**
Return $\theta^{(K+1)}$

## 3.4 Error Analysis

In this section, we analyze the error of our particle buffered stochastic gradient $g_\theta^{\mathrm{PF}}$ and its effect on approximating posterior means with finite sample averages using Algorithm 3. We first present error bounds for approximating posterior means using SGLD with biased gradients (Theorem 3.4.1). We then present bounds on the gradient bias and MSE of $g_\theta^{\mathrm{PF}}$, extending the error bounds of Aicher et al. (2019) (Theorem 3.4.2). In particular, we provide bounds for the Lipschitz constant $L_\theta$ of the smoothing kernels (3.2.9) without requiring an explicit form for the smoothing kernels (Theorem 3.4.3), allowing (3.2.8) to apply to nonlinear SSMs.

### 3.4.1 Error of Biased SGLD's Finite Sample Averages

We consider the estimation error of the posterior expected value of some test function of the parameters $\phi : \Theta \to \mathbb{R}$ using samples $\theta^{(k)}$ drawn using SGLD with a fixed step size $\epsilon$ and stochastic gradients $g_\theta$.

Let $\bar{\phi}$ be the posterior expected value

$$\bar{\phi} = \mathbb{E}_{p(\theta|y)}[\phi(\theta)] \ , \tag{3.4.1}$$

and let $\hat{\phi}_{K,\epsilon}$ be the $K$-sample estimator for $\bar{\phi}$

$$\hat{\phi}_{K,\epsilon} = \frac{1}{K}\sum_{k=1}^{K}\phi(\theta^{(k)}) \ . \tag{3.4.2}$$

The error of the *finite sample average* $|\hat{\phi}_{K,\epsilon} - \bar{\phi}|$ has been previously studied for SGLD with *unbiased* gradients by Vollmer et al. (2016) and Chen et al. (2015). Following Chen et al. (2015), we make the following assumption on $\phi$.

**Assumption 1.** *Let $\mathcal{L}$ be the generator of the Langevin diffusion*

$$\mathcal{L}[\psi(\theta_t)] = -\nabla \log p(\theta_t)\cdot\nabla\psi(\theta_t) + \frac{\epsilon^2}{2}\operatorname{tr}(\nabla^2\psi(\theta_t)) \ .$$

*Then, we define $\psi$ to solve the* Poisson equation

$$\frac{1}{K}\sum_{k=1}^{K}\mathcal{L}[\psi(\theta^{(k)})] = \hat{\phi}_{K,\epsilon} - \bar{\phi} \ . \tag{3.4.3}$$

*We assume that $\psi(\theta)$ and its derivatives (up to third order) are bounded.*

We now present Theorem 3.4.1, which bounds the error of a finite sample Monte Carlo estimator based on SGLD when the stochastic gradients $\hat{g}_\theta$ are potentially *biased*.

**Theorem 3.4.1** (Error of Finite Sample Average)**.** *If the gradient $g_\theta$ is smooth in $\theta$, the test function $\phi$ satisfies a moment condition (Assumption 1) and the bias and MSE of the gradient estimates $\hat{g}_\theta$ are uniformly bounded, that is,*

$$\|\mathbb{E}\,\hat{g}_\theta - g_\theta\| \le \delta \text{ and } \mathbb{E}\,\|\hat{g}_\theta - g_\theta\|^2 \le \sigma^2 \text{ for all } \theta \ , \tag{3.4.4}$$

*then there exists some constant $C > 0$, such that the bias and MSE of $\hat{\phi}_{K,\epsilon}$ satisfy*

$$|\mathbb{E}\,\hat{\phi}_{K,\epsilon} - \bar{\phi}\,| \le C\cdot\left(\frac{1}{K\epsilon} + \delta\right) + \mathcal{O}(\epsilon) \ , \tag{3.4.5}$$

$$\mathbb{E}\,|\hat{\phi}_{K,\epsilon} - \bar{\phi}\,|^2 \le C\left(\frac{1}{K^2\epsilon^2} + \frac{\sigma^2}{K} + \delta^2 + \frac{\delta}{\epsilon}\right) + \mathcal{O}\left(\frac{1}{K\epsilon} + \delta\epsilon + \epsilon^2\right) \ . \tag{3.4.6}$$

The bias bound, (3.4.5), is a direct application of Theorem 2 in Chen et al. (2015). The MSE bound, (3.4.6), is an extension of Theorem 3 in Chen et al. (2015) when the stochastic gradient estimates $\hat{g}_\theta$ are biased (i.e., $\delta \neq 0$). The additional bias

terms $\delta$ arise from keeping track of additional cross terms in $(\hat{\phi}_{K,\epsilon} - \bar{\phi})^2$. The proof of Theorem 3.4.1 is presented in Section 3.7.1.

From Theorem 3.4.1, we see that the error bounds on $\hat{\phi}_{K,\epsilon}$ are more sensitive to the bias $\delta$ of $\hat{g}$ than the variance $\sigma^2$: the term involving $\sigma^2$ decays with increasing $K$, while terms involving $\delta$ do not decay regardless of stepsize $\epsilon$ or number of samples $K$. A similar conclusion comes from the bound on error of SGLD in Theorem 4 of Dalalyan and Karagulyan (2019): the impact of bias on the error bound is not affected by step size, whereas the impact of the variance can be reduced by taking more steps of smaller size; however, we do not require the posterior distribution be log-concave.

Therefore for the samples from Algorithm 3 to be useful, it is important for the bias of $g_\theta^{\mathrm{PF}}$ to be controlled.

### 3.4.2 Gradient Bias and MSE Bounds

To apply Theorem 3.4.1 to the samples from Algorithm 3, we develop bounds on the bias $\delta$ and MSE $\sigma^2$ of our particle buffered stochastic gradients $g_\theta^{\mathrm{PF}}$.

**Theorem 3.4.2** (Bias and MSE Bounds for $g_\theta^{\mathrm{PF}}$)**.** *For fixed $\theta$, if the model and gradient satisfy a Lipschitz condition and the autocorrelation between $\mathbb{E}_{X|y_{1:T}} \nabla \log p(y_t, X_t | X_{t-1}, \theta)$ for different $t$ is bounded and decays geometrically, then the bias $\delta$ and MSE $\sigma^2$ of $g_\theta^{\mathrm{PF}}$ is bounded by*

$$\delta \leq \gamma \cdot \left[ C_1 \cdot L_\theta^B + \mathcal{O}\left( \frac{S + 2B}{N} \right) \right] \ , \tag{3.4.7}$$

$$\sigma^2 \leq 3\gamma^2 \cdot \left[ C_1^2 \cdot L_\theta^{2B} + C_2 S + \mathcal{O}\left( \frac{(S + 2B)^2}{N} \right) \right] \ , \tag{3.4.8}$$

*where $\gamma = \max_t \Pr(t \in \mathcal{S})^{-1}$ and $C_1, C_2$ are constants with respect to $S, B, N$.*

From Theorem 3.4.2, we see that the bias $\delta$ (3.4.7) can be controlled by selecting large enough $N$ and $B$ when $L_\theta < 1$.

We now sketch the proof of Theorem 3.4.2 and discuss its assumptions. The complete proof can be found in Section 3.7.2.

We decompose the error between $g_\theta^{\mathrm{PF}}$ and the full gradient $g_\theta$ through $\hat{g}_\theta(S,B)$ and $\hat{g}_\theta(S,T)$ into three error sources:

$$\|g_\theta^{\mathrm{PF}}(S,B,N) - g_\theta\| \leq \underbrace{\|g_\theta^{\mathrm{PF}}(S,B,N) - \hat{g}_\theta(S,B)\|}_{\text{particle error (I)}} + \underbrace{\|\hat{g}_\theta(S,B) - \hat{g}_\theta(S,T)\|}_{\text{buffering error (II)}} + \underbrace{\|\hat{g}_\theta(S,T) - g_\theta\|}_{\text{subsequence error (III)}} . \tag{3.4.9}$$

(I) *Particle error*: the Monte Carlo error of the particle filter. From Kantas et al. (2015), the asymptotic bias and MSE of a particle approximation to the sum of $R$ test functions (using Algorithm 2) is $\mathcal{O}(R/N)$ and $\mathcal{O}(R^2/N)$ respectively. Since $g^{\mathrm{PF}}(S,B,N)$ is a particle approximation to the sum of $R = S + 2B$ test functions (i.e., $h_t(x_t, x_{t-1})$), we have

$$\begin{aligned} \|\mathbb{E}\, g_\theta^{\mathrm{PF}}(S,B,N) - \hat{g}_\theta(S,B)\| &= \mathcal{O}\left(\gamma \cdot \frac{S+2B}{N}\right) \\ \mathbb{E}\, \|g_\theta^{\mathrm{PF}}(S,B,N) - \hat{g}_\theta(S,B)\|^2 &= \mathcal{O}\left(\gamma^2 \cdot \frac{(S+2B)^2}{N}\right) , \end{aligned} \tag{3.4.10}$$

where $\gamma$ is a upper bound on the sampling scale factor $\gamma = \max_t \Pr(t \in \mathcal{S})^{-1}$.

Using a more advanced particle filter, such as the "PaRIS" or "Poyiadjis $N^2$" algorithm, Corollary 6 of Olsson and Westerborn (2017) gives a tighter bound for the MSE

$$\mathbb{E}\, \|g_\theta^{\mathrm{PF}}(S,B,N) - \hat{g}_\theta(\,S,B)\|^2 = \mathcal{O}\left(\gamma^2 \cdot \frac{S+2B}{N}\right) .$$

However in our experiments, we found that the improved MSE of these other particle filters was not worth the additional computational overhead for the small subsequences we considered, where $S + 2B \lesssim 100$. See the experiments in Appendix A.

(II) *Buffering error*,: error in approximating the latent state posterior $p(x_{1:T}|y_{1:T})$ with $p(x_{1:T}|y_{\mathcal{S}^*})$. The error stems from conditioning on only a buffered subsequence $y_{\mathcal{S}^*}$ instead of $y_{1:T}$ and the initial distribution approximation $\nu_0$ for $X_{s+1-B}$. If the smoothing kernels $\{\vec{\Psi}_t, \overleftarrow{\Psi}_t\}$ are contractions for all $t$ (i.e., $L_\theta < 1$), then according to (3.2.8), the error in this term is proportional to $\gamma L_\theta^B$. In Section 3.4.3, we show sufficient conditions for $L_\theta < 1$.

(III) *Subsequence error*: the error in approximating Fisher's identity using a randomly chosen subsequence of data points. The error in this term depends on the subsequence size $S$ and how subsequences are sampled. Because we sample random *contiguous* subsequences of size $S$, the MSE scales $\mathcal{O}(\gamma^2 S \frac{1+\rho}{1-\rho})$, where $\rho$ is a bound on the autocorrelation between $\mathbb{E}_{X|y_{1:T}} \nabla \log p(y_t, X_t|X_{t-1}, \theta)$ for different $t$. See Section 3.7.2 for more details.

Combining these error bounds gives us Theorem 3.4.2.

We present examples of the asymptotic bias and MSE bounds given by Theorem 3.4.2 for four different gradient estimators in Table 3.4.1. The four gradient estimators are: (i) naive stochastic subsequence (without buffering) $g^{\mathrm{PF}}(S, 0, N)$ (ii) buffered stochastic subsequence $g^{\mathrm{PF}}(S, B, N)$, (iii) fully buffered stochastic subsequence $g^{\mathrm{PF}}(S, T, N)$, and (iv) full sequence $g^{\mathrm{PF}}(T, T, N)$. For simplicity, we assume the subsequences $\mathcal{S}$ are sampled from a *strict* partition of $1:T$ such that $\gamma = T/S$ and assume $B$ is on the same order as $S$ (i.e., $B$ is $\mathcal{O}(S)$).

Table 3.4.1: Asymptotic bias and compute cost for four different gradient estimators.

| Gradient | $(S, B, N)$ | Bias $\delta$ | Compute |
|---|---|---|---|
| Naive Subsequence | $(S, 0, N)$ | $C_1 \cdot T/S + \mathcal{O}(T/N)$ | $\mathcal{O}(SN)$ |
| Buffered Subsequence | $(S, B, N)$ | $C_1 \cdot L_\theta^B \cdot T/S + \mathcal{O}(T/N)$ | $\mathcal{O}(SN)$ |
| Fully Buffered Subsequence | $(S, T, N)$ | $\mathcal{O}(T/N)$ | $\mathcal{O}(TN)$ |
| Full Sequence | $(T, T, N)$ | $\mathcal{O}(T/N)$ | $\mathcal{O}(TN)$ |

From Table 3.4.1, we see that without buffering, the naive stochastic gradient has a $C_1 \cdot T/S$ term in the bias bound $\delta$. The fully buffered subsequence and full sequence gradients remove the buffering error entirely, but require $\mathcal{O}(TN)$ computation. Instead, our proposed buffered stochastic gradient controls the bias, with the geometrically decaying factor $L_\theta^B$, using only $\mathcal{O}(SN)$ computation.

### 3.4.3 Buffering Error Bound for Nonlinear SSMs

To obtain a bound for the buffering error term (II), we require the Lipschitz constant $L_\theta$ of smoothing kernels $\{\vec{\Psi}_t, \overleftarrow{\Psi}_t\}$ to be less than 1. Typically the smoothing kernels $\vec{\Psi}_t, \overleftarrow{\Psi}_t$ are not available in closed-form for nonlinear SSMs and therefore directly bounding the Lipschitz constant is difficult. However, we now show that when the model's transition and emission densities are *log-concave* in $x_t, x_{t-1}$, we can bound the Lipschitz constant of $\vec{\Psi}_t, \overleftarrow{\Psi}_t$ in terms of the Lipschitz constant of either the *prior kernels* $\vec{\Psi}_t^{(0)}, \overleftarrow{\Psi}_t^{(0)}$, or the *filtered kernels* $\vec{\Psi}_t^{(1)}, \overleftarrow{\Psi}_t^{(1)}$

$$\begin{aligned} \vec{\Psi}_t^{(0)} &:= p(x_t \,|\, x_{t-1}, \theta), & \vec{\Psi}_t^{(1)} &:= p(x_t \,|\, x_{t-1}, y_t, \theta), \\ \overleftarrow{\Psi}_t^{(0)} &:= p(x_t \,|\, x_{t+1}, \theta), & \overleftarrow{\Psi}_t^{(1)} &:= p(x_t \,|\, x_{t+1}, y_t, \theta), \end{aligned} \tag{3.4.11}$$

Unlike the smoothing kernels, the prior kernels are defined by the model and are therefore usually available. If the filtered kernels are available, then they can be used to obtain even tighter bounds.

**Theorem 3.4.3** (Lipschitz Kernel Bound)**.** *Assume the prior for $x_0$ is log-concave in $x$. If the transition density $p(x_t \,|\, x_{t-1}, \theta)$ is log-concave in $(x_t, x_{t-1})$ and the emission density $p(y_t \,|\, x_t)$ is log-concave in $x_t$, then*

$$\|\vec{\Psi}_t\|_{Lip} \;\le\; \|\vec{\Psi}_t^{(1)}\|_{Lip} \;\le\; \|\vec{\Psi}_t^{(0)}\|_{Lip} \tag{3.4.12}$$

$$\|\overleftarrow{\Psi}_t\|_{Lip} \;\le\; \|\overleftarrow{\Psi}_t^{(1)}\|_{Lip} \;\le\; \|\overleftarrow{\Psi}_t^{(0)}\|_{Lip}. \tag{3.4.13}$$

*Therefore*

$$\begin{aligned} L_\theta &= \max_t \{\|\vec{\Psi}_t\|_{Lip}, \|\overleftarrow{\Psi}_t\|_{Lip}\} \\ &\le \max_t \{\|\vec{\Psi}_t^{(1)}\|_{Lip}, \|\overleftarrow{\Psi}_t^{(1)}\|_{Lip}\} \\ &\le \max_t \{\|\vec{\Psi}_t^{(0)}\|_{Lip}, \|\overleftarrow{\Psi}_t^{(0)}\|_{Lip}\} \end{aligned} \tag{3.4.14}$$

This theorem lets us bound $L_\theta$ with the Lipschitz constant of either the prior kernels or filtered kernels. The proof of Theorem 3.4.3 is provided in Section 3.7.3 and uses Caffarelli's log-concave perturbation theorem (Villani, 2008; Colombo et al., 2017). Examples of SSMs for which Theorem 3.4.3 applies include the linear Gaussian

SSM, the stochastic volatility model, or any linear SSM with log-concave transition and emission distributions.

Theorem 3.4.3 lets us calculate analytic bounds on $L_\theta$ for the buffering error of Theorem 3.4.2. We provide explicit bounds for $L_\theta$ for the linear Gaussian SSM and stochastic volatility model in Section 3.5.1 with proofs in the Section 3.7.4.

## 3.5 Experiments

We first empirically test the bias of our particle buffered gradient estimator $g_\theta^{\mathrm{PF}}$ on synthetic data for fixed $\theta$. We then evaluate the performance of our proposed SGLD algorithm (Algorithm 3) on both real and synthetic data.

### 3.5.1 Models

For our experiments, we consider three models: (i) the linear Gaussian SSM (LGSSM), a case where analytic buffering is possible, to assess the impact of the particle filter; (ii) the stochastic volatility model (SVM) (Shephard, 2005), where the emissions are non-Gaussian; and (iii) the generalized autoregressive conditional heteroskedasticity (GARCH) model (Bollerslev, 1986), where the latent transitions are nonlinear.

#### Linear Gaussian SSM

The *linear Gaussian SSM* (LGSSM) is

$$\begin{aligned} X_t \,|\, (X_{t-1} = x_{t-1}, \theta) &\sim \mathcal{N}(x_t \,|\, \phi x_{t-1}\,,\, \sigma^2), \\ Y_t \,|\, (X_t = x_t, \theta) &\sim \mathcal{N}(y_t \,|\, x_t\,,\, \tau^2), \end{aligned} \tag{3.5.1}$$

with $\nu_0(x_0) = \mathcal{N}(x_0 \,|\, 0, \frac{\phi^2}{1-\sigma^2})$ and parameters $\theta = (\phi, \sigma, \tau)$.

The transition and emission distributions are both Gaussian and log-concave in $x$, so Theorem 3.4.3 applies. In Section 3.7.4, we show that the filtered kernels of the LGSSM are bounded with the Lipschitz constant $L_\theta = |\phi| \cdot \sigma^2/(\sigma^2 + \tau^2)$. Thus, the buffering error decays geometrically with increasing buffer size $B$ when $|\phi| < (1 + \frac{\tau^2}{\sigma^2})$.

This linear model serves as a useful baseline since the various terms in (3.4.9) can be calculated analytically.

**Stochastic Volatility Model**

The *stochastic volatility model* (SVM) is

$$\begin{aligned} X_t \,|\, (X_{t-1} = x_{t-1}, \theta) &\sim \mathcal{N}(x_t \,|\, \phi x_{t-1}\,,\, \sigma^2), \\ Y_t \,|\, (X_t = x_t, \theta) &\sim \mathcal{N}(y_t \,|\, 0\,,\, \exp(x_t)\tau^2), \end{aligned} \tag{3.5.2}$$

with $\nu_0(x_0) = \mathcal{N}(x_0 \,|\, 0, \frac{\phi^2}{1-\sigma^2})$ and parameters $\theta = (\phi, \sigma, \tau)$.

For the SVM, the transition and emission distributions are log-concave in $x$, allowing Theorem 3.4.3 to apply. In Section 3.7.4, we show that the prior kernels $\{\vec{\Psi}_t^{(0)}, \overleftarrow{\Psi}_t^{(0)}\}$ of the SVM are bounded with the Lipschitz constant $L_\theta = |\phi|$. Thus, the buffering error decays geometrically with increasing buffer size $B$ when $|\phi| < 1$.

**GARCH Model**

We finally consider a GARCH(1,1) model (with noise)

$$\begin{aligned} X_t \,|\, (X_{t-1} = x_{t-1}, \sigma_t^2, \theta) &\sim \mathcal{N}(x_t \,|\, 0,\, \sigma_t^2), \\ \sigma_t^2(x_{t-1}, \sigma_{t-1}^2, \theta) &= \alpha + \beta x_{t-1}^2 + \gamma \sigma_{t-1}^2, \\ Y_t \,|\, (X_t = x_t, \theta) &\sim \mathcal{N}(y_t \,|\, x_t\,,\, \tau^2), \end{aligned} \tag{3.5.3}$$

with $\nu_0(x_0) = \mathcal{N}(0, \frac{\alpha}{1-\beta-\gamma})$ and parameters $\theta = (\alpha, \beta, \gamma, \tau)$. Unlike the LGSSM and SVM, the noise between $X_t$ and $X_{t-1}$ is multiplicative in $X_{t-1}$ rather than additive. This model's transition distribution is *not* log-concave in $(x_t, x_{t-1})$ and therefore our theory (Theorem 3.4.3) does not hold. However, we see empirically that buffering can help reduce the gradient error for the GARCH in the experiments below and in Appendix A.

### 3.5.2 Stochastic Gradient Bias

We compare the error of stochastic gradient estimates using a buffered subsequence with $S = 16$, while varying $B$ and $N$ on synthetic data from each model. We generated

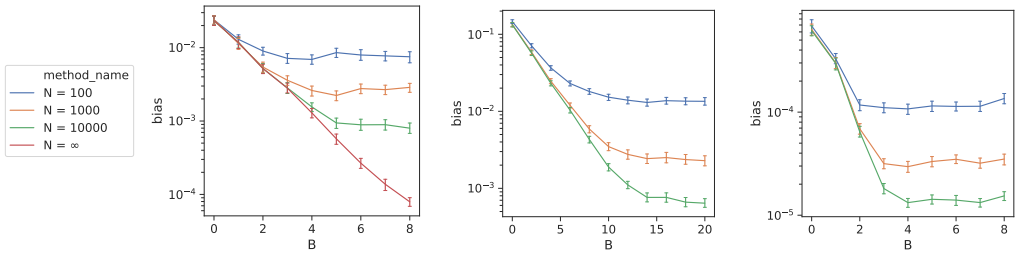



Figure 3.5.1: Stochastic gradient bias varying buffer size $B$ for $S = 16$ for different values of $N$. (left) LGSSM $\phi$, (middle) SVM $\phi$, (right) GARCH $\beta$. Error bars are 95% confidence interval over 1000 replications.

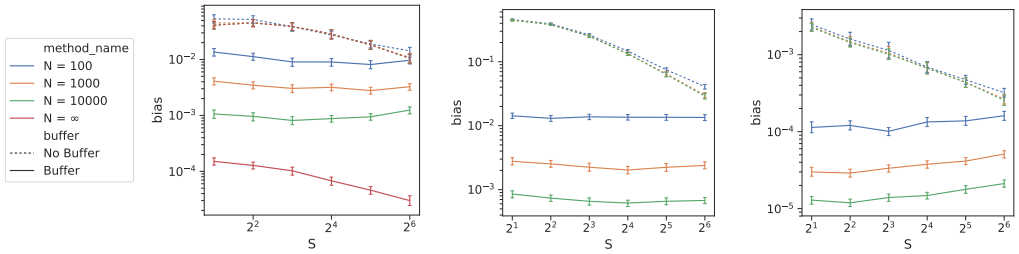



Figure 3.5.2: Stochastic gradient bias varying subsequence size $S$ for No Buffer ($B = 0$) and Buffer ($B > 0$) for different values of $N$. (left) LGSSM $\phi$, (middle) SVM $\phi$, (right) GARCH $\beta$. The buffer size $B = 8$ for LGSSM and GARCH and $B = 16$ for the SVM. Error bars are 95% confidence interval over 1000 replications.

synthetic data of length $T = 256$ using $(\phi = 0.9, \sigma = 0.7, \tau = 1.0)$ for the LGSSM, $(\phi = 0.9, \sigma = 0.5, \tau = 0.5)$ for the SVM, and $(\alpha = 0.1, \beta = 0.8, \gamma = 0.05, \tau = 0.3)$ for the GARCH model.

Figures 3.5.1-3.5.3 display the bias of our particle buffered stochastic gradient estimator, $g_\theta^{\mathrm{PF}}(S, B, N)$, and the ground truth, $g_\theta$, averaged over 1000 replications. The stochastic gradient bias is calculated as $\|g_\theta^{\mathrm{PF}} - g_\theta\|$. We evaluate the gradients at $\theta$ equal to the data generating parameters. We vary the buffer size $B \in [0, 16]$, the subsequence size $S \in [1, T]$ and the number of samples $N \in \{100, 1000, 10000\}$. For the LGSSM, we also consider $N = \infty$, by calculating $g_\theta^{\mathrm{PF}}(S, B, \infty)$ using the Kalman filter (Kalman, 1960), which is tractable in the linear setting. In order to calculate our ground truth $g_\theta$, we use the Kalman filter for the LGSSM, and use $g_\theta \approx g_\theta^{\mathrm{PF}}(T, 0, 10^7)$ as a substitute for the SVM and the GARCH model. We have assumed here that used

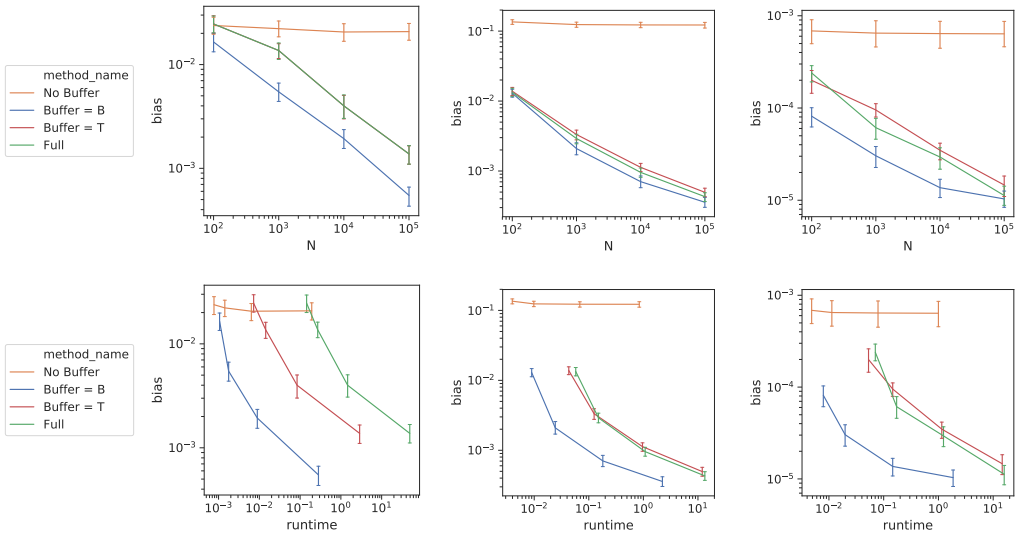



Figure 3.5.3: Stochastic gradient bias varying $N$ for different $S, B$. (left) LGSSM $\phi$, (middle) SVM $\phi$, (right) GARCH $\beta$. (top) $x$-axis is $N$, (bottom) $x$-axis is runtime in seconds. **No Buffer** is $g^{\mathrm{PF}}(16, 0, N)$, **Buffer** $B = B$ is $g^{\mathrm{PF}}(16, B, N)$, **Buffer** $B = T$ is $g^{\mathrm{PF}}(16, T, N)$, and **Full** is $g^{\mathrm{PF}}(T, T, N)$. The moderate buffer size $B = 8$ for LGSSM and GARCH and $B = 16$ for the SVM. Error bars are 95% confidence interval over 1000 replications.

$N = 10^7$ particles is sufficient for an accurate approximation of the ground truth in these 1-dimensional settings.

Figure 3.5.1 shows the bias as we vary the buffer size $B$ for different $N$ and $S = 16$. From Figure 3.5.1, we see the trade-off between the buffering error (II) and the particle error (III) in the bias bound, (3.4.7) of Theorem 3.4.2. For all $N$, when $B$ is small, the buffering error (II) dominates, and therefore the MSE decays exponentially as $B$ increases. However for $N < \infty$, the particle error (III) dominates for larger values of $B$. In fact, the bias slightly increases due to particle degeneracy, as $|\mathcal{S}^*| = S + 2B$ increases with $B$. For $N = \infty$ in the LGSSM case, we see that the bias continues to decreases exponentially with large $B$ as there is no particle filter error when using the Kalman filter.

Figure 3.5.2 shows the bias as we vary the subsequence size $S$ for different $N$ and with and without buffering. We see that buffering helps regardless of subsequence size (as the bias for all buffered methods are lower than the no buffer methods for all

$S \in [2, 64]$). We also see that increasing $S$ can increase the bias for fixed $N$ (when buffering) as the particle error (III) dominates.

Figure 3.5.3 shows the bias as we vary the number of particles $N$ for the four different methods correspond to Table 3.4.1. In the top row, we compare the bias against $N$ and in the bottom row, we compare the bias against the runtime required to calculate $g_\theta^{\mathrm{PF}}$. We see that the method without buffering (orange) is significantly biased regardless of $N$, where as buffering with moderate $B$ (blue), buffering with large $B = T$ (red), and using the full sequence (green) have similar (lower) bias as we increase $N$. However the runtime plots show that buffering with moderate $B$ takes significantly less time.

In summary, Figures 3.5.1-3.5.3 show that buffering cannot be ignored in these three example models: there is high bias for $B = 0$. In general, buffering has diminishing returns when $B$ is excessively large relative to $N$.

In Appendix A, we present plots of the bias varying $B, S, N$ using different particle filters (PaRIS and Poyiadjis $N^2$) instead of the naive PF. We find that they perform similarly to the naive PF for the small subsequence lengths $|\mathcal{S}^*|$ considered, while taking $\approx 10$ times longer to run. We also present plots of the bias as we vary the parameters of the data generating model. We find that as the parameters become more challenging (e.g., $L_\theta \to 1$), we need to increase both $B$ and $N$ to control bias; otherwise, the buffer stochastic subsequence methods are more biased than using full sequence gradient.

### 3.5.3 SGLD Experiments

Having examined the stochastic gradient bias, we now examine using our buffered stochastic gradient estimators in SGLD (Algorithm 3).

#### SGLD Evaluation Method

We measure the sample quality of our MCMC chains $\{\theta^{(k)}\}_{k=1}^K$ using the *kernel Stein discrepancy* (KSD) for equal compute time (Gorham and Mackey, 2017; Liu et al., 2016). We choose to use KSD rather than classic MCMC diagnostics such as effective

sample size (ESS) (Gelman et al., 2013), because KSD penalizes the bias present in our MCMC chains. Whilst it can be hard to interpret the absolute value of KSD for any problem, it is informative for comparing between different algorithms. Given a sample chain (after burnin and thinning) $\{\theta^{(k)}\}_{k=1}^{\tilde{K}}$, let $\hat{p}(\theta|y)$ be the empirical distribution of the samples. Then the KSD between $\hat{p}(\theta|y)$ and the posterior distribution $p(\theta|y)$ is

$$\mathrm{KSD}(\hat{p}, p) = \sum_{d=1}^{\dim(\theta)} \sqrt{\sum_{k,k'=1}^{\tilde{K}} \frac{\mathcal{K}_0^d(\theta^{(k)}, \theta^{(k')})}{\tilde{K}^2}}, \tag{3.5.4}$$

where

$$\mathcal{K}_0^d(\theta, \theta') = \tfrac{1}{p(\theta|y)p(\theta'|y)} \nabla_{\theta_d} \nabla_{\theta'_d} (p(\theta|y)\mathcal{K}(\theta, \theta')p(\theta'|y)) \tag{3.5.5}$$

and $\mathcal{K}(\cdot, \cdot)$ is a valid kernel function. Following Gorham and Mackey (2017), we use the inverse multiquadratic kernel $\mathcal{K}(\theta, \theta') = (1 + \|\theta - \theta'\|_2^2)^{-0.5}$ in our experiments. Since (3.5.5) requires full gradient evaluations of $\log p(\theta|y)$ that are computationally intractable, we replace these terms with corresponding stochastic estimates using the full particle filter estimate, $g_\theta^{\mathrm{PF}}$ Gorham et al. (2020).

### SGLD on Synthetic LGSSM Data

To assess the effect of using particle filters with buffered stochastic gradients, we first focus on SGLD on synthetic LGSSM data, where calculating $\widehat{g}_\theta(S, B)$ is possible. We generate training sequences of length $T = 10^3$ or $10^6$ using the same parametrization as Section 3.5.2.

We consider three pairs of different gradient estimators: **Full** ($S = T$), **Buffered** ($S = 40, B = 10$) and **No Buffer** ($S = 40, B = 0$) each with $N = 1000$ particles using the particle filter and with $N = \infty$ using the Kalman filter. To select the stepsize, we performed a grid search over $\epsilon \in \{1, 0.1, 0.01, 0.001\}$ and selected the method with smallest KSD to the posterior on the training set. We present the KSD results (for the best $\epsilon$) in Table 3.5.1 and trace plots of the metrics in Figure 3.5.4.

From Figure 3.5.4, we see that the methods without buffering ($B = 0$) have higher MSE as they are biased. We also see that the full sequence methods ($S = T$) perform poorly for large $T = 10^6$.

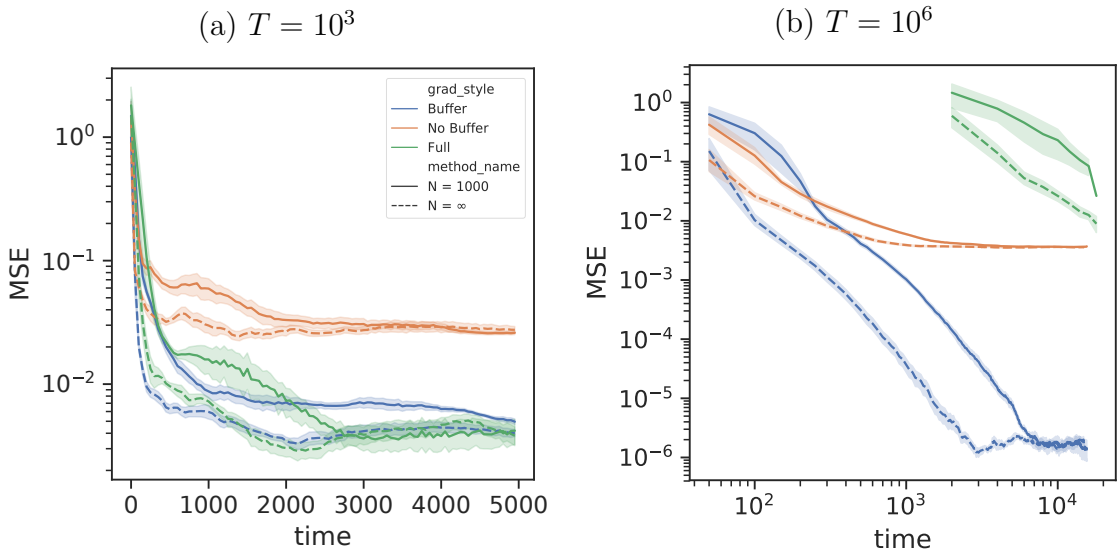



Figure 3.5.4: Comparison of SGLD with different gradient estimates on synthetic LGSSM data: $T = 10^3$ (left), $T = 10^6$ (right). MSE of estimated posterior mean to true $\phi = 0.9$.

The KSD results further support this story. Table 3.5.1 presents the mean and standard deviation on our estimated $\log_{10}$ KSD for $\theta$. Tables of the marginal KSD for individual components of $\theta$ can be found in Appendix A. The methods without buffering have larger KSD, as the inherent bias of $\widehat{g}_\theta(S, B = 0)$ led to an incorrect stationary distribution. The full sequence methods perform poorly for $T = 10^6$ because of a lack of samples that can be computed in a fixed runtime.

In Appendix A, we present similar results on synthetic SVM and GARCH data. Also in Appendix A, we present results on LGSSM in higher dimensions. As is typical in the particle filtering literature, the performance degrades with increasing dimensions for $N$ fixed.

### SGLD on Exchange Rate Log-Returns

We now consider fitting the SVM and the GARCH model to EUR-USD exchange rate data at the minute resolution from November 2017 to October 2018. The data consists of 350,000 observations of demeaned log-returns. As the market is closed

Table 3.5.1: KSD for Synthetic LGSSM. Mean and SD. Results are shown after running each method for a fixed computational time.

| | | | $\log_{10}$KSD | |
|---|---|---|---|---|
| $S$ | $B$ | $N$ | $T = 10^3$ | $T = 10^6$ |
| $T$ | – | 1000 | 0.85 (0.08) | 4.92 (0.40) |
| | | $\infty$ | **0.64 (0.17)** | 4.85 (0.36) |
| 40 | 0 | 1000 | 1.58 (0.03) | 4.68 (0.10) |
| | | $\infty$ | 1.55 (0.03) | 4.68 (0.11) |
| 40 | 10 | 1000 | **0.68 (0.25)** | **3.43 (0.19)** |
| | | $\infty$ | **0.61 (0.21)** | **3.25 (0.29)** |

during non-business hours, we further break the data into 53 weekly segments of roughly 7,000 observations each. In our model, we assume independence between weekly segments and divide the data into a training set of the first 45 weeks and a test set of the last 8 weeks. Full processing details and example plots are in Appendix A. Our method (Algorithm 3) easily scales to the unsegmented series; however the abrupt changes between starts of weeks are not adequately modeled by (3.5.2)

We fit both the SVM and the GARCH model using SGLD with four different gradient methods: (i) **Full**, the full gradient over all segments in the training set; (ii) **Weekly**, a stochastic gradient over a randomly selected segment in the training set; (iii) **No Buffer**, a stochastic gradient over a randomly selected *subsequence* of length $S = 40$; and (iv) **Buffer**, our buffered stochastic gradient for a subsequence of length $S = 40$ with buffer length $B = 10$. To estimate the stochastic gradients, we use Algorithm 2 with $N = 1000$. To select the stepsize parameter, we performed a grid search over $\epsilon \in \{1, 0.1, 0.01, 0.001\}$ and selected the method with smallest KSD. We present the KSD results in Table 3.5.2.

For the SVM, we see that buffering leads to more accurate MCMC samples, Table 3.5.2 (left). In particular, the samples from SGLD without buffering have

Table 3.5.2: KSD for SGLD on exchange rate data. Mean and SD over 5 chains each. Results are shown after running each method for a fixed computational time.

| | $\log_{10}$KSD | |
|---|---|---|
| METHOD | SVM | GARCH |
| **Full** | 4.03 (0.14) | 2.84 (0.30) |
| **Weekly** | 3.87 (0.08) | 2.81 (0.21) |
| **No Buffer** | 4.48 (0.01) | **2.09 (0.09)** |
| **Buffer** | **3.56 (0.08)** | **2.19 (0.05)** |

smaller $\phi, \tau^2$ and a larger $\sigma^2$, indicating that its posterior is (inaccurately) centered around a SVM with larger latent state noise. We also again see that the full sequence and weekly segment methods perform poorly due to the limited number of samples that can be computed in a fixed runtime.

For the GARCH model, Table 3.5.2 (right), we see that the subsequence methods out perform the full sequence methods, but unlike in the SVM, buffering does not help with inference on the GARCH data. This is because the GARCH model that we recover on the exchange rate data (for all gradient methods) is close to white noise $\beta \approx 0$. Therefore the model believes the observations are close to independent, hence no buffer is necessary.

## 3.6 Discussion

In this work, we developed a particle buffered stochastic gradient estimators for nonlinear SSMs. Our key contributions are (i) extending buffered stochastic gradient MCMC with particle filtering for nonlinear SSMs, (ii) analyzing the error of our proposed particle buffered stochastic gradient $g_\theta^{\mathrm{PF}}$ (Theorem 3.4.2) and its affect on our SGLD Algorithm 3 (Theorem 3.4.1), and (iii) generalizing the geometric decay bound for buffering to nonlinear SSMs with log-concave likelihoods (Theorem 3.4.3). We evaluated our proposed gradient estimator with SGLD on both synthetic data and

EUR-USD exchange rate data. We find that buffering is necessary to control bias and that our stochastic gradient methods (Algorithm 3) are able to out perform batch methods on long sequences.

Possible future extensions of this work include relaxing the log-concave restriction of Theorem 3.4.3, extensions to Algorithm 3 as discussed at the end of Section 3.3.2, and applying our particle buffered stochastic gradient estimates to other applications than SGMCMC, such as maximising loglikelihoods or optimization in variational autoencoders for sequential data (Maddison et al., 2017; Naesseth et al., 2018).

## 3.7 Error Analysis Proofs

In this section, we provide additional details and proofs for the error analysis of Section 3.4. In particular, we provide the proof of Theorem 3.4.1 in Section 3.7.1, the proof of Theorem 3.4.2 in Section 3.7.2, the proof of Theorem 3.4.3 in Section 3.7.3 and applications of Theorem 3.4.3 for LGSSM and SVM in Section 3.7.4.

### 3.7.1 Proof of Theorem 3.4.1

We now prove the error bounds for biased SGLD's finite sample average found in Section 3.4.1. The proof is a modification of the proof of Theorem 3 found in Supplement E of Chen et al. (2015).

Recall our assumption on $\phi$ is that $\psi(\theta)$ and its derivatives are bounded by some finite constant $M$ (Assumption 1). This is the implicit moment condition for $\phi$, which is also assumed by Vollmer et al. (2016) and Chen et al. (2015).

The proof of Theorem 3.4.1 then proceeds as in Theorem 3 of Chen et al. (2015), except that we allow for a $\delta > 0$ such that $\mathbb{E}\,\|\hat{g}(\theta) - g(\theta)\| \leq \delta$ for all $\theta$ rather than restrict $\delta = 0$.

For compactness of notation, we will use $g_k$ to denote $g(\theta^{(k)})$, $\hat{g}_k$ to denote $\hat{g}(\theta^{(k)})$, and $\psi_k$ to denote $\psi(\theta^{(k)})$.

*Proof of Theorem 3.4.1.* Following Chen et al. (2015), from the definition of the func-

tional $\psi$ and generator $\mathcal{L}$, we have

$$\hat{\phi}_{K,\epsilon} - \bar{\phi} = \frac{\mathbb{E}\,\psi_K - \psi_1}{K\epsilon} - \frac{\sum_{k=1}^{K}(\mathbb{E}\,\psi_k - \psi_k)}{K\epsilon} + \frac{\sum_{k=1}^{K}(\hat{g}_k - g_k)\cdot\nabla\psi_k}{K} + \mathcal{O}(\epsilon)\ , \quad (3.7.1)$$

and $\mathbb{E}\,(\mathbb{E}\,\psi_k - \psi_k)^2$ is $\mathcal{O}(\epsilon)$. Because $\psi$ is bounded by $M$, we also have $\mathbb{E}\,\psi_K - \mathbb{E}\,\psi_1 < 2M$ and $\mathbb{E}\,(\mathbb{E}\,\psi_K - \psi_1)^2 < 4M^2$.

Let $\xi_k = (\hat{g}_k - g_k)\cdot\nabla\psi_k$. From our assumptions on the bias and MSE of $\hat{g}$ and as $\nabla\psi$ is bounded, we have $|\mathbb{E}\,\xi_k| \le M\delta$ and $\mathbb{E}\,[\xi_k^2] \le M^2\sigma^2$ for all $k$. In addition, we have for all $k \neq k'$

$$|\mathbb{E}\,[\xi_k\xi_{k'}]| \le M^2 \cdot \|\mathbb{E}\,[\hat{g}_k - g_k]\| \cdot \|\mathbb{E}\,[\hat{g}_{k'} - g_{k'}]\| \le M^2\delta^2\ , \quad (3.7.2)$$

where the expectations are over *independent* stochastic subsequences $\mathcal{S}$ chosen at steps $k$ and $k'$.

To prove the bias bound, we take the expectation of (3.7.1), let $C = 2M$ and bound each term

$$|\mathbb{E}\,\hat{\phi}_{K,\epsilon} - \bar{\phi}| \le \frac{2M}{K\epsilon} + M\delta + \mathcal{O}\,(\epsilon) \le C\cdot\left(\frac{1}{K\epsilon} + \delta\right) + \mathcal{O}(\epsilon)\ . \quad (3.7.3)$$

To prove the MSE bound, we take the square and expectation of both sides of (3.7.1),

$$\begin{aligned}\mathbb{E}\,(\hat{\phi}_{K,\epsilon} - \bar{\phi}\,)^2 \le \\ \mathbb{E}\Bigg[&\frac{(\mathbb{E}\,\psi_K - \psi_0))^2}{K^2\epsilon^2} + \frac{\sum_{k=1}^{K}(\mathbb{E}\,\psi_k - \psi_k)^2}{K^2\epsilon^2} + \frac{\sum_{k=1}^{K}\xi_k^2 + \sum_{k\neq k'=1}^{K}\xi_k\xi_{k'}}{K^2} \\ &+ \frac{\sum_{k=1}^{K}\xi_k}{K}\cdot\left(\frac{\mathbb{E}\,\psi_K - \psi_1}{K\epsilon} - \frac{\sum_{k=1}^{K}(\mathbb{E}\,\psi_k - \psi_k)}{K\epsilon} + \mathcal{O}\,(\epsilon)\right) + \mathcal{O}(\epsilon^2)\Bigg].\end{aligned} \quad (3.7.4)$$

The first line contains the squared terms and the second line contains the cross terms that do not go to zero. In particular, we do not assume $\hat{g}(\theta)$ is unbiased for $g(\theta)$, therefore we keep the cross-terms involving $\xi_k$. Bounding each term of (3.7.4) gives

the MSE bound

$$\begin{aligned}\mathbb{E}\,(\hat{\phi}_{K,\epsilon}-\bar{\phi})^2 &\leq \frac{4M^2}{K^2\epsilon^2}+\frac{K\cdot\mathcal{O}(\epsilon)}{K^2\epsilon^2}+\frac{KM^2\sigma^2+K^2M^2\delta^2}{K^2}\\ &\quad+M\delta\cdot\left(\frac{2M}{K\epsilon}+\frac{K\cdot 2M}{K\epsilon}+\mathcal{O}\left(\epsilon\right)\right)+\mathcal{O}(\epsilon^2)\\ &\leq C\cdot\left(\frac{1}{K^2\epsilon^2}+\frac{\sigma^2}{K}+\delta^2+\frac{\delta}{\epsilon}\right)+\mathcal{O}\left(\frac{1}{K\epsilon}+\delta\epsilon+\epsilon^2\right)\ . \qquad (3.7.5)\end{aligned}$$

□

### 3.7.2 Proof of Theorem 3.4.2

We now prove Theorem 3.4.2, which bounds the bias and MSE of our buffered stochastic gradient $g_\theta^{\mathrm{PF}}(S,B,N)$. In our proof, we use Lemma 3.7.1 to bound the subsequence error which is proved in Section 3.7.2.

*Proof of Theorem 3.4.2.* For the bias bound, (3.4.7), we apply the triangle inequality to decompose the error into three terms

$$\|\mathbb{E}\, g_\theta^{\mathrm{PF}}(S,B,N)-g_\theta\|\leq \qquad (3.7.6)$$
$$\underbrace{\|\mathbb{E}\,(g_\theta^{\mathrm{PF}}(S,B,N)-\hat{g}_\theta(S,B))\|}_{\text{particle bias (I)}}+\underbrace{\|\mathbb{E}\,(\hat{g}_\theta(S,B)-\hat{g}_\theta(S,T))\|}_{\text{buffering bias (II)}}+\underbrace{\|\mathbb{E}\,\hat{g}_\theta(S,T)-g_\theta\|}_{\text{subsequence bias (III)}},$$

where expectations are over the random subsequence $\mathcal{S}$ and particles. Each term is bounded separately (recalling that $\gamma=\max_t \Pr(t\in\mathcal{S})^{-1}$)

(I) *Particle bias*: the particle filter bias is $\mathcal{O}(\gamma\frac{S+2B}{N})$ (see Eq. 3.15 of Kantas et al. (2015)).

(II) *Buffering bias*: from Aicher et al. (2019), we know there exists a finite constant $C_1<\infty$ that is independent of $T,S,B,N$, such that

$$\mathbb{E}\,\|\hat{g}(S,B)-\hat{g}(S,T)\|\leq\gamma\cdot C_1\cdot(L_\theta)^B\ . \qquad (3.7.7)$$

Thus, the buffering bias can be upper bounded using Jensen's inequality

$$\begin{aligned}\|\mathbb{E}\,(\hat{g}(S,B)-\hat{g}(S,T))\| &\leq \mathbb{E}\,\|\hat{g}(S,B)-\hat{g}(S,T)\|\\ &\leq\gamma\cdot C_1\cdot(L_\theta)^B\ . \qquad (3.7.8)\end{aligned}$$

(III) *Subsequence bias*: For this term the randomness is only with respect to the choice of subsampler. By our our decomposition, the subsequence bias is zero, $\mathbb{E}\,\hat{g}_\theta(S,T) = g_\theta$, as the bias due to using a finite buffer is accounted for in (II).

Applying these bounds gives us the bias bound

$$\|\mathbb{E}\, g_\theta^{\mathrm{PF}}(S,B,N) - g_\theta\| \leq \gamma \cdot \left[C_1 (L_\theta)^B + \mathcal{O}\left(\frac{S+2B}{N}\right)\right] . \tag{3.7.9}$$

For the MSE bound, (3.4.8), we again apply the triangle inequality and recall that $2XY \leq X^2 + Y^2$ implies $(X+Y+Z)^2 \leq 3(X^2+Y^2+Z^2)$ to decompose the error into three terms

$$\begin{aligned}\mathbb{E}\,\|g_\theta^{\mathrm{PF}}(S,&B,N) - g_\theta\|^2 \leq \\ &3\underbrace{\mathbb{E}\,\|g_\theta^{\mathrm{PF}}(S,B,N) - \hat{g}_\theta(S,B)\|^2}_{\text{particle MSE (I)}} \\ &+\ 3\underbrace{\mathbb{E}\,\|\hat{g}_\theta(S,B) - \hat{g}_\theta(S,T)\|^2}_{\text{buffering MSE (II)}} \\ &+\ 3\underbrace{\mathbb{E}\,\|\hat{g}_\theta(S,T) - g_\theta\|^2}_{\text{subsequence MSE (III)}},\end{aligned} \tag{3.7.10}$$

where expectations are over the random subsequence $\mathcal{S}$ and particles. Again, each term is bounded separately,

(I) *Particle MSE*: the particle filter MSE bound is $\mathcal{O}(\gamma^2 \frac{(S+2B)^2}{N})$ (see Eq. 3.15 of Kantas et al. (2015)).

(II) *Buffering MSE*: from Aicher et al. (2019), the buffering MSE is bounded

$$\mathbb{E}\,\|\hat{g}_\theta(S,B) - \hat{g}_\theta(S,T)\|^2 \leq \gamma^2 \cdot C_1^2 \cdot (L_\theta)^{2B} \ . \tag{3.7.11}$$

(III) *Subsequence MSE*: from Lemma 3.7.1, there exists a constant $C_2 < \infty$ independent of $T, S, B, N$ such that

$$\mathbb{E}\,\|\hat{g}_\theta(S,T) - g_\theta\|^2 \leq \gamma^2 \cdot C_2 \cdot S \ . \tag{3.7.12}$$

Combining these bounds gives us the MSE bound

$$\mathbb{E}\,\|g_\theta^{\mathrm{PF}}(S,B,N) - g_\theta\|^2 \leq 3\gamma^2 \cdot \left[C_1^2 (L_\theta)^{2B} + C_2 S + \mathcal{O}\left(\frac{(S+2B)^2}{N}\right)\right] \ . \tag{3.7.13}$$

□

**Stochastic Subsequence MSE**

For the proof of Theorem 3.4.2, we bound the MSE between the full gradient $g_\theta$ and the unbiased stochastic gradient estimate $\hat{g}_\theta(S,T)$, specifically for the case of randomly sampling a *contiguous* subsequence $\mathcal{S}$. Because $\hat{g}_\theta(S,T)$ is unbiased for $g_\theta$, this reduces to calculating the variance of $\hat{g}_\theta(S,T)$ with respect to the sampling distribution of the subsequence $\mathcal{S}$.

Let $f_t$ denote the $t$-th gradient term in Fisher's identity

$$f_t = \mathbb{E}_{X_{1:T}|y_{1:T},\theta}[\nabla \log p(y_t, X_t \,|\, X_{t-1}, \theta)] \ . \tag{3.7.14}$$

Therefore

$$g_\theta = \sum_{t=1}^{T} f_t \quad \text{and} \quad \hat{g}_\theta(S,T) = \sum_{t\in\mathcal{S}} \Pr(t \in \mathcal{S})^{-1} \cdot f_t \ .$$

We now present the lemma that bounds the variance of $\hat{g}_\theta(S,T)$, under the assumption that the autocorrelation of $f_t$ decays geometrically $|\operatorname{Corr}(f_t, f_{t+s})| \le \rho^s$.

**Lemma 3.7.1.** *If for all $t$, the variance of $f_t$ is bounded and the autocorrelation of $f_t$ is geometrically bounded, then there exists a constant $C_2 < \infty$ (not dependent on $T, S, B$) such that*

$$\operatorname{Var}(\hat{g}_\theta(S,T)) \le \gamma^2 \cdot C_2 \cdot S \ . \tag{3.7.15}$$

The assumption that the autocorrelation of $f_t$ decays geometrically is reasonable when both the observations $Y_{1:T}$ and the posterior latent states $X_{1:T}|Y_{1:T}$ are *ergodic* (i.e., exhibit an exponential forgetting property common for most finite dimensional SSMs Chan and Palma (1998); Cappé et al. (2005)).

We now present the proof.

*Proof of Lemma 3.7.1.* Let $V < \infty$ be a bound on the variance of $f_t$ for all $t$ (i.e., $\operatorname{Var}(f_t) \le V$). Let $\rho \in [0,1)$ be a bound on the geometric decay of the autocorrelation of $f_t$. Then we have $|\operatorname{Corr}(f_t, f_{t+s})| \le \rho^s$ for all $t$ and $s \in \mathbb{N}$, Together these bounds imply a bound on the covariance between any $f_t$ and $f_{t+s}$

$$\operatorname{CoV}(f_t, f_{t+s}) \le |\operatorname{Corr}(f_t, f_{t+s})| \cdot \sqrt{\operatorname{Var}(f_t) \cdot \operatorname{Var}(f_{t+s})} \le V\rho^s \ . \tag{3.7.16}$$

Then we have

$$\begin{aligned}
\mathrm{Var}(\hat{g}_\theta(S,T)) &\le \gamma^2 \cdot \mathrm{Var}\left[\sum_{t\in\mathcal{S}} f_t\right] \\
&= \gamma^2 \cdot \left[\sum_{t\in\mathcal{S}} \mathrm{Var}(f_t) + \sum_{t\neq t'\in\mathcal{S}} \mathrm{CoV}(f_t, f_{t'})\right] \\
&\le \gamma^2 \cdot \left[S\cdot V + \sum_{s=1}^{S-1} 2(S-s)\cdot V\rho^s\right] \\
&= \gamma^2 \cdot S \cdot \left[V + 2V\sum_{s=1}^{S-1}(1-s/S)\cdot\rho^s\right] \\
&\le \gamma^2 \cdot S \cdot \left[2V\sum_{s=0}^{S-1}\rho^s\right] \\
&\le \gamma^2 \cdot S \cdot 2V/(1-\rho) \ . \qquad (3.7.17)
\end{aligned}$$

As $S \ge 1$, if $C_2 = 2V/(1-\rho)$, we have

$$\mathbb{E}\,\|\hat{g}_\theta(S,T) - g_\theta\|^2 = \mathrm{Var}(\hat{g}_\theta(S,T)) \le \gamma^2 \cdot S \cdot C_2 \ . \qquad (3.7.18)$$

□

### 3.7.3 Proof of Theorem 3.4.3

Theorem 3.4.3 states that if the prior distribution for $x_0$, the transition distribution $p(x_t \,|\, x_{t-1}, \theta)$ and the emission distribution $p(y_t \,|\, x_t)$ are log-concave, then we can bound the Lipschitz constant of $\vec{\Psi}_t$ in terms of $\vec{\Psi}_t^{(0)}$ and $\vec{\Psi}_t^{(1)}$.

We first briefly review Wasserstein distance, random mappings, and Lipschitz constants of kernels Villani (2008); Aicher et al. (2019). Then we review *Caffarelli's log-concave perturbation theorem*, the main tool we use in our proof. Finally, we present the proof in Section 3.7.3.

#### Wasserstein Distance and Random Maps

The *p-Wasserstein distance* with respect to Euclidean distance is

$$\mathcal{W}_p(\gamma, \widetilde{\gamma}) := \left[\inf_{\xi} \int \|x - \widetilde{x}\|_2^p \, d\xi(x, \widetilde{x})\right]^{1/p} \ , \qquad (3.7.19)$$

where $\xi$ is a joint measure or *coupling* over $(x, \widetilde{x})$ with marginals $\int_{\widetilde{x}} d\xi(x, \widetilde{x}) = d\gamma(x)$ and $\int_x d\xi(x, \widetilde{x}) = d\widetilde{\gamma}(x)$.

To bound the Wasserstein distance, we first must introduce the concept of a *random mapping* associated with a transition kernel.

Let $\Psi : \mathcal{U} \to \mathcal{V}$ be a transition kernel for random variables $u$ and $v$, then for any measure $\mu(u)$ over $\mathcal{U}$, we define the induced measure $(\mu\Psi)(v)$ over $\mathcal{V}$ as $(\mu\Psi)(v) = \int \Psi(u, v)\mu(du)$.

A *random mapping* $\psi$ is a random function that maps $\mathcal{U}$ to $\mathcal{V}$ such that if $u \sim \mu$ then $\psi(u) \sim \mu\Psi$. For example, if $\Psi(u, v) = \mathcal{N}(v \,|\, u, 1)$, then a random mapping for $\Psi$ is the identity function plus Gaussian noise $\psi(u) = u + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$. If $\psi$ is deterministic $(\mu\Psi)(v)$ is the *push-forward* measure of $\mu$ through the mapping $\psi$; otherwise it is the average (or marginal) over $\psi$ of push-forward measures Villani (2008).

We say the kernel has Lipschitz constant $L$ with respect to Euclidean distance if

$$\|\Psi\|_{Lip} = L \Leftrightarrow \sup_{u,u'} \left\{ \frac{\mathbb{E}_\psi[\|\psi(u) - \psi(u')\|_2]}{\|u - u'\|_2} \right\} \leq L \ . \tag{3.7.20}$$

Note that $L$ is an upper-bound on the *expected value* of Lipschitz constants for random instances of $\psi$.

These definitions are useful for proving bounds in Wasserstein distance. For example, we can show the kernel $\Psi$ induces a contraction in $p$-Wasserstein distance if $\|\Psi\|_{Lip} < 1$. That is $\mathcal{W}_p(\mu\Psi, \tilde{\mu}\Psi) \leq \|\Psi\|_{Lip} \cdot \mathcal{W}_p(\mu, \tilde{\mu})$

$$\begin{aligned}
\mathcal{W}_p(\mu\Psi, \tilde{\mu}\Psi)^p &= \inf_{\xi(\mu\Psi, \tilde{\mu}\Psi)} \int \|v - \tilde{v}\|_2^p \, d\xi(v, \tilde{v}) \\
&\leq \inf_{\xi(\mu, \tilde{\mu})} \int \|\psi(u) - \psi(\tilde{u})\|_2^p \, d\xi(u, \tilde{u}) d\mu(\psi) \\
&\leq \inf_{\xi(\mu, \tilde{\mu})} \int \|\Psi\|_{Lip}^p \cdot \|u - \tilde{u}\|_2^p \, d\xi(u, \tilde{u}) \\
&= \|\Psi\|_{Lip}^p \cdot \mathcal{W}_p(\mu, \tilde{\mu})^p \ ,
\end{aligned} \tag{3.7.21}$$

where in the second line we replace $v, \tilde{v}$ with a random mapping $\psi$ that has measure $d\mu(\psi)$ and in the third line we bound $\psi$ by its Lipschitz constant $L$ and integrate $\int d\mu(\psi) = 1$.

### Caffarelli's Perturbation Theorem

Caffarelli's log-concave perturbation theorem allows us to connect Lipschitz constants between kernels that are log-concave perturbations of one another.

**Theorem 3.7.2** (Caffarelli's)**.** *Suppose $\gamma(x)$ is a log-concave measure for $x$ and $\ell(x)$ is a log-concave function such that $\gamma'(x) = \ell(x)\gamma(x)$ is a probability measure over $x$. Then there exists a 1-Lipschitz mapping $T : \mathcal{X} \to \mathcal{X}$ such that if $x \sim \gamma(x)$ then $T(x) \sim \gamma'(x)$.*

We can think of $\gamma(x)$ as a prior distribution $p(x)$, $\ell(x)$ as a normalized conditional likelihood $p(y|x)/p(y)$ and $\gamma'(x)$ as the posterior $p(x|y)$. As $\ell(x)$ is log-concave, we call $\gamma'(x)$ a *log-concave perturbation* of $\gamma$.

The original version of Caffarelli's log-concave perturbation theorem Colombo et al. (2017); Saumard and Wellner (2014) requires the prior $\gamma(x)$ to be *strongly* log-concave (e.g., a Gaussian) to show that the mapping $T$ is a *strict* contraction $\|T\|_{Lip} < 1$; however this weaker version, Theorem 3.7.2 of Villani (2008), is sufficient for our purposes.

### Proof of Theorem 3.4.3

Using Theorem 3.7.2, we can now prove Theorem 3.4.3.

*Proof of Theorem 3.4.3.* Let $\vec{\psi}_t, \vec{\psi}_t^{(0)}, \vec{\psi}_t^{(1)}$ be random mappings associated respectively with forward kernels $\vec{\Psi}_t, \vec{\Psi}_t^{(0)}, \vec{\Psi}_t^{(1)}$. Because the transition and emission distributions are log-concave and log-concavity is preserved under product and marginalization Saumard and Wellner (2014), $\vec{\Psi}_t$, $\vec{\Psi}_t^{(0)}$, $\vec{\Psi}^{(1)}$ are log-concave and $p(y_{t\geq} \,|\, x_t)$ and $p(y_{>t} \,|\, x_t)$ are also log-concave (where $y_{t\geq} = \{y_t, \ldots, y_T\}$ and $y_{>t} = \{y_{t+1}, \ldots, y_T\}$).

Since $p(y_{\geq t} \,|\, x_t)$ is log-concave, we can write $\vec{\Psi}_t$ as a log-concave perturbation of $\vec{\Psi}_t^{(0)}$,

$$\begin{aligned} \vec{\Psi}_t = p(x_t \,|\, x_{t-1}, y_{t:T}, \theta) &\propto p(y_{\geq t} \,|\, x_t) p(x_t \,|\, x_{t-1}, \theta) \\ &= p(y_{\geq t} \,|\, x_t) \cdot \vec{\Psi}_t^{(0)} \quad . \end{aligned} \tag{3.7.22}$$

Therefore, there exists $T_t^{(0)}$ with $\|T_t^{(0)}\|_{Lip} \leq 1$ such that $\vec{\psi}_t = (T_t^{(0)} \circ \vec{\psi}_t^{(0)})$. Thus,

$$\|\vec{\Psi}_t\|_{Lip} = \|T_t^{(0)}\|_{Lip} \cdot \|\vec{\Psi}_t^{(0)}\|_{Lip} \leq \|\vec{\Psi}_t^{(0)}\|_{Lip} \ . \tag{3.7.23}$$

Similarly, we can write $\vec{\Psi}_t$ as a log-concave perturbation of $\vec{\Psi}_t^{(1)}$ using $p(y_{>t} \,|\, x_t)$, thus $\|\vec{\Psi}_t\|_{Lip} \leq \|\vec{\Psi}_t^{(1)}\|_{Lip}$.

$$\begin{aligned} \vec{\Psi}_t = p(x_t \,|\, x_{t-1}, y_{t:T}, \theta) &\propto p(y_{>t} \,|\, x_t) p(x_t \,|\, y_t, x_{t-1}, \theta) \\ &= p(y_{>t} \,|\, x_t) \cdot \vec{\Psi}_t^{(1)} \ . \end{aligned} \tag{3.7.24}$$

□

Note the assumptions for equivalent results in the backward smoothers $\overleftarrow{\Psi}_t$ are identical. Log-concavity in $p(x_t \,|\, x_{t+1}, \theta)$ is implied from both $p(x_t \,|\, x_{t-1}, \theta)$ and the prior $p(x_t)$ being log-concave.

### 3.7.4 Bounds for Specific Models

We now provide specific bounds for the buffering error for models we consider in Section 3.5 (LGSSM and SVM) using Theorem 3.4.3.

For both the LGSSM and SVM, we assume the prior $\nu(x_0|\theta) = \mathcal{N}(0, \sigma^2/(1-\phi^2))$. Then the latent state transitions are

$$\begin{aligned} p(x_t \,|\, x_{t-1}, \theta) &= \mathcal{N}(x_t \,|\, \phi x_{t-1}, \sigma^2) \\ p(x_t \,|\, x_{t+1}, \theta) &= \mathcal{N}(x_t \,|\, \phi x_{t+1}, \sigma^2) \ , \end{aligned}$$

which are both Gaussian and therefore log-concave in $x$.

Similarly, the emissions for the LGSSM and SVM are also log-concave in $x$:
For the LGSSM,

$$p(y_t \,|\, x_t, \theta) \propto \exp\left(-\tfrac{(y_t - x_t)^2}{2\sigma^2}\right) \ ,$$

which is log-concave.
For the SVM,

$$p(y_t \,|\, x_t, \theta) \propto \exp\left(-\tfrac{y_t^2}{2\sigma^2} \cdot e^{-x_t} - \tfrac{x_t}{2}\right) \ ,$$

which is log-concave as $e^{-x}$ is convex.

### Contraction Bound for LGSSM

We assume the prior $\nu(x_0|\theta) = \mathcal{N}(0, \sigma^2/(1-\phi^2))$. For the LGSSM, the filtered kernels are

$$\begin{aligned}
\vec{\Psi}_t^{(1)}(x_t \,|\, x_{t-1}) &= p(x_t \,|\, x_{t-1}, y_t, \theta) \\
&\propto \mathcal{N}(x_t|\phi x_{t-1}, \sigma^2) \cdot \mathcal{N}(y_t|x_t, \tau^2), \\
\overleftarrow{\Psi}_t^{(1)}(x_t \,|\, x_{t+1}) &= p(x_t \,|\, x_{t+1}, y_t, \theta) \\
&\propto \mathcal{N}(x_t|\phi x_{t+1}, \sigma^2) \cdot \mathcal{N}(y_t|x_t, \tau^2).
\end{aligned} \tag{3.7.25}$$

Therefore,

$$\begin{aligned}
\vec{\Psi}_t^{(1)}(x_t \,|\, x_{t-1}) &= \mathcal{N}\left(x_t \,\middle|\, \frac{\sigma^2 y_t + \phi\tau^2 x_{t-1}}{\sigma^2+\tau^2}\,,\, \frac{\sigma^2\tau^2}{\sigma^2+\tau^2}\right), \\
\overleftarrow{\Psi}_t^{(1)}(x_t \,|\, x_{t+1}) &= \mathcal{N}\left(x_t \,\middle|\, \frac{\sigma^2 y_t + \phi\tau^2 x_{t+1}}{\sigma^2+\tau^2}\,,\, \frac{\sigma^2\tau^2}{\sigma^2+\tau^2}\right).
\end{aligned} \tag{3.7.26}$$

The associated random mapping are,

$$\begin{aligned}
\vec{\psi}_t^{(1)}(x_t \,|\, x_{t-1}) &= \frac{\sigma^2 y_t}{\sigma^2+\tau^2} + \frac{\phi\tau^2}{\sigma^2+\tau^2} \cdot x_{t-1} + \vec{z}_t\,, \\
\overleftarrow{\psi}_t^{(1)}(x_t \,|\, x_{t+1}) &= \frac{\sigma^2 y_t}{\sigma^2+\tau^2} + \frac{\phi\tau^2}{\sigma^2+\tau^2} \cdot x_{t+1} + \overleftarrow{z}_t\,,
\end{aligned} \tag{3.7.27}$$

where $\vec{z}_t$ and $\overleftarrow{z}_t$ are $\mathcal{N}\left(0\,,\, \frac{\sigma^2\tau^2}{\sigma^2+\tau^2}\right)$ random variables.

Since these maps are linear, we have $\|\vec{\Psi}_t^{(1)}\|_{Lip} = \|\overleftarrow{\Psi}_t^{(1)}\|_{Lip} = |\phi| \cdot \frac{\tau^2}{\sigma^2+\tau^2}$. Applying Theorem 3.4.3, we obtain

$$L_\theta \leq \max_t\{\|\vec{\Psi}_t^{(1)}\|, \|\overleftarrow{\Psi}_t^{(1)}\|\} = |\phi| \cdot (1 + \sigma^2/\tau^2)^{-1}. \tag{3.7.28}$$

Therefore $L_\theta < 1$ whenever $|\phi| < 1 + \sigma^2/\tau^2$.

### Contraction Bound for SVM

We assume the prior $\nu(x_0\theta) = \mathcal{N}(0, \sigma^2/(1-\phi^2))$. For the SVM, the prior kernels are,

$$\begin{aligned}
\vec{\Psi}_t^{(0)}(x_t \,|\, x_{t-1}) &= p(x_t \,|\, x_{t-1}, \theta) \propto \mathcal{N}(x_t|\phi x_{t-1}, \sigma^2), \\
\overleftarrow{\Psi}_t^{(0)}(x_t \,|\, x_{t+1}) &= p(x_t \,|\, x_{t+1}, \theta) \propto \mathcal{N}(x_t|\phi x_{t+1}, \sigma^2).
\end{aligned} \tag{3.7.29}$$

The associated random mapping are

$$\begin{aligned}\vec{\psi}_t^{(0)}(x_t \,|\, x_{t-1}) &= \phi \cdot x_{t-1} + \mathcal{N}\left(0\,,\,\sigma^2\right),\\ \overleftarrow{\psi}_t^{(0)}(x_t \,|\, x_{t+1}) &= \phi \cdot x_{t+1} + \mathcal{N}\left(0\,,\,\sigma^2\right).\end{aligned} \tag{3.7.30}$$

Applying Theorem 3.4.3, we obtain $L_\theta \leq |\phi|$.

# Chapter 4

# Preferential Subsampling for Stochastic Gradient Langevin Dynamics

## 4.1 Introduction

Markov chain Monte Carlo (MCMC) algorithms are a popular family of methods to conduct Bayesian inference. Unfortunately, running MCMC on large datasets is generally computationally expensive, which often limits the use of MCMC by practitioners. The Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970), in particular, requires a scan of the full dataset at each iteration to calculate the acceptance probability.

Stochastic gradient Markov chain Monte Carlo (SGMCMC) algorithms are a family of scalable methods, which aim to address this issue (Welling and Teh, 2011; Nemeth and Fearnhead, 2021). These algorithms aim to leverage the efficiency of gradient-based MCMC proposals (Roberts and Tweedie, 1996; Neal, 2011). They reduce the per-iteration computational cost by constructing an unbiased, noisy estimate of the gradient of the log-posterior, using only a small data subsample. In this chapter, we focus on samplers that rely on the overdamped Langevin diffusion (Roberts and Tweedie, 1996), however, our proposed methodology can be applied more generally

to other SGMCMC algorithms, such as those based on Hamiltonian dynamics (Chen et al., 2014).

Thereom 4 of Dalalyan and Karagulyan (2019) demonstrates that the non-asymptotic convergence rates of SGLD-type algorithms depend upon: (i) bounded stochastic gradient bias, (ii) bounded stochastic gradient variance, and (iii) and independent random updates. Although in Chapter 3 we saw that bias control is important, it is also important to reduce the variance for unbiased stochastic gradient methods. The higher variance inherently present from utilising smaller subsamples can degrade sampler performance and lead to poor convergence. As such, variance control has become an important area of research within the SGMCMC literature, and is often required to make these algorithms practical (Dubey et al., 2016; Chatterji et al., 2018; Baker et al., 2019a; Chen et al., 2019).

In this chapter, we propose a new method designed to reduce the variance in the stochastic gradient. We use a discrete, non-uniform probability distribution to preferentially subsample data points and to re-weight the stochastic gradient. In addition, we present a method for adaptively adjusting the size of the subsample chosen at each iteration.

## 4.2 Stochastic gradient MCMC

Let $\theta \in \mathbb{R}^d$ be a parameter vector and denote independent observations $\mathbf{x} = \{x_i\}_{i=1}^N$ ($N \gg 1$). The probability density of the $i$-th observation, given parameter $\theta$, is $p(x_i|\theta)$ and the prior density for the parameters is $p(\theta)$. In a Bayesian context, the target of interest is the posterior density, $\pi(\theta) := p(\theta|\mathbf{x}) \propto p(\theta) \prod_{i=1}^N p(x_i|\theta)$.

For convenience, we define $f_i(\theta) = -\log p(x_i|\theta)$ for $i = 1, \ldots, N$, with $f_0(\theta) = -\log p(\theta)$ and $f(\theta) = f_0(\theta) + \sum_{i=1}^N f_i(\theta)$. In this setting, the posterior density can be rewritten as, $\pi(\theta) \propto \exp(-f(\theta))$.

### 4.2.1 The Langevin diffusion

The Langevin diffusion, $\theta(t)$, is defined by the stochastic differential equation,

$$d\theta(t) = -\frac{1}{2}\nabla f(\theta(t))dt + dB_t, \tag{4.2.1}$$

where $\nabla f(\theta(t))dt$ is a drift term and $B_t$ denotes a $d$-dimensional Wiener process. Under certain regularity conditions, the stationary distribution of this diffusion is the posterior $\pi$ (Roberts and Tweedie, 1996). In practice, we need to discretise (4.2.1) in order to simulate from it and this introduces error. For a small step-size $\epsilon > 0$, the Langevin diffusion can be approximated by

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\epsilon}{2}\nabla f\left(\theta^{(t)}\right) + \sqrt{\epsilon}\,\eta^{(t)}, \tag{4.2.2}$$

where the noise $\eta^{(t)} \sim \mathcal{N}_d(\mathbf{0}, I_{d\times d})$ is drawn independently at each update. The dynamics implied by (4.2.2) provide a simple way to sample from the Langevin diffusion. The level of discretisation error in the approximation is controlled by the size of $\epsilon$ and we can achieve any required degree of accuracy if we choose $\epsilon$ small enough.

The unadjusted Langevin algorithm (ULA) (Parisi, 1981) is a simple sampler that simulates from (4.2.2) but does not use a Metropolis-Hastings correction (Metropolis et al., 1953; Hastings, 1970). Thus, the samples obtained from ULA produce a biased approximation of $\pi$. The per-iteration computational cost of ULA is smaller than that of the Metropolis-adjusted Langevin algorithm (Roberts and Rosenthal, 1998) due to the removal of the Metropolis-Hastings step. However, the computational bottleneck for ULA lies in the $O(N)$ calculation of the full data gradient $\nabla f\left(\theta^{(t)}\right) = \nabla f_0\left(\theta^{(t)}\right) + \sum_{i=1}^{N} \nabla f_i\left(\theta^{(t)}\right)$ at every iteration. This calculation can be problematic if $N$ is large.

### 4.2.2 Stochastic gradient Langevin dynamics

The stochastic gradient Langevin dynamics (SGLD) algorithm attempts to improve the per-iteration computational burden of ULA by replacing the full-data gradient with an unbiased estimate (Welling and Teh, 2011). Let the full-data gradient of $f(\theta)$

be given by

$$g^{(t)} = \nabla f\big(\theta^{(t)}\big) = \nabla f_0\big(\theta^{(t)}\big) + \sum_{i=1}^{N} \nabla f_i\big(\theta^{(t)}\big).$$

The unbiased estimate of $g^{(t)}$ proposed by Welling and Teh (2011) takes the form

$$\hat{g}^{(t)} = \nabla f_0\big(\theta^{(t)}\big) + \frac{N}{n} \sum_{i \in \mathcal{S}^t} \nabla f_i\big(\theta^{(t)}\big), \tag{4.2.3}$$

where $\mathcal{S}^t$ is a subset of $\{1, \ldots, N\}$ and $|\mathcal{S}^t| = n$ ($n \ll N$) is the subsample size. A single update of SGLD is thus given by,

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \frac{\epsilon^{(t)}}{2} \cdot \hat{g}^{(t)} + \xi^{(t)}, \tag{4.2.4}$$

where $\xi^{(t)} \sim \mathcal{N}_d(0, \epsilon^{(t)} I_{d \times d})$ and $\{\epsilon^{(t)}\}$ corresponds to a schedule of step-sizes which may be fixed (Vollmer et al., 2016) or decreasing (Teh et al., 2016). The full SGLD pseudocode is provided in Algorithm 4.

**Algorithm 4:** SGLD

1: Input: initialise $\theta^{(1)}$, batch size $n$, step-sizes $\{\epsilon^{(t)}\}$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Sample indices $S^t \subset \{1, \ldots, N\}$ with or without replacement.
4: Calculate $\hat{g}^{(t)}$ using (4.2.3).
5: Update parameters according to (4.2.4).
6: **end for**
7: **return** $\theta^{(T+1)}$

Welling and Teh (2011) note that if the step-size $\epsilon^{(t)} \to 0$ as $t \to \infty$, then the Gaussian noise (generated by $\xi^{(t)}$) dominates the noise in the stochastic gradient term. For large $t$, the algorithm approximately samples from the posterior using an increasingly accurate discretisation of the Langevin diffusion. In practice, SGLD does not mix well when the step-size is decreased to zero and so a small fixed step-size $\epsilon$ is typically used instead.

### 4.2.3 Control variates for SGLD

The naive stochastic gradient proposed by Welling and Teh (2011) may exhibit a relatively high variance for small subsamples of data. The more faithful a stochastic gradient estimator is to the full-data gradient, the better we can expect SGLD to perform. Therefore, it is natural to consider alternatives to the estimator given in (4.2.3) which minimise the variance.

Let $\hat{\theta}$ be a fixed value of the parameter, typically chosen to be close to the mode of the target posterior density. The control variate gradient estimator proposed by Baker et al. (2019a) takes the form,

$$\hat{g}_{cv}^{(t)} = \big[\nabla f(\hat{\theta}) + \nabla f_0\big(\theta^{(t)}\big) - \nabla f_0(\hat{\theta})\big] + \frac{N}{n}\sum_{i\in\mathcal{S}^t}\big[\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta})\big]. \tag{4.2.5}$$

When $\theta^{(t)}$ is close to $\hat{\theta}$, the variance of the gradient estimator will be small. This is shown formally in Lemma 1 of Baker et al. (2019a).

The SGLD-CV algorithm is the same as SGLD given in Algorithm 4, except with $\hat{g}_{cv}^{(t)}$ substituted in place of $\hat{g}^{(t)}$. Implementing the SGLD-CV estimator involves a one-off pre-processing step to find $\hat{\theta}$, which is typically done using stochastic gradient descent (SGD) (Bottou et al., 2018; Baker et al., 2019a). The gradient terms $\nabla f_i\big(\hat{\theta}\big)$ are calculated and stored. While these steps are both $O(N)$ in computational cost, the optimisation step to find the mode can replace the typical burn-in phase of the SGLD chain. The MCMC chain can then be initialised at the posterior mode itself. The full pseudocode for SGLD-CV is provided in Algorithm 3 within Appendix B.1.

## 4.3 Preferential data subsampling

Let $\mathcal{S}$ be a subsample of size $n$ generated with replacement, such that the probability that the $i$-th data point, $x_i$, appears in $\mathcal{S}$ is $p_i$. The expected number of times $x_i$ is drawn from the dataset is $np_i$. A standard implementation of SGLD would assume uniform subsampling, i.e., that $p_i = \frac{1}{N}$ for all $i$. However, if the observations vary in

their information about the parameters, then assigning a larger probability to the more informative observations would be advantageous. Preferential subsampling assigns a strictly-positive, user-chosen weight to each data point, such that we minimise the variance of the estimator of the gradient.

We need to construct a discrete probability distribution $\mathbf{p}^{(t)} = (p_1^t, \ldots, p_N^t)^T$ (where $p_i^{(t)} > 0$ for all $i$ and $\sum_i p_i^t = 1$) that can be used to draw subsamples of size $n$ at each iteration $t$ and to reweight the stochastic gradient accordingly. If $\mathbf{p}^{(t)}$ is time-invariant (i.e., $p_i^{(t)} = p_i$ for all $i$), the preferential subsampling scheme is static. Otherwise, the subsampling weights will be dynamic or state-dependent.

For a given stochastic gradient $\tilde{g}$, the noise term associated with $\tilde{g}$ is given by $\xi^{(t)} = \tilde{g}^{(t)} - g^{(t)}$. Taking expectations over $\mathbf{p}^{(t)}$, a simple scalar summary of the variance of the noise $\xi^{(t)}$ can be found by evaluating:

$$\mathbb{E}\left(\left\|\xi^{(t)}\right\|^2\right) = \operatorname{tr}\left(\operatorname{Cov}\left(\tilde{g}^{(t)}\right)\right). \tag{4.3.1}$$

We will refer to (4.3.1) as the *pseudo-variance*[1] of $\tilde{g}^{(t)}$, $\mathbb{V}\big(\tilde{g}^{(t)}\big)$, from now on. We intend to use the pseudo-variance as a proxy for the variance of the stochastic gradient.[2] In all further analysis, $\|\cdot\|$ refers to the Euclidean norm.

In order to minimise the pseudo-variance, we need to find a preferential subsampling distribution $\mathbf{p}^*$ which minimises the following problem:

$$\min_{\mathbf{p}^{(t)},\, p_i^t \in [0,1], \sum_i p_i^t = 1} \mathbb{V}\big(\tilde{g}^{(t)}\big). \tag{4.3.2}$$

Existing non-asymptotic convergence results for SGLD-type methods[3] demonstrate the importance in controlling the variance of the stochastic gradient. These results give the error of SGLD in terms of bounds on the (bias and) variance of the estimator of the gradient of the log posterior. Therefore, constructing a better stochastic gradient

[1] See Section 4.7.1 for a full derivation of the pseudo-variance.

[2] Note that $\tilde{g}$ is a $d$-dimensional random vector (where typically $d > 1$). (4.3.1) is the sum of the variances of the elements of $\tilde{g}$. The term "pseudo-variance" allows us to easily distinguish between (4.3.1) and the variance-covariance matrix of $\tilde{g}$.

[3] Theorem 4 of Dalalyan and Karagulyan (2019) and Theorem 1 of Baker et al. (2019a) are two such examples.

estimator - for instance via preferential subsampling - will lead to a reduction in the error bound of the underlying SGLD method.

### 4.3.1 SGLD with preferential subsampling

An alternative gradient estimator for SGLD can be given by reweighting the simple estimator defined in (4.2.3) (Welling and Teh, 2011),

$$\tilde{g}^{(t)} = \nabla f_0\big(\theta^{(t)}\big) + \frac{1}{n}\sum_{i\in\mathcal{S}^t}\frac{1}{p_i^t}\nabla f_i\big(\theta^{(t)}\big), \tag{4.3.3}$$

where $S^t \subset \{1,\ldots,N\}$ is selected according to $\mathbf{p}^{(t)}$ and $|S^t| = n$ ($n \ll N$). The pseudocode for the SGLD with preferential subsampling (SGLD-PS) algorithm is outlined in Algorithm 8 within Appendix B.1.

As we correct for the non-uniform subsampling of data points by reweighting each gradient term, it follows that the stochastic gradient estimator given in (4.3.3) is unbiased. This is synonymous with the standard properties of importance sampling estimators (Robert and Casella, 2004). We note that there is an extra $O(n)$ computational cost associated with reweighting the stochastic gradient in this manner at each iteration.

The following result obtains the optimal solution to Problem (4.3.2).

**Lemma 4.3.1.** *For the unbiased SGLD-PS gradient estimator in* (4.3.3)*, minimising Problem* (4.3.2) *is equivalent to minimising the following*

$$\min_{\mathbf{p}^{(t)}} \frac{1}{n}\sum_{i=1}^{N}\frac{1}{p_i^t}\big\|\nabla f_i\big(\theta^{(t)}\big)\big\|^2. \tag{4.3.4}$$

*The optimal weights which minimise the pseudo-variance are thus given by*

$$p_i^t = \frac{\|\nabla f_i\big(\theta^{(t)}\big)\|}{\sum_{k=1}^{N}\|\nabla f_k\big(\theta^{(t)}\big)\|} \quad \textit{for } i = 1,\ldots,N. \tag{4.3.5}$$

Although a solution to (4.3.4) can be found, the resulting sampling algorithm would not be practical, as the optimal weights depend on the current state $\theta^{(t)}$. Therefore the subsampling distribution given in (4.3.5) requires $N$ gradient calculations per iteration.

For large datasets, these weights would be very expensive to store and calculate at each iteration, making the algorithm impractical.

We can instead approximate the optimal weights given in (4.3.5), such that they are not state-dependent and therefore do not need to be updated at each iteration. These approximate weights can be calculated as an initial pre-processing step before the main sampling algorithm is run.

A fairly simple approximation of the optimal weights given in (4.3.5) for SGLD-PS would require substituting the current state $\theta^{(t)}$ with some alternative fixed point. The posterior mode, $\hat{\theta}$, is a sensible choice as it represents the most probable estimate of the parameters in a Bayesian paradigm. In this case, the approximate subsampling scheme would be given by,

$$p_i = \frac{\|\nabla f_i(\hat{\theta})\|}{\sum_{k=1}^{N} \|\nabla f_k(\hat{\theta})\|} \text{ for } i = 1, \ldots, N. \tag{4.3.6}$$

As with SGLD-CV, the posterior mode could be estimated using SGD and the MCMC chain could then be initialised at the posterior mode.

In practice, the subsampling weights in (4.3.6) are calculated only once with an $O(N)$ preprocessing step and then used statically (i.e., without update). The resulting SGLD-PS algorithm would calculate the stochastic gradient given in (4.3.3) using these fixed weights.

### 4.3.2 SGLD-CV with preferential subsampling

The control-variates gradient estimator can be modified to accommodate a preferential subsampling scheme in a similar manner. In this case, we would obtain

$$\begin{aligned} \tilde{g}^{(t)} = \big[\nabla f(\hat{\theta}) + \nabla f_0\big(\theta^{(t)}\big) - \nabla f_0(\hat{\theta})\big] + \\ \frac{1}{n} \sum_{i \in \mathcal{S}^t} \frac{1}{p_i^t} \big[\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta})\big]. \end{aligned} \tag{4.3.7}$$

The pseudocode for the modified SGLD-CV algorithm (SGLD-CV-PS) is given in Algorithm 5 within Appendix B.1. The following result provides the optimal solution to Problem (4.3.2).

**Lemma 4.3.2.** *For the unbiased SGLD-CV-PS gradient estimator in* (4.3.7)*, minimising Problem* (4.3.2) *is equivalent to minimising the following*

$$\min_{\mathbf{p}^{(t)}} \frac{1}{n} \sum_{i=1}^{N} \frac{1}{p_i^t} \left\| \nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta}) \right\|^2. \tag{4.3.8}$$

*The optimal weights which minimise the pseudo-variance are thus given by*

$$p_i^t = \frac{\|\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta}))\|}{\sum_{k=1}^{N} \|\nabla f_k\big(\theta^{(t)}\big) - \nabla f_k(\hat{\theta})\|} \tag{4.3.9}$$

*for* $i = 1, \ldots, N$.

As in Section 4.3.1, we can derive a solution to (4.3.8). However, the resulting sampling algorithm would once again depend on the current state of the chain $\theta^{(t)}$. The process of finding a suitable approximation to the optimal weights given in (4.3.9) for the control-variate gradient estimator is non-trivial. Our approach will be to choose a set of subsampling weights that could be used for all iterations of the MCMC chain.

We consider an alternative minimisation problem

$$\min_{\mathbf{p}^{(t)},\, p_i^t \in [0,1], \sum_i p_i^t = 1} \mathbb{E}_\theta \left[ \mathbb{V}\big(\tilde{g}^{(t)}\big) \right], \tag{4.3.10}$$

where the outer expectation is taken with respect to the posterior distribution. Due to the linearity of expectation, (4.3.10) is equivalent to solving the following problem:

$$\min_{\mathbf{p}^{(t)}} \mathbb{E}_\theta \left[ \frac{1}{n} \sum_{i=1}^{N} \frac{1}{p_i^t} \left\| \nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta}) \right\|^2 \right].$$

This can easily be shown by using a modified version of the argument given for Lemmas 4.3.1 and 4.3.2. The optimal subsampling weights in (4.3.9) can be approximated by

$$p_i \propto \sqrt{\operatorname{tr}\left( \nabla^2 f_i(\hat{\theta})\, \hat{\Sigma}\, \nabla^2 f_i(\hat{\theta})^T \right)} \text{ for } i = 1, \ldots, N, \tag{4.3.11}$$

where $\nabla^2 f_i(\cdot)$ is the Hessian matrix of $f_i(\cdot)$ and $\hat{\Sigma}$ is the covariance matrix of the Gaussian approximation to the target posterior centred at the mode.

See Section 4.7.4 for a full discussion of how these weights can be obtained analytically and Appendix B.3 for an assessment of the computational cost associated with

calculating them as a preprocessing step. The computational cost required to calculate the Hessian matrix means that this approach can be computationally expensive for high-dimensional parameters.

### 4.3.3 Adaptive subsampling

In this section, we present a method for adaptively adjusting the size of the subsample chosen at each iteration. We do this by first finding an upper bound for the pseudo-variance of the stochastic gradient estimator given in (4.3.7) and then by rearranging the result to find a lower bound on the subsample size.

Let us begin by placing a Lipschitz condition on each of the likelihood terms.

**Assumption 2.** *(Lipschitz continuity of gradients)*
*There exists constants $L_0, \ldots, L_N$ such that*

$$\|\nabla f_i(\theta) - \nabla f_i(\theta')\| \leq L_i \|\theta - \theta'\| \tag{4.3.12}$$

*for $i = 0, \ldots, N$.*

We can then obtain the following result using Assumption 2.

**Lemma 4.3.3.** *Under Assumption 2, the pseudo-variance of the stochastic gradient estimator defined in* (4.3.7) *can be bounded above by*

$$\mathbb{V}(\tilde{g}) \leq \frac{1}{n} \|\theta^{(t)} - \hat{\theta}\|^2 \left( \sum_{i=1}^{N} \frac{L_i^2}{p_i^t} \right). \tag{4.3.13}$$

*where $\mathbf{p}^{(t)} = (p_1^t, \ldots, p_N^t)^T$ is a set of user-defined discrete weights.*

We can minimise the upper bound provided in (4.3.13) if we plug in the optimal weights given in (4.3.9). In practice, however, it is advantageous to choose the preferential subsampling scheme based on its ease of computation.

The bound provided in Lemma 4.3.3 can still be used more generally to control the size of the pseudo-variance of the stochastic gradient estimator given in (4.3.7). If we want to set the upper threshold of the pseudo-variance to be some fixed value

$V_0 > 0$, we need to ensure that

$$\frac{1}{n}\|\theta^{(t)} - \hat{\theta}\|^2 \left( \sum_{i=1}^{N} \frac{L_i^2}{p_i^t} \right) < V_0,$$

for all iterations $t = 1, \ldots, T$. We can rearrange the inequality above to obtain the following lower bound on the subsample size,

$$n > \frac{1}{V_0}\|\theta^{(t)} - \hat{\theta}\|^2 \left( \sum_{i=1}^{N} \frac{L_i^2}{p_i^t} \right). \tag{4.3.14}$$

For a given preferential subsampling scheme $\mathbf{p}^{(t)}$, we can control the noise of the stochastic gradient estimator given in (4.3.7) by choosing the subsample size $n \propto \|\theta^{(t)} - \hat{\theta}\|^2$. Our proposed algorithm is provided in Algorithm 5. The subsample size at iteration $t$, $n^{(t)}$, will be updated using the lower bound obtained in (4.3.14).

For a fixed noise threshold $V_0$, it will be possible to decrease or increase the size of $n^{(t)}$ depending on how far or close $\theta^{(t)}$ is to the posterior mode, $\hat{\theta}$. This means that the subsample size can be set adaptively according to the current state of the chain. The subsample size can be increased automatically in areas of the sample space where the gradient is harder to estimate. We note here that fixing the pseudo-variance of the gradient estimator to be constant may not always be the best thing to do, especially in areas of the sample space where the gradient is large. Further investigation on this point is left to future work.

This method is suitable for use on models that satisfy Assumption 2. Appendix B.2 provides a selection of examples where the Lipschitz constants can be calculated exactly.

## 4.4 Related work

The idea of using a non-uniform discrete distribution to draw subsamples and reweight a gradient estimator has been well-explored within the stochastic optimisation literature. Typically, the aim of these methods is to control the variance of the gradient estimator, in order to improve the speed of convergence of the algorithm.

**Algorithm 5:** Adaptive SGLD-CV with preferential subsampling (ASGLD-CV-PS)

1: Input: initialise $\theta^{(1)}$ close to $\hat{\theta}$, gradients $\nabla f_i(\hat{\theta})$, weights $\mathbf{p}^{(1)}$, step-size $\epsilon$, noise threshold $V_0$, Lipschitz constants $\{L_i\}_{i=1}^N$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Update $\mathbf{p}^{(t)}$.
4: Find smallest possible $n^{(t)}$ using (4.3.14).
5: Sample $n^{(t)}$ indices $\mathcal{S}^t$ according to $\mathbf{p}^{(t)}$ with replacement.
6: Calculate $\tilde{g}^{(t)}$ using (4.3.7)
7: Update parameters according to (4.2.4).
8: **end for**
9: **return** $\theta^{(T+1)}$

Various papers explore the use of a static or time-invariant subsampling schemes. Zhao and Zhang (2014b, 2015) and Kern and Gyorgy (2016) propose the use of an *importance sampling* approach for SGD-type algorithms, where the subsampling weights are chosen according to the Lipschitz smoothness constants of $N$ individual cost functions, i.e., $p_i = \frac{L_i}{\sum_{j=1}^N L_j}$. Zhao and Zhang (2014a) consider a *stratified sampling* approach, where data points are assigned the the same weight if they belong to the same strata or cluster. Zhang et al. (2017a) propose the use of determinantal point processes to diversify the subsamples selected for SGD, constructing a soft similarity measure to reweight data points.

Inspired by active learning methods, Salehi et al. (2017) create a multi-armed bandit (MAB) framework to dynamically update the subsampling weights over several iterations of the SGD algorithm. Feedback is collected via the most recent stochastic gradients and passed into the MAB at each iteration. Liu et al. (2020) adapts the work of Salehi et al. and extends it to the minibatch setting for the ADAM algorithm (Kingma and Ba, 2016).

There have only been a handful of papers considering similar ideas within the stochastic gradient MCMC literature. Fu and Zhang (2017) extend the stratified

sampling methodology of Zhao and Zhang (2014a) to the general class of SGMCMC algorithms. Li et al. (2021) meanwhile propose an exponentially weighted stochastic gradient method, which can be combined with other variance reduction techniques.

## 4.5 Numerical experiments

In the experiments to follow, we compare our proposed preferential subsampling approaches in a number of different scenarios. Our aim here is to demonstrate the value of preferential subsampling as a variance control measure. To communicate the idea succinctly, we only include the benchmark methods SGLD and SGLD-CV in our results.

We evaluate the performance of our proposed methods on both real and synthetic data. Please refer to Appendix B.2 for detailed information about the datasets considered.

A fixed step-size scheme for $\epsilon$ is used throughout, as suggested by Vollmer et al. (2016). To ensure a fair comparison, all samplers were run with the same step-size (with $\epsilon \approx \frac{1}{N}$). This allowed us to control for discretisation error and to independently assess the performance benefits offered by preferential subsampling. See Appendix B.4 for further details.

For samplers where the burn-in phase is replaced by an optimiser, we have opted to use an off-the-shelf implementation of ADAM (Kingma and Ba, 2016) to find the posterior mode. Unless stated otherwise, all samplers are implemented using sampling with replacement.

**Computing environment** We used the `jax` autograd module to implement the SGMCMC methods. Our results were obtained on a four-core 3.00GHz Intel Xeon(R) Gold virtual desktop. The code for this chapter is hosted on GitHub[4].

[4]Repository: `https://github.com/srshtiputcha/sgmcmc_preferential_subsampling`

### 4.5.1 Models

We compare sampler performance on the following three models: (i) bivariate Gaussian, (ii) binary logistic regression and (iii) linear regression. Full model details (including the derivation gradients and Lipschitz constants) are provided in Appendix B.2.

The examples have been deliberately chosen to be simple and our reasons for doing so are threefold. Firstly, our proposed methods rely upon being able to estimate the posterior mode well and as such, we are prioritising models where the mode is easy to find. Secondly, the Lipschitz constants for these models are known and this allows us to test our adaptive subsampling approach. And lastly, the SGLD-CV-PS subsampling scheme outlined in (4.3.11) requires the Hessian matrix, $\nabla^2 f_i(\cdot)$, to be computed for all data points and this is a costly preprocessing step for large parameter spaces[5].

#### Bivariate Gaussian

We simulate independent data from $X_i|\theta \sim \mathcal{N}_2(\theta, \Sigma_x)$ for $i = 1, \ldots, N$. It is assumed that $\theta$ is unknown and $\Sigma_x$ is known. The conjugate prior for $\theta$ is set to be $\theta \sim \mathcal{N}_2(\mu_0, \Lambda_0)$. The prior hyperparameters of the prior are $\mu_0 = (0, 0)^T$and $\Lambda_0 = \text{diag}(1 \times 10^3, 2)$. The target posterior is a non-isotropic Gaussian with negatively correlated parameters.

#### Binary logistic regression

Suppose we have data $x_1, \ldots, x_N$ of dimension $d$ taking values in $\mathbb{R}^d$, where each $x_i = (1, x_{i1}, \ldots, x_{ip})^T$ (where $d = p + 1$). Let us suppose that we also have the corresponding response variables $y_1, \ldots, y_N$ taking values in $\{0, 1\}$. Then a logistic regression model with parameters $\theta = (\beta_0, \beta_1, \ldots, \beta_p)$ will have the following density function

$$p(y_i|x_i, \theta) = \left(\frac{1}{1 + e^{-\theta^T x_i}}\right)^{y_i} \left(1 - \frac{1}{1 + e^{-\theta^T x_i}}\right)^{1-y_i}.$$

[5]We have provided an extended discussion of the preprocessing costs associated with SGLD-CV-PS in Appendix B.3.

The prior for $\theta$ is set to be $\theta \sim \mathcal{N}_d(\mu_0, \Lambda_0)$. The hyperparameters of the prior are $\mu_0 = (0, \ldots, 0)^T$ and $\Lambda_0 = \text{diag}(10, d)$.

**Linear regression**

Suppose we have data $x_1, \ldots, x_N$ of dimension $d$ taking values in $\mathbb{R}^d$, where each $x_i = (1, x_{i1}, \ldots, x_{ip})^T$ $(d = p + 1)$. Let us suppose that we also have the corresponding response variables $y_1, \ldots, y_N$ taking values on the real line.

We define the following linear regression model,

$$y_i = x_i^T \theta + \eta_i, \;\; \eta_i \sim \mathcal{N}(0, 1),$$

with parameters $\theta = (\beta_0, \beta_1, \ldots, \beta_p)$. The prior for $\theta$ is the same as above.

### 4.5.2 Metrics

We assess the performance of our samplers using the following metrics.

**Kullbeck-Leibler (KL) divergence**

The KL divergence is a measure of difference between two probability distributions with densities $p(\cdot)$ and $q(\cdot)$ and is given by,

$$D_{KL}(p||q) = \int p(\theta) \log \frac{p(\theta)}{q(\theta)} d\theta.$$

In the case of our bivariate Gaussian model, we know that the target posterior is conjugate and the KL divergence between two Gaussians can be written analytically. We use the KL divergence to measure the difference between the target posterior and our generated samples in Figure B.5.1(a).

**Log-loss**

The log-loss is a popular metric for assessing the predictive accuracy of the logistic regression model on a test dataset, $\mathcal{T}^*$. For binary classification, the log-loss is given by

$$l(\theta, \mathcal{T}^*) = -\frac{1}{|\mathcal{T}^*|} \sum_{i \in |\mathcal{T}^*|} \log p(y_i^* | x_i^*, \theta).$$

We compute the log-loss for our logistic regression example in Figure 4.5.2(b)(ii).

**Kernel Stein discrepancy**

We measure the sample quality of our MCMC chains using the kernel Stein discrepancy (KSD). The KSD assesses the discrepancy between the target posterior $\pi$ and the empirical distribution $\tilde{\pi}_K$ formed by SGMCMC samples $\{\theta\}_{k=1}^K$ (Liu et al., 2016; Gorham and Mackey, 2017). A key benefit of the KSD is that it penalises the bias present in our MCMC chains. We can define the KSD as,

$$KSD(\tilde{\pi}_K, \pi) = \sum_{j=1}^{d} \sqrt{\sum_{k,k'=1}^{K} \frac{k_j^0(\theta_k, \theta_{k'})}{K^2}}, \tag{4.5.1}$$

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

$$k_j^0(\theta, \theta') = \frac{1}{\pi(\theta)\pi(\theta')} \nabla_{\theta_j} \nabla_{\theta'_j} \big(\pi(\theta) \mathcal{K}(\theta, \theta') \pi(\theta')\big) \tag{4.5.2}$$

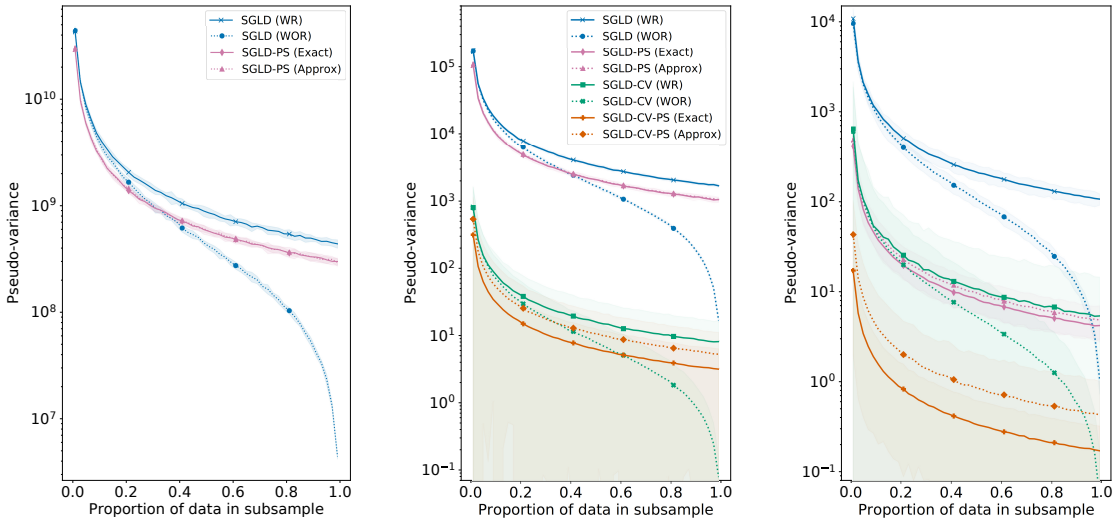
and $\mathcal{K}(\cdot, \cdot)$ is a valid kernel function. Gorham and Mackey (2017) recommend using the inverse multi-quadratic kernel, $\mathcal{K}(\theta, \theta') = \big(c^2 + \|\theta - \theta'\|_2^2\big)^{\beta}$, which detects non-convergence for $c > 0$ and $\beta \in (-1, 0)$. In practice, the full-data gradients in (4.5.2) can be replaced by noisy, unbiased estimates. We compute the KSD for our linear and logistic regression examples in Figures 4.5.2(a) - (b)(i), 4.5.3(a), B.5.1(b) and B.5.2(a).

### 4.5.3 Numerical results

**Evaluating the quality of the stochastic gradients**

In this experiment, our objective was to compare the pseudo-variance of our proposed gradient estimators against the proportion of data used in a subsample, $\frac{n}{N}$. We consider three scenarios: (a) bivariate Gaussian, (b) balanced binary logistic regression, and (c) imbalanced binary logistic regression. For ease of computation, a synthetic dataset of size $N = 10^3$ was used for all models.

In each scenario, we generated ten candidate draws of $\theta$ and calculated an empirical estimate of the pseudo-variance at each. The candidate draws were sampled either from the posterior (for scenario (a)) or from a normal approximation to the posterior




(a) (b) (c)

Figure 4.5.1: Empirical pseudo-variance against proportion of data in a subsample, $\frac{n}{N}$. (a) bivariate Gaussian, (b) balanced logistic regression, (c) imbalanced logistic regression.

(for scenarios (b) and (c)). We plot the mean empirical estimate of the pseudo-variance for various subsample sizes.

Figure 4.5.1(a) compares the stochastic gradients for SGLD with and without replacement (WR and WOR respectively) and for SGLD-PS with exact and approximate subsampling schemes. Figures 4.5.1(b) and (c) additionally compare the gradient estimators of SGLD-CV and those of SGLD-CV-PS with exact and approximate subsampling schemes.[6]

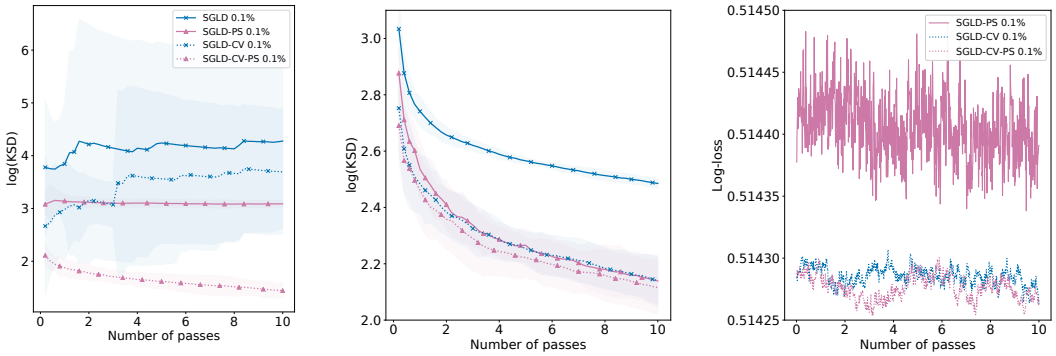
SGLD and SGLD-CV without replacement offer the best variance reduction for larger subsamples, as $\frac{n}{N}$ tends towards 1. For more reasonable subsample sizes of $n \leq 0.2N$, however, there is no major benefit in generating subsamples without replacement. Figure 4.5.1 illustrates that there is a marked reduction in the pseudo-variance when a preferential subsampling scheme is used.

SGLD-PS and SGLD-CV-PS consistently outperform their vanilla counterparts

[6]In the case of a Gaussian posterior, the SGLD-CV stochastic gradient offers optimal variance reduction, with optimal weights $p_i = \frac{1}{N}$. For this reason, there is no extra improvement gain to be obtained here by implementing SGLD-CV-PS

and there seems to be very little difference between the exact and approximate schemes for SGLD-PS. Whereas, there is a difference in performance between the exact and approximate subsampling schemes for SGLD-CV-PS. This difference is noticeable in Figure 4.5.1(c) for the synthetic imbalanced logistic regression data. Practically, it is not feasible to use the exact subsampling weights, but as illustrated here, using approximate preferential weights is always better than using uniform weights.

**Performance of fixed subsample size methods**




(a) (b)(i) (b)(ii)

Figure 4.5.2: Sampler performance of SGLD, SGLD-CV, SGLD-PS and SGLD-CV-PS for 0.1% subsample size over 10 passes through the data. (a) linear regression model on the CASP data (y-axis: KSD); (b) logistic regression on the covertype data (y-axis: (i) KSD, (ii) log-loss).

In Figure 4.5.2, we compare the sampler performance of SGLD, SGLD-CV, SGLD-PS and SGLD-CV-PS for a subsample size of 0.1% of the dataset size over 10 passes of the data. We have run ten MCMC chains allowing for an equal number of iterations for both burn-in and sampling.

Figure 4.5.2(a) plots the KSD results for the linear regression model fitted on the CASP dataset. The CASP dataset has been obtained from the UCI Machine Learning repository and contains 45,730 instances and 9 features.

Furthermore, Figure 4.5.2(b) plots the results of fitting the logistic regression model to the covertype dataset (Blackard and Dean, 1998). The covertype dataset contains

581,012 instances and 54 features. The KSD results are shown in Figure 4.5.2(b)(i) and the log-loss (evaluated every 10 iterations) is computed on the test set in Figure 4.5.2(b)(ii).

In addition, we separately compare the sampler performance of SGLD and SGLD-PS in Figure B.5.1 (see Appendix B.5) for subsample sizes of 1%, 5% and 10% of the dataset size, over 500 passes of the data. As before, ten MCMC chains were run for each subsample size tested, allowing for an equal number of iterations for both burn-in and sampling

Figure B.5.1(a) plots the KL divergence for the bivariate Gaussian model fitted on synthetic data of size $N = 10^4$. Figure B.5.1(b) plots the KSD for the logistic regression model fitted on the covertype dataset (Blackard and Dean, 1998).

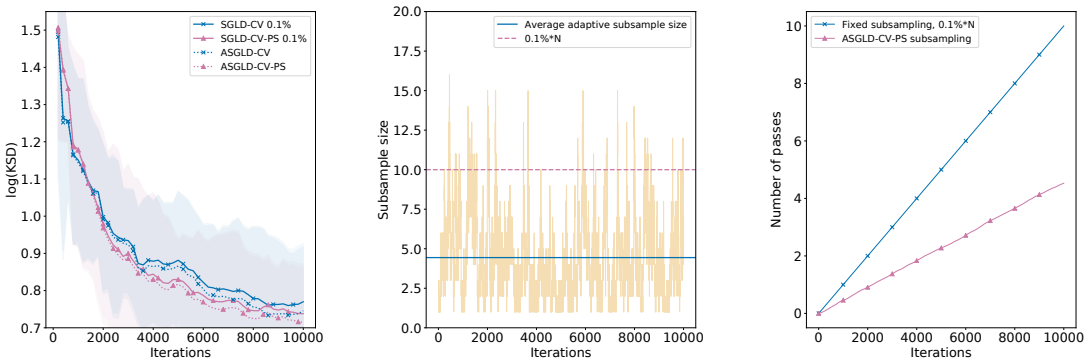
We can see that SGLD-CV-PS exhibits the best performance overall. More generally, we find that there is a benefit in implementing preferential subsampling for vanilla SGLD as well. In practice, we find that the largest performance gains are found for preferential subsampling when the subsampling size is reasonably small.

**Performance of adaptive subsampling**

We are interested in assessing the sampler performance of ASGLD-CV and ASGLD-CV-PS over $10^4$ iterations. In this experiment, we considered two scenarios: (i) the logistic regression model on balanced synthetic data of size $N = 10^4$; and (ii) the linear regression model on the CASP data. The results for scenario (i) are plotted in Figure 4.5.3. Please refer to Figure B.5.2 in Appendix B.5 for the results of scenario (ii). Both models satisfy Assumption 2.

In order to implement the adaptive subsampling methods, we had to first pick a suitable pseudo-variance threshold, $V_0$. We generated ten chains of SGLD-CV and SGLD-CV-PS for a fixed subsample of size 0.1% of the dataset size. For each chain, we then:

1. calculated the squared Euclidean distances between the mode and the samples, $\|\theta - \hat{\theta}\|^2$,




(a) (b) (c)

Figure 4.5.3: A logistic regression model fitted on balanced synthetic data of size $N = 10^4$. (a) KSD comparison of SGLD-CV, SGLD-CV-PS, ASGLD-CV and ASGLD-CV-PS over $10^4$ iterations; (b) adaptive subsample sizes selected along one ASGLD-CV-PS chain; (c) the number of passes through the data considered by fixed versus adaptive subsampling.

2. found the 95-th percentile of the array of squared distances, and

3. calculated a proposal for $V_0$ using the bound in (4.3.13).

We set $V_0$ to be the largest proposal amongst the ten chains.

Figure 4.5.3(a) plots the KSD for all four methods and Figure 4.5.3(b) displays the adaptive subsample sizes selected along one chain of ASGLD-CV-PS. Figure 4.5.3(c) compares the number of passes through the data considered by fixed subsampling versus ASGLD-CV-PS over $10^4$ iterations.

Overall, we see that the performance of ASGLD-CV-PS is somewhat better than that of ASGLD-CV. We cannot always presume that the adaptive subsampling methods will outperform their fixed subsampling counterparts (see Figure B.5.2(a) for instance) in terms of sample quality. However, it is clear that the adaptive subsampling methods successfully process far less of the data over $10^4$ iterations with no significant reduction in statistical accuracy.

## 4.6 Conclusions

We have used preferential subsampling to reduce the variance of the stochastic gradient estimator for both SGLD and SGLD-CV. In addition, we have extended SGLD-CV to allow for adaptive subsampling.

We have empirically studied the impact of preferential subsampling on a range of synthetic and real-world datasets. Our numerical experiments successfully demonstrate the performance improvement from both preferential subsampling and adaptively selecting the subsample size.

Future work in this area could explore the potential for using multi-armed bandits to preferentially select data subsamples for SGMCMC. These concepts have only been previously considered within the context of stochastic optimisation (Salehi et al., 2017; Liu et al., 2020).

The methods outlined in this chapter are not limited to Langevin dynamics and can be applied to other SGMCMC samplers. Furthermore, it would be worth considering how preferential subsampling could be extended for other variance control methods, such as SAGA-LD or SVRG-LD (Dubey et al., 2016). This could potentially be done by adapting the ideas presented in Schmidt et al. (2015), Kern and Gyorgy (2016) and Schmidt et al. (2017). Arguments that mirror those presented in Lemmas 4.3.1 and 4.3.2 could be used to obtain the optimal preferential subsampling scheme in each new case.

## 4.7 Results from Section 4.3

### 4.7.1 Full derivation of the pseudo-variance

The pseudo-variance $\tilde{g}^{(t)}$ is given by:

$$\mathbb{V}\big(\tilde{g}^{(t)}\big) = \mathbb{E}\left(\big\|\tilde{g}^{(t)} - g^{(t)}\big\|^2\right) \tag{4.7.1}$$

$$= \mathbb{E}\left(\big(\tilde{g}^{(t)} - g^{(t)}\big)^T\big(\tilde{g}^{(t)} - g^{(t)}\big)\right) \tag{4.7.2}$$

$$= \sum_{j=1}^{d} \mathbb{E}\left(\big(\tilde{g}_j^{(t)} - g_j^{(t)}\big)^2\right) \qquad \text{(decompose expectation over } d \text{ parameters)} \tag{4.7.3}$$

$$= \sum_{j=1}^{d} \mathbb{E}\left(\big(\tilde{g}_j^{(t)} - \mathbb{E}\left[\,\tilde{g}_j^{(t)}\right]\big)^2\right) \tag{4.7.4}$$

$$= \sum_{j=1}^{d} \mathrm{Var}\left(\tilde{g}_j^{(t)}\right) \tag{4.7.5}$$

$$= \mathrm{tr}\left(\,\mathrm{Cov}\left(\tilde{g}^{(t)}\right)\right). \tag{4.7.6}$$

### 4.7.2 Proof of Lemma 4.3.1

*Proof.* From Section 4.7.1, we know that

$$\mathbb{V}\big(\tilde{g}^{(t)}\big) = \mathbb{E}\left[\big\|\tilde{g}^{(t)} - g^{(t)}\big\|^2\right] = \sum_{j=1}^{d} \mathrm{Var}\left(\tilde{g}_j^{(t)}\right). \tag{4.7.7}$$

Taking expectations with respect to $\mathbf{p}^{(t)}$, we know that the $j$-th component of the sum in (4.7.7) is given by:

$$\mathrm{Var}\left(\tilde{g}_j^{(t)}\right) = \mathrm{Var}\left(\nabla_j f_0\big(\theta^{(t)}\big) + \frac{1}{n}\sum_{i\in\mathcal{S}^t}\frac{1}{p_i^t}\nabla_j f_i\big(\theta^{(t)}\big)\right) = \mathrm{Var}\left(\frac{1}{n}\sum_{i\in\mathcal{S}^t}\frac{1}{p_i^t}\nabla_j f_i\big(\theta^{(t)}\big)\right).$$

Given that all stochastic gradient estimators are unbiased with the same mean, we know that minimising $\mathbb{V}\big(\tilde{g}^{(t)}\big)$ with respect to $\mathbf{p}^{(t)}$ is equivalent to minimising the component-wise sum of second moments, $\sum_{j=1}^{d}\mathbb{E}\left(\left(\frac{1}{n}\sum_{i\in\mathcal{S}^t}\frac{1}{p_i^t}\nabla_j f_i\big(\theta^{(t)}\big)\right)^2\right)$.

So, for $j = 1, \ldots, d$, we consider

$$\begin{aligned}
\mathbb{E}\left(\left(\frac{1}{n}\sum_{i\in\mathcal{S}^t}\frac{1}{p_i^t}\nabla_j f_i\big(\theta^{(t)}\big)\right)^2\right) &= \frac{1}{n^2}\,\mathbb{E}\left(\sum_{i\in\mathcal{S}^t}\left(\frac{1}{p_i^t}\nabla_j f_i\big(\theta^{(t)}\big)\right)^2\right.\\
&\qquad\left.+\sum_{i\in\mathcal{S}^t}\sum_{k\in\mathcal{S}^t,\,i\neq k}\frac{1}{p_i^t}\cdot\frac{1}{p_k^t}\cdot\nabla_j f_i\big(\theta^{(t)}\big)\cdot\nabla_j f_k\big(\theta^{(t)}\big)\right)\\
&= \frac{1}{n}\,\mathbb{E}\left(\left(\frac{1}{p_i^t}\nabla_j f_i\big(\theta^{(t)}\big)\right)^2\right)+\frac{n(n-1)}{n^2}\mathbb{E}\left(\frac{1}{p_i^t}\nabla_j f_i\big(\theta^{(t)}\big)\right)^2\\
&= \frac{1}{n}\sum_{i=1}^{N}\frac{1}{p_i^t}\big[\nabla_j f_i\big(\theta^{(t)}\big)\big]^2+\frac{n-1}{n}\left(\sum_{i=1}^{N}\nabla_j f_i\big(\theta^{(t)}\big)\right)^2\\
&= \frac{1}{n}\sum_{i=1}^{N}\frac{1}{p_i^t}\big[\nabla_j f_i\big(\theta^{(t)}\big)\big]^2+C_j,
\end{aligned}$$

where $C_j$ is a constant that does not depend on $\mathbf{p}^{(t)}$. Adding up over all components, we see that,

$$\begin{aligned}
\sum_{j=1}^{d}\mathbb{E}\left(\left(\frac{1}{n}\sum_{i\in\mathcal{S}^t}\frac{1}{p_i^t}\nabla_j f_i\big(\theta^{(t)}\big)\right)^2\right) &= \sum_{j=1}^{d}\left[\frac{1}{n}\sum_{i=1}^{N}\frac{1}{p_i^t}\big[\nabla_j f_i\big(\theta^{(t)}\big)\big]^2+C_j\right]\\
&= \frac{1}{n}\sum_{i=1}^{N}\frac{1}{p_i^t}\|\nabla f_i\big(\theta^{(t)}\big)\|^2+C'.
\end{aligned}$$

Therefore,

$$\min_{\mathbf{p}^{(t)},\,p_i^t\in[0,1],\sum_i p_i^t=1}\mathbb{V}\big(\tilde{g}^{(t)}\big)\iff\min_{\mathbf{p}^{(t)},\,p_i^t\in[0,1],\sum_i p_i^t=1}\frac{1}{n}\sum_{i=1}^{N}\frac{1}{p_i^t}\big\|\nabla f_i\big(\theta^{(t)}\big)\big\|^2.$$

The minimisation problem of interest is

$$\min_{\mathbf{p}^{(t)},\,p_i^t\in[0,1],\sum_i p_i^t=1}\frac{1}{n}\sum_{i=1}^{N}\frac{1}{p_i^t}\big\|\nabla f_i\big(\theta^{(t)}\big)\big\|^2.$$

We know that via the Cauchy-Schwarz inequality[7],

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{N}\frac{1}{p_i^t}\left\|\nabla f_i\left(\theta^{(t)}\right)\right\|^2 &= \frac{1}{n}\left(\sum_{i=1}^{N}\frac{1}{p_i^t}\left\|\nabla f_i\left(\theta^{(t)}\right)\right\|^2\right)\left(\sum_{i=1}^{N}p_i^t\right)\\ &\geq \frac{1}{n}\left(\sum_{i=1}^{N}\sqrt{\frac{1}{p_i^t}\left\|\nabla f_i\left(\theta^{(t)}\right)\right\|^2 p_i^t}\right)^2\\ &= \frac{1}{n}\left(\sum_{i=1}^{N}\left\|\nabla f_i\left(\theta^{(t)}\right)\right\|\right)^2.\end{aligned}$$

The equality is only obtained when there exists a constant $c \in \mathbb{R}$ such that

$$\begin{pmatrix}(p_1^t)^{-1}\left\|\nabla f_1\left(\theta^{(t)}\right)\right\|^2\\(p_2^t)^{-1}\left\|\nabla f_2\left(\theta^{(t)}\right)\right\|^2\\\vdots\\(p_N^t)^{-1}\left\|\nabla f_N\left(\theta^{(t)}\right)\right\|^2\end{pmatrix} = c\begin{pmatrix}p_1^t\\p_2^t\\\vdots\\p_N^t\end{pmatrix},$$

which is equivalent to writing

$$(p_1^t)^{-2}\left\|\nabla f_1\left(\theta^{(t)}\right)\right\|^2 = (p_2^t)^{-2}\left\|\nabla f_2\left(\theta^{(t)}\right)\right\|^2 = \ldots = (p_N^t)^{-2}\left\|\nabla f_N\left(\theta^{(t)}\right)\right\|^2.$$

Under this constraint, Problem (4.3.8) is minimised. We can therefore conclude that the optimal weights which minimise the pseudo-variance are,

$$p_i^t = \frac{\left\|\nabla f_i\left(\theta^{(t)}\right)\right\|}{\sum_{k=1}^{N}\left\|\nabla f_k\left(\theta^{(t)}\right)\right\|} \quad \text{for } i = 1, \ldots, N.$$

□

### 4.7.3 Proof of Lemma 4.3.2

*Proof.* By the similar argument to that used for Lemma 4.3.1, we can show that

$$\min_{\mathbf{p}^{(t)},\, p_i^t \in [0,1], \sum_i p_i^t = 1} \mathbb{V}\left(\tilde{g}^{(t)}\right) \iff \min_{\mathbf{p}^{(t)},\, p_i^t \in [0,1], \sum_i p_i^t = 1} \frac{1}{n}\sum_{i=1}^{N}\frac{1}{p_i^t}\left\|\nabla f_i\left(\theta^{(t)}\right) - \nabla f_i(\hat{\theta})\right\|^2.$$

[7]The Cauchy-Schwarz inequality states that for $d$-dimensional real vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, of all inner product space it is true that

$$|\langle \mathbf{u}, \mathbf{v}\rangle| \leq \langle \mathbf{u}, \mathbf{u}\rangle \cdot \langle \mathbf{v}, \mathbf{v}\rangle$$

Furthermore, the equality holds only when either $\mathbf{u}$ or $\mathbf{v}$ is a multiple of the other.

Once again, using the Cauchy-Schwarz inequality allows us to see that the optimal weights which minimise the pseudo-variance are,

$$p_i^t = \frac{\|\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta}))\|}{\sum_{k=1}^N \|\nabla f_k\big(\theta^{(t)}\big) - \nabla f_k(\hat{\theta})\|} \quad \text{for } i = 1, \ldots, N.$$

□

### 4.7.4 Deriving approximate weights for the control variates gradient

The minimisation problem of interest is

$$\min_{\mathbf{p}^{(t)},\, p_i^t \in [0,1], \sum_i p_i^t = 1} \mathbb{E}_\theta\left[\frac{1}{n}\sum_{i=1}^N \frac{1}{p_i^t}\big\|\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta})\big\|^2\right].$$

So,

$$\begin{aligned}
\mathbb{E}_\theta\left[\frac{1}{n}\sum_{i=1}^N \frac{1}{p_i^t}\big\|\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta})\big\|^2\right] &= \frac{1}{n}\sum_{i=1}^N \frac{1}{p_i^t}\mathbb{E}_\theta\left[\big\|\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta})\big\|^2\right] \quad \text{(linearity of } \mathbb{E}_\theta\text{)} \\
&= \left(\frac{1}{n}\right)\left(\sum_{i=1}^N \frac{1}{p_i^t}\mathbb{E}_\theta\left[\big\|\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta})\big\|^2\right]\right)\left(\sum_{i=1}^N p_i^t\right) \\
\text{(Cauchy-Schwarz)} \quad &\geq \frac{1}{n}\left(\sum_{i=1}^N \sqrt{\frac{1}{p_i^t}\,\mathbb{E}_\theta\left[\big\|\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta})\big\|^2\right] p_i^t}\right) \\
&= \frac{1}{n}\left(\sum_{i=1}^N \sqrt{\mathbb{E}_\theta\left[\big\|\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta})\big\|^2\right]}\ \right)^2.
\end{aligned}$$

The problem is minimised when

$$p_i^t \propto \sqrt{\mathbb{E}_\theta\left[\big\|\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta})\big\|^2\right]} \text{ for } i = 1, \ldots, N.$$

Let's assume that $\theta \dot{\sim} \mathcal{N}(\hat{\theta}, \hat{\Sigma})$ at stationarity, where $\hat{\Sigma} = -H(\hat{\theta})^{-1}$. Using a first-order Taylor expansion about $\hat{\theta}$,

$$\big\|\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta})\big\|^2 \approx \|\nabla f_i(\hat{\theta}) + \nabla^2 f_i(\hat{\theta})(\theta^{(t)} - \hat{\theta}) - \nabla f_i(\hat{\theta})\big\|^2 = \big\|\nabla^2 f_i(\hat{\theta})(\theta^{(t)} - \hat{\theta})\big\|^2.$$

We know that $(\theta - \hat{\theta}) \dot{\sim} \mathcal{N}(\mathbf{0}, \hat{\Sigma})$. So,

$$\nabla^2 f_i(\hat{\theta})(\theta^{(t)} - \hat{\theta}) \dot{\sim} \mathcal{N}\big(\mathbf{0}, \nabla^2 f_i(\hat{\theta})\hat{\Sigma}\nabla^2 f_i(\hat{\theta})^T\big).$$

Then,

$$\begin{aligned}\mathbb{E}_\theta\left[\left\|\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta})\right\|^2\right] &\approx \mathbb{E}_\theta\left[\left\|\nabla^2 f_i(\hat{\theta})(\theta^{(t)} - \hat{\theta})\right\|^2\right] \\ &= \operatorname{tr}\left(\operatorname{Cov}\left(\nabla^2 f_i(\hat{\theta})\big(\theta^{(t)} - \hat{\theta}\big)\right)\right) \\ &\approx \operatorname{tr}\left(\nabla^2 f_i(\hat{\theta})\,\hat{\Sigma}\,\nabla^2 f_i(\hat{\theta})^T\right).\end{aligned}$$

So, we can set

$$p_i \propto \sqrt{\operatorname{tr}\left(\nabla^2 f_i(\hat{\theta})\,\hat{\Sigma}\,\nabla^2 f_i(\hat{\theta})^T\right)} \text{ for } i = 1, \ldots, N.$$

### 4.7.5 Proof of Lemma 4.3.3

*Proof.* We know that the pseudo-variance can be decomposed into

$$\begin{aligned}\mathbb{V}(\tilde{g}) &= \mathbb{E}\left[\|\tilde{g} - g\|^2\right] \\ &= \frac{1}{n}\sum_{i=1}^{N}\frac{1}{p_i^t}\|\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta})\|^2 - \underbrace{\frac{1}{n}\left\|\sum_{i=1}^{N}\left[\nabla_j f_i\big(\theta^{(t)}\big) - \nabla_j f_i(\hat{\theta})\right]\right\|^2}_{\geq 0} \\ &\leq \frac{1}{n}\sum_{i=1}^{N}\frac{1}{p_i^t}\|\nabla f_i\big(\theta^{(t)}\big) - \nabla f_i(\hat{\theta})\|^2.\end{aligned}$$

Under Assumption 2, we know that

$$\mathbb{V}(\tilde{g}) \leq \frac{1}{n}\|\theta^{(t)} - \hat{\theta}\|^2\left(\sum_{i=1}^{N}\frac{L_i^2}{p_i^t}\right).$$

□

# Chapter 5

# A Feasibility Study: Utilising Determinantal Point Processes for Subsampling

## 5.1 Overview

The goal of this chapter is evaluate the feasibility of using determinantal point processes (DPPs) to construct informed, low-variance data subsamples for SGLD. The DPP defines a similarity measure between datapoints and assigns a larger probability to subsets of datapoints that are more diverse. By modelling the repulsive correlations between items, a random draw from a DPP should provide a balanced representation of the underlying data. DPPs can be used to automate many real-world tasks, including text summarisation and generating high-quality search results (Kulesza et al., 2012).

The ideas presented in this chapter are motivated by the mini-batch diversification approach proposed in Zhang et al. (2017a) and Zhang et al. (2017b) for stochastic gradient descent. These works provide some empirical evidence that minibatches drawn from DPPs (and other repulsive point processes) can yield gradient estimators for SGD with a lower variance. Given that these ideas have been applied successfully in the optimisation setting, our aim is to assess how they could be applied in the sampling setting.

## 5.2 Determinantal point processes

A DPP is a distribution over subsets of a fixed ground set of $N$ items. The defining characteristic of a DPP is the notion of negative correlation. That is, the inclusion of one item in a subset makes the inclusion of other similar items less likely. A measure of similarity between any pair of items in the ground set can be obtained via a suitable kernel matrix. By construction, the DPP will assign a higher probability value to subsets that are more diverse.

A point process $\mathcal{P}$ over a discrete set $\mathcal{Y} = \{1, \ldots, N\}$ is a probability measure on $2^{\mathcal{Y}}$. Specifically, $\mathcal{P}$ is a determinantal point process if, when $\mathbf{Y}$ is a random subset drawn according to $\mathcal{P}$, we have that for every $A \subseteq \mathcal{Y}$,

$$\mathcal{P}(A \subseteq \mathbf{Y}) = \det(K_A). \tag{5.2.1}$$

Here, $K$ is a real, symmetric $N \times N$ indexed by the items in $\mathcal{Y}$ and $K_A \equiv [K_{ij}]_{i,j \in A}$.[1] Kulesza et al. (2012) state that normalisation is unnecessary at this stage, given that we are inherently defining marginal probabilities that need not sum to 1. We refer to $K$ as the marginal kernel, as it allows us to define the marginal probability of any subset $A$. Since $\mathcal{P}$ is a probability measure, all principal minors of the matrix $K$ must be nonnegative, and therefore $K$ must be positive semi-definite (psd).

If $A = \{i\}$, then by Eq. (5.2.1) we have,

$$\mathcal{P}(i \in \mathbf{Y}) = K_{ii}. \tag{5.2.2}$$

In other words, the diagonal elements of the marginal kernel gives us the marginal probabilities of selecting an individual item from $\mathcal{Y}$. Similarly if $A = \{i, j\}$, we know that,

$$\begin{aligned} \mathcal{P}(i, j \in \mathbf{Y}) &= \begin{vmatrix} K_{ii} & K_{ij} \\ K_{ji} & K_{jj} \end{vmatrix} \\ &= K_{ii}K_{jj} - K_{ij}K_{ji} \\ &= \mathcal{P}(i \in \mathbf{Y})\mathcal{P}(j \in \mathbf{Y}) - K_{ij}^2. \end{aligned} \tag{5.2.3}$$

[1] We note that $\det(K_\emptyset) = 1$.

Eq. (5.2.3) demonstrates that the off-diagonal elements of $K$ describe the negative correlation between pairs of elements in $\mathcal{Y}$. For a given pair $i, j \in \mathcal{Y}$, a larger value of $K_{ij}$ would imply that $i$ and $j$ are very similar and should therefore not be chosen together.

### $L$-ensembles

To fully characterise a DPP of the form of Eq. (5.2.1), the eigenvalues of the marginal kernel $K$ must be bounded above by 1. In practice, it is therefore more convenient (especially when modelling real data) to construct DPPs using an alternative framework. An $L$-ensemble specifies a DPP in terms of the atomic probabilities for every possible instantiation of $\mathbf{Y}$ (Borodin and Rains, 2005; Kulesza et al., 2012). This is done via a real, symmetric matrix $L$, indexed by the items in $\mathcal{Y}$, such that,

$$\mathcal{P}_L(\mathbf{Y} = Y) \propto \det(L_Y). \tag{5.2.4}$$

As above, we once again require $L$ to be real, symmetric and psd. The normalisation constant of Eq. (5.2.4) can be given in closed form, since $\sum_{Y \subseteq \mathcal{Y}} \det(L_Y) = \det(L + I)$. It is also possible to recover the marginal kernel of an $L$-ensemble using the relation: $K = (L + I)^{-1} L$ (Macchi, 1975).

Let $\lambda_1, \lambda_2, \ldots, \lambda_N$ be the eigenvalues of $L$. It can be shown that the cardinality of a random subset $\mathbf{Y} \subseteq \mathcal{Y}$, $|\mathbf{Y}|$, is distributed as the number of successes observed amongst $N$ Bernoulli trials, where trial $i$ succeeds with probability $\frac{\lambda_i}{\lambda_i + 1}$. One immediate consequence of this is that $|\mathbf{Y}|$ cannot be larger than $\text{rank}(L)$.

### $k$-DPPs

A $k$-DPP on a discrete set $\mathcal{Y} = \{1, \ldots, N\}$ is a distribution over all subsets $\mathbf{Y} \subseteq \mathcal{Y}$ with cardinality $k$ (Kulesza and Taskar, 2011; Kulesza et al., 2012). Specifically, we can obtain a $k$-DPP by conditioning a standard DPP on the event that a random subset $\mathbf{Y}$ has size $k$. The $k$-DPP $\mathcal{P}_L^k$ gives probabilities,

$$\mathcal{P}_L^k(Y) = \frac{\det(L_Y)}{\sum_{|Y'|=k} \det(L'_Y)}, \tag{5.2.5}$$

where $|Y| = k$ and $L$ is a psd kernel. Apart from conditioning on the cardinality of the subset, the $k$-DPP incorporates the same diversification effect as the standard DPP.

Figure 5.2.1 provides a visual comparison of the difference between a uniform random sample and DPP sample. The left panel shows $N = 1,000$ datapoints drawn from a bivariate normal distribution with mild correlation. The middle panel plots a uniform random sample of size $k = 60$ from the original data. For comparison, the right panel plots a $k$-DPP subsample of size $k = 60$ from the original data. The $k$-DPP was constructed using a Gaussian kernel of Euclidean distances. In this example, the $k$-DPP scheme best restores the shape of the original data set, introducing more repulsion between the subsampled points.

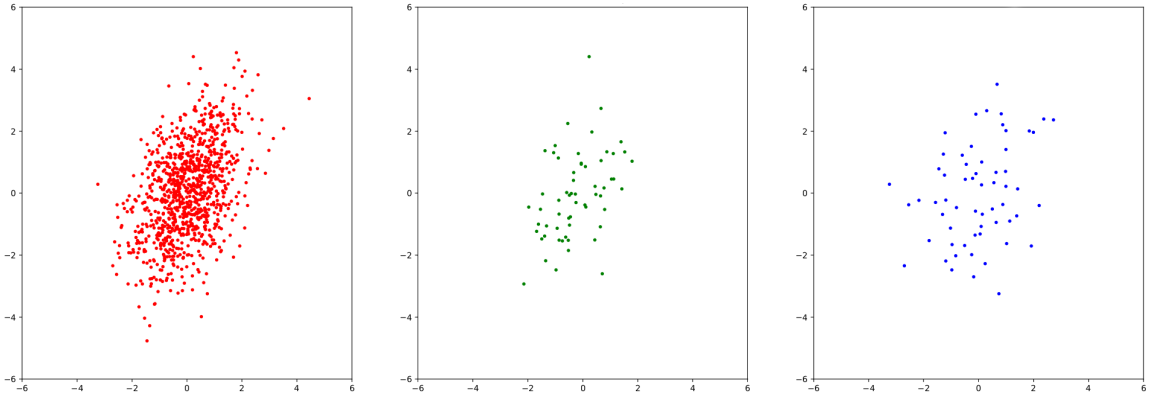


Figure 5.2.1: Sampling comparison - (left) bivariate Normal, (centre) uniform sample, (right) $k$-DPP sample.

### Sampling from a $k$-DPP

Kulesza et al. (2012) propose an exact spectral algorithm to sample from $k$-DPPs (and more generally finite DPPs). Broadly speaking, there are three stages involved in sampling from a $k$-DPP:

(i) Compute the elementary symmetric polynomials of $L$;

(ii) Sample $k$ eigvenvectors $V$ by using the elementary symmetric polynomials; and,

(iii) Generate a sample $Y$ using the selected eigenvectors $V$.

Drawing a subset $Y$ of size $k$ from the ground set has computational complexity of $O(Nk^3)$. Algorithm 6 outlines the full sampling procedure. The input to the algorithm is an eigendecomposition of the kernel matrix $L$, the subset size $k$, and number of samples to be drawn $T$. We note that $\mathbf{e}_i$ is the $i$-th standard basis $N$-vector, which is all zeros except for a one in the $i$-th position.

The eigendecomposition of the matrix $L$ can be very costly to compute for large datasets. To circumvent this issue, various $k$-DPP sampling approaches (both exact and approximate) have been outlined in the DPP literature. Launay et al. (2020) propose an exact sampling algorithm which relies on a Cholesky decomposition of the kernel matrix and a sequential thinning procedure. For certain applications, this procedure can be faster than the original spectral algorithm. Calandriello et al. (2020) propose an efficient method based on the concept of a distortion-free intermediate sample, where a larger sample of points is drawn in such a way that we can then downsample to the correct DPP distribution. Affandi et al. (2013) propose the use of a Nystrom approximation to project the kernel matrix into a low-dimensional space and the dual representation of the approximated kernel is applied to reduce sampling complexity. MCMC-based $k$-DPP samplers (Li et al., 2016a; Anari et al., 2016) offer a promising alternative to the spectral algorithm (with convergence guarantees) provided the MCMC chain mixes fast enough. We refer the reader to the `DPPy` Python package documentation (Gautier et al., 2019) for a more comprehensive list of the available options.

## 5.3 Proposed approach

In this section, we propose and subsequently evaluate a potential approach for variance control in SGLD, where $k$-DPPs are used to draw subsamples and reweight the stochastic gradient. The ideas presented below have been adapted from the work conducted by Zhang et al. (2017a) and Zhang et al. (2017b) for minibatch stochastic gradient descent. Modifying vanilla SGLD to allow for a DPP-based subsampling process necessitates a separate preprocessing phase.

**Algorithm 6:** $k$-DPP sampling

Input: subset size $k$; eigendecomposition of $L$,$\{\lambda_m, \mathbf{v}_m\}_{m=1}^N$, number of draws $T$
*Stage 1: Compute the elementary symmetric polynomials of $L$.*
$e_0^m \leftarrow 1$ for all $m \in \{0, 1, 2, \ldots, N\}$
$e_l^0 \leftarrow 1$ for all $l \in \{0, 1, 2, \ldots, k\}$
**for** $l = 1, 2, \ldots, k$ **do**
  **for** $m = 1, 2, \ldots, N$ **do**
    $e_l^m \leftarrow e_l^{m-1} + \lambda_j e_{l-1}^{m-1}$
  **end for**
**end for**
**for** $t = 1, \ldots, T$ **do**
  *Stage 2: Sample $k$ eigenvectors $V$ with indices $J$.*
  $J \leftarrow \emptyset$
  $l \leftarrow k$
  **for** $m = N, \ldots, 2, 1$ **do**
    **if** $l = 0$ **then**
      **break**
    **end if**
    **if** $u \sim U[0, 1] < \lambda_m \frac{e_{l-1}^{m-1}}{e_l^m}$ **then**
      $J \leftarrow J \cup \{m\}$
      $l \leftarrow l - 1$
    **end if**
  **end for**
  *Stage 3: Sample $k$ points indexed by $Y$ using $V$.*
  $V \leftarrow \{\mathbf{v}_m\}_{m \in J}$
  $Y \leftarrow \emptyset$
  **while** $|V| > 0$ **do**
    Select $i$ with probability $\Pr(i) = \sum_{\mathbf{v} \in V} (\mathbf{v}^T \mathbf{e}_i)^2$
    $Y \leftarrow Y \cup \{i\}$
    $V \leftarrow V_\perp$, an orthonormal basis for the subspace of $V$ orthogonal to $\mathbf{e}_i$
  **end while**
**end for**
**return** samples $Y_1, Y_2, \ldots, Y_T$

### 5.3.1 Preprocessing

There are three preprocessing steps that would need to be implemented ahead of running SGLD.

(i) Constructing a suitable $k$-DPP kernel, $L$. This is application specific and the kernel matrix should be an $N \times N$ symmetric, psd matrix with large rank.

(ii) Pre-selecting data subsamples using Algorithm 6. This process is independent of the model parameters and could even be parallelised. Depending on the method used, this can be a fairly lengthy involved computational task.

(iii) Computing a vector of marginal probabilities $\mathbf{p} = (p_1, p_2, \ldots, p_N)^T$ for each subsample, where $p_i = \frac{L_{ii}}{\text{trace}(L)}$ for all $i$. This can be seen from leveraging Eq. (5.2.5) for $k = 1$.

### 5.3.2 DPP-SGLD

The unbiased gradient estimator is given by reweighting the simple estimator defined in (4.2.3) (Welling and Teh, 2011),

$$\tilde{g}^{(t)} = \nabla f_0\big(\theta^{(t)}\big) + \frac{1}{k}\sum_{i \in \mathcal{S}^t} \frac{1}{p_i} \nabla f_i\big(\theta^{(t)}\big), \tag{5.3.1}$$

where $S^t \sim k\text{-DPP}(L)$ and $|S^t| = k$ ($k \ll N$). The pseudocode for DPP-SGLD would largely remain the same as that of Algorithm 8 (SGLD-PS), allowing for the aforementioned changes in preprocessing.

### 5.3.3 Worked example

Inspired by the experiments in Section 4.5.3 of Chapter 4, we now present a comparison of the pseudo-variance of our proposed gradient estimator against vanilla SGLD. For this worked example, we considered an imbalanced binary logistic regression on a synthetic dataset of size $N = 10^3$ [2]. This setup was chosen for its heavy class imbalance

[2]The model specification and data generating mechanism remained unchanged from Chapter 4. See Appendix B.2 for full details.

- with 95% of datapoints falling under one class ($y_i = 1$) and only 5% of datapoints categorised as the other ($y_i = 0$). All stochastic gradient evaluations were calculated at the mode (estimated using the ADAM optimiser) and the empirical pseudo-variance was computed for subsamples of up to size $k = 0.5N$.

The $k$-DPP kernel used for this example was taken from Experiment 5.2 of Zhang et al. (2017a), where the authors implemented a multi-class logistic regression. In this case, we used a linear kernel $L = FF^T$, where $F$ is a weighted concatenation of the features of the dataset $X$ and the class labels $\mathbf{y}$, with

$$F = [(1 - w)X \ w\mathbf{y}], \ 0 \leq w \leq 1.$$

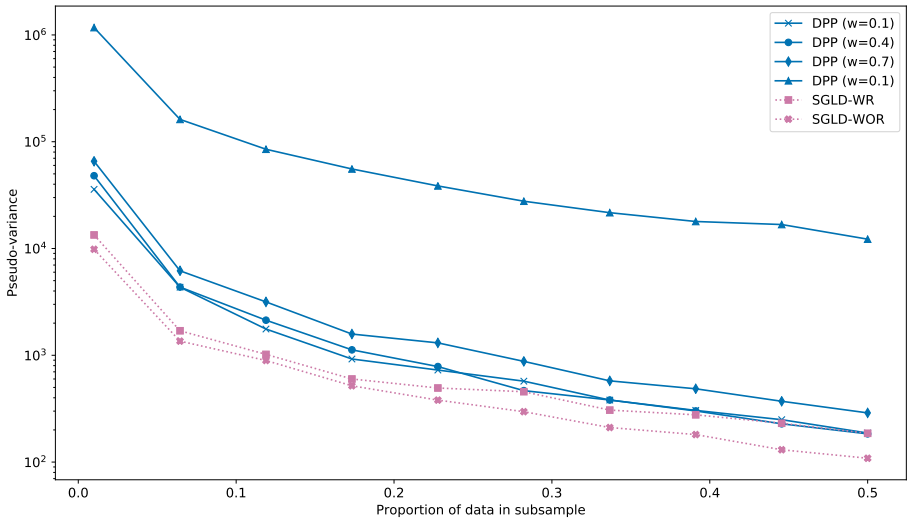
By changing the value of the weight $w$, the kernel can place greater emphasis on the class labels in the target. Random subsets were drawn from the $k$-DPP using the `DPPy` Python implementation of Algorithm 6 (Gautier et al., 2019).

Figure 5.3.1(a) compares the empirical pseudo-variance of the stochastic gradients for SGLD with and without replacement (WR and WOR respectively) and for DPP-SGLD with kernel weighting $w = 0.1, 0.4, 0.7$ and 1. Figure 5.3.1(b) plots histograms of the marginal probabilities used in reweighting the DPP-SGLD estimator Eq. (5.3.1). SGLD with and without replacement seems to offer the best variance reduction. As the value of $w$ increases, the performance of the DPP-SGLD gradient estimator is seen to degrade.

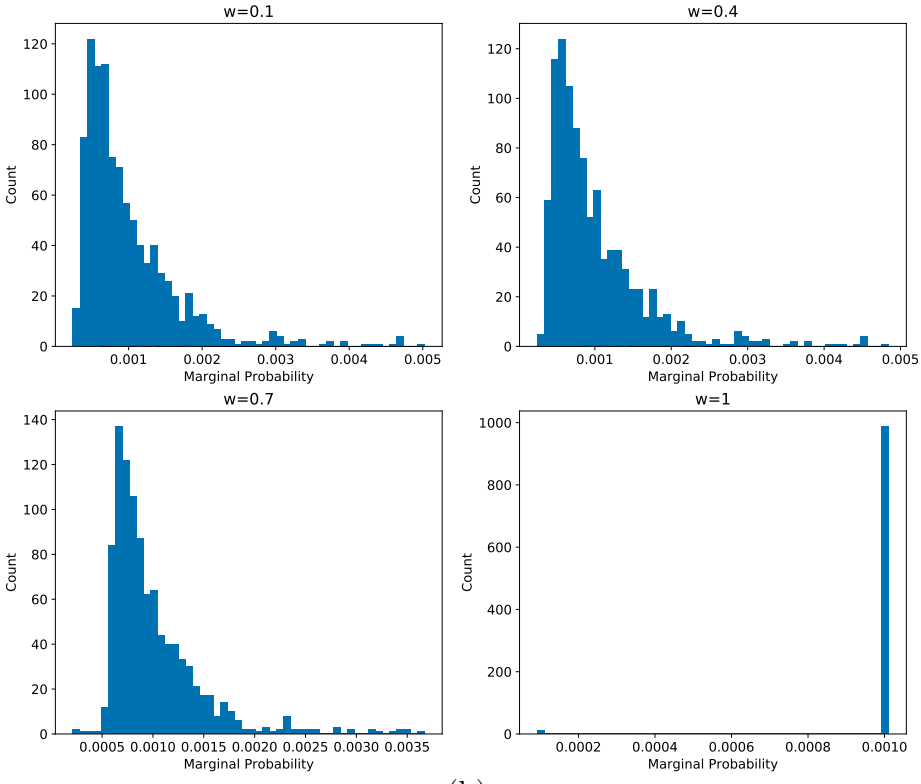
### 5.3.4 Discussion

In this section, we outline two practical issues that we have identified in trying to implement our approach. The performance of the proposed method is inherently tethered to the choice of kernel and the choice of dataset. At this stage, we are not concerned about the computational burden associated with drawing subsets from the $k$-DPP in preprocessing. There are various fast and scalable alternatives to Algorithm 6 available that do not rely upon an $O(N^3)$ eigendecomposition of the kernel matrix (Anari et al., 2016; Calandriello et al., 2020).

The worked example in Section 5.3.3 demonstrates that using $k$-DPPs to conduct




(a)




(b)

Figure 5.3.1: Empirical pseudo-variance against proportion of data in subsample $\frac{n}{N}$; (b) histograms of the marginal probabilities for used in computing Eq. (5.3.1) for $w = 0.1, 0.4, 0.7$ and 1.

subsampling for SGLD does not always guarantee a reduction in the variance of the estimator. Theorem 1 of Zhang et al. (2017a) sets out conditions in which we would expect to see variance reduction for SGD - implying that there are scenarios where we can reasonably expect to perform worse than uniform subsampling. In other words, the kernel matrix cannot be chosen in an agnostic manner. There unfortunately seems to be very little guidance in the existing literature on how to pick a similarity measure for unstructured problems that do not have a clear geometric interpretation.

One would also reasonably expect to be able to draw subsamples for SGLD of any size $k < N$. The rank of the kernel matrix $L$ serves as an upper bound on the value of $k$ that we can pick (Kulesza et al., 2012). We can temporarily introduce some numerical stability by defining the $k$-DPP model with $L + \epsilon I_{N \times N}$ for some small value of $\epsilon > 0$. However, the underlying dataset must also contain a substantial number of independent features so as to preserve the rank of the kernel as much as possible (i.e., wider datasets with limited multicollinearity are more preferable).

# Chapter 6

# Conclusions

## 6.1 Discussion

This thesis has focused on providing methodological contributions to two open areas of stochastic gradient MCMC research: extending the class of algorithms to dependent datasets and improving variance control.

We first considered the problem of extending SGLD to time series data. Stochastic gradient MCMC approaches typically require the parsed data sources to be independent. In considering dependent data sources, we were no longer able to rely on the convenient product structure of the likelihood to construct unbiased gradient estimators. In Chapter 3, we demonstrated how SGLD could be used to conduct scalable inference on nonlinear, non-Gaussian state space models, borrowing upon ideas from the particle filtering literature. This work served as the third and final paper in a series (Ma et al., 2017; Aicher et al., 2019). We proposed the use of particle buffered gradient estimators to account for the temporal dependencies in the data. We also presented an analysis of the sources of error (buffering error and particle error) introduced in our approach.

We then investigated how novel subsampling techniques can be used to improve variance control. Stochastic gradient MCMC constructs an unbiased estimate of the gradient of the log-posterior using a small, uniformly-weighted subsample of the data. While efficient to compute, the resulting estimator may have a relatively high variance and this can impact the performance of the sampler. There has been considerable

interest in developing tools for variance control in the literature (Dubey et al., 2016; Chatterji et al., 2018; Baker et al., 2019a). We note that much of this work builds upon existing variance reduction techniques for stochastic optimisation methods.

In Chapter 4, we proposed the use of a discrete, non-uniform probability distribution to preferentially subsample data points that have a greater impact on the stochastic gradient. We also presented a method for adaptively adjusting the subsample size at each SGLD iteration, so that we could increase the subsample size in areas of the sample space where the gradient was harder to estimate.

Chapter 5 sought to assess the viability of using determinantal point processes to construct informed data subsamples for SGLD, with a view to improve variance control. This idea was inspired by the work of Zhang et al. (2017a). We ran a worked example for an imbalanced logistic regression model on a small, synthetic dataset. In doing so, we understood that there were multiple practical issues associated with the approach.

## 6.2 Future work

The challenge of scalable inference for complex time series models has been explored in detail in Ma et al. (2017), Aicher et al. (2019) and Chapter 3. A natural extension of this work lies in adapting the proposed methods for streaming time series data. This could perhaps be done by borrowing ideas from Arulampalam et al. (2002) and Crisan and Míguez (2018). Given the similarities that dynamic latent space models share with state space models in the network literature (Sewell and Chen, 2015), it would be interesting to see if a buffered subsequence approach could be transferred across. To the best of our knowledge, the problem of subsampling spatial data, such that long-range and short-range dependency structure is preserved, has also not been solved.

The ideas presented in Chapter 4 open up quite a few extensions. Some further work should be done to improve the adaptive subsampling approach proposed in Chapter 4. At this stage, we are reliant upon running a full SGLD-CV chain ahead of

its adaptive counterpart, in order to obtain an upper bound on the variance of the stochastic gradient. A cheaper approximation could be to store the gradients obtained from the second half of optimiser burn-in and use those to propose a bound.

We could extend the preferential subsampling methodology of Chapter 4 further by employing multi-armed bandits (MAB) to achieve dynamic subsampling. Inspired by active learning methods, Salehi et al. (2017) propose a MAB framework for stochastic gradient descent to find the best possible approximation of the optimal subsampling distribution over time. Feedback is collected via the $l$ most recent stochastic gradients and passed into the MAB at each iteration. In this context, the MAB has $N$ arms, each of which corresponds to one of the N data points. Selecting arm (or data point) $i$ at time $t$ gives a negative reward (losses) $r_i^t$ and the losses vary between the arms. At time $t$, a MAB algorithm is used to update the arm sampling distribution $\mathbf{p}^{(t)}$ based on the loss associated with arm $i_t$, $r_{i_t}^t$. This is done without giving the algorithm access to any of the losses $r_{j_t}$ for $j \neq i_t$. Liu et al. (2020) went further and proposed a minibatching approach with an ADAM optimiser based on MABs, allowing for multiple datapoints (arms) to be considered per iteration. A similar adaptive subsampling strategy could be explored for SGLD, incorporating ideas proposed in both works.

In recent years, there has been considerable interest in the use of coresets for scalable applications. A coreset is a small, weighted subset of a large-scale dataset that approximates the full dataset and can be used in inference. Huggins et al. (2016) propose a method of efficient coreset construction for Bayesian logistic regression models. A similar approach could be considered for gradient-based MCMC samplers. Instead of iteratively using data subsampling, we could build up a coreset of the training data as a preprocessing step and use it to construct an unbiased estimate of the gradient of the log-posterior.

# Appendix A

# Appendix to Chapter 3

This Appendix is organised as follows. In Section A.1, we provide additional particle filter and gradient details for the models in Section 3.5.1. In Section A.2, we provide additional details and figures of experiments.

## A.1 Model details

### A.1.1 LGSSM

The LGSSM in this paper is given by

$$\begin{aligned} X_t \,|\, (X_{t-1} = x_{t-1}), \theta &\sim \mathcal{N}(x_t \,|\, \phi x_{t-1}\,,\, \sigma^2), \\ Y_t \,|\, (X_t = x_t), \theta &\sim \mathcal{N}(y_t \,|\, x_t\,,\, \tau^2), \end{aligned} \tag{A.1.1}$$

with parameters $\theta = (\phi, \sigma, \tau)$.

When applying the particle filter, Algorithm 2, to the LGSSM, we consider two proposal densities $q(\cdot|\cdot)$:

- The prior (transition) kernel

$$X_t \,|\, (X_{t-1} = x_{t-1}), \theta \sim \mathcal{N}(x_t \,|\, \phi x_{t-1}, \sigma^2), \tag{A.1.2}$$

where the weight update, (3.2.3), is

$$w_t^{(i)} \propto \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(\frac{-(y_t - x_t^{(i)})^2}{2\tau^2}\right). \tag{A.1.3}$$

- The 'optimal instrumental kernel'

$$\begin{aligned} &X_t \,|\, (X_{t-1} = x_{t-1}, Y_t = y_t), \theta \\ &\quad \sim \mathcal{N}\left(x_t \,\middle|\, \frac{\tau^2\phi x_{t-1} + \sigma^2 y_t}{\sigma^2 + \tau^2}, \frac{\sigma^2\tau^2}{\sigma^2 + \tau^2}\right), \end{aligned} \tag{A.1.4}$$

where the weight update, (3.2.3), is

$$w_t^{(i)} \propto \frac{1}{\sqrt{2\pi(\sigma^2 + \tau^2)}} \exp\left(\frac{-(y_t - \phi x_{t-1}^{(a_i)})^2}{2(\sigma^2 + \tau^2)}\right). \tag{A.1.5}$$

In our experiments with the LGSSM, we use the optimal instrumental kernel.

For this model, the (elementwise) complete data loglikelihood is

$$\begin{aligned} \log p(y_t, x_t \,|\, x_{t-1}, \theta) = &-\log(2\pi) - \log(\sigma) \\ &- \frac{(x_t - \phi x_{t-1})^2}{2\sigma^2} - \log(\tau) - \frac{(y_t - x_t)^2}{2\tau^2}. \end{aligned} \tag{A.1.6}$$

The gradient of the complete data loglikelihood is then,

$$\begin{aligned} \nabla_\phi \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{(x_t - \phi x_{t-1}) \cdot x_{t-1}}{\sigma^2}, \\ \nabla_\sigma \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{(x_t - \phi x_{t-1})^2 - \sigma^2}{\sigma^3}, \\ \nabla_\tau \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{(y_t - x_t)^2 - \tau^2}{\tau^3}. \end{aligned} \tag{A.1.7}$$

We reparametrize the gradients with $\sigma^{-1}$ and $\tau^{-1}$ to obtain,

$$\begin{aligned} \nabla_{\sigma^{-1}} \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{\sigma^2 - (x_t - \phi x_{t-1})^2}{\sigma}, \\ \nabla_{\tau^{-1}} \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{\tau^2 - (y_t - x_t)^2}{\tau}. \end{aligned} \tag{A.1.8}$$

To complete the SGMCMC scheme, the prior distributions of the parameters $\theta$ are

given as follows:

$$\begin{aligned}\phi &\sim \mathcal{N}(0, 100 \cdot \sigma^2)\\ \sigma^{-1} &\sim Gamma(1+100, (1+100)^{-1})\\ \tau^{-1} &\sim Gamma(1+100, (1+100)^{-1}).\end{aligned} \tag{A.1.9}$$

The initial parameter values for synthetic experiments were drawn from:

$$\begin{aligned}\phi &\sim \mathcal{N}(0, 1 \cdot \sigma^2)\\ \sigma^{-1} &\sim Gamma(2, 0.5)\\ \tau^{-1} &\sim Gamma(2, 0.5).\end{aligned} \tag{A.1.10}$$

### A.1.2 SVM

The SVM in this paper is given by,

$$\begin{aligned}X_t \,|\, (X_{t-1} = x_{t-1}), \theta &\sim \mathcal{N}(x_t \,|\, \phi x_{t-1}\,,\, \sigma^2),\\ Y_t \,|\, (X_t = x_t), \theta &\sim \mathcal{N}(y_t \,|\, 0\,,\, \exp(x_t)\tau^2),\end{aligned} \tag{A.1.11}$$

with parameters $\theta = (\phi, \sigma, \tau)$. In this model, the observations, $y_{1:T}$, represent the logarithm of the daily difference in the exchange rate and $X$ is the unobserved volatility. We assume that the volatility process is stationary (such that $0 < \phi < 1$), where $\phi$ is the persistence in volatility and $\tau$ is the instantaneous volatility.

For the particle filter, we use the prior kernel as the proposal density $q$

$$X_t \,|\, (X_{t-1} = x_{t-1}), \theta \sim \mathcal{N}(x_t \,|\, \phi x_{t-1}, \sigma^2), \tag{A.1.12}$$

with weight update

$$w_t^{(i)} \propto \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(\frac{-y_t^2}{2\exp(x_t^{(i)})\tau^2}\right). \tag{A.1.13}$$

The elementwise complete data loglikelihood is

$$\begin{aligned}\log p(\, y_t, x_t \,|\, x_{t-1}, \theta) = &-\log(2\pi) - \log(\sigma) - \log(\tau)\\ &- \frac{(x_t - \phi x_{t-1})^2}{2\sigma^2} - 0.5x_t - \frac{(y_t)^2}{2\exp(x_t)\tau^2}.\end{aligned} \tag{A.1.14}$$

The gradient of the complete data loglikelihood is then,

$$\begin{aligned}\nabla_\phi \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{(x_t - \phi x_{t-1}) \cdot x_{t-1}}{\sigma^2},\\ \nabla_\sigma \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{(x_t - \phi x_{t-1})^2 - \sigma^2}{\sigma^3},\\ \nabla_\tau \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{y_t^2/\exp(x_t) - \tau^2}{\tau^3}.\end{aligned} \tag{A.1.15}$$

We parametrize with $\sigma^{-1}$ and $\tau^{-1}$ to obtain,

$$\begin{aligned}\nabla_{\sigma^{-1}} \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{\sigma^2 - (x_t - \phi x_{t-1})^2}{\sigma},\\ \nabla_{\tau^{-1}} \log p(y_t, x_t \,|\, x_{t-1}, \theta) &= \frac{\tau^2 - y_t^2/\exp(x_t)}{\tau}.\end{aligned} \tag{A.1.16}$$

The prior distributions and initializations of the parameters $\theta$ are taken to be the same as in the LGSSM case.

### A.1.3 GARCH Model

The GARCH(1,1) model in this paper is given by,

$$\begin{aligned}&X_t \,|\, (X_{t-1} = x_{t-1}), \sigma_t^2, \theta \sim \mathcal{N}(x_t \,|\, 0,\, \sigma_t^2),\\ &\sigma_t^2(x_{t-1}, \sigma_{t-1}^2, \theta) = \alpha + \beta x_{t-1}^2 + \gamma \sigma_{t-1}^2,\\ &Y_t \,|\, (X_t = x_t), \theta \sim \mathcal{N}(y_t \,|\, x_t\,,\, \tau^2),\end{aligned} \tag{A.1.17}$$

where parameters are $\theta = (\log \mu, \operatorname{logit} \phi, \operatorname{logit} \lambda, \tau)$ for $\alpha = \mu(1-\phi)$, $\beta = \phi\lambda$, $\gamma = \phi(1-\lambda)$. Note that $\sigma_t^2 = \mu(1-\phi) + \phi(\lambda x_{t-1}^2 + (1-\lambda)\sigma_{t-1}^2)$.

We consider two proposal densities $q(\cdot|\cdot)$ for the GARCH model:

- The prior kernel

$$\begin{aligned}\begin{bmatrix} X_t \\ \sigma_t^2 \end{bmatrix} \;\Big|\; \begin{bmatrix} X_{t-1} = x_{t-1} \\ \sigma_{t-1}^2 \end{bmatrix}, \theta \\ \sim \begin{bmatrix} \mathcal{N}(x_t \,|\, 0, \alpha + \beta x_{t-1}^2 + \gamma \sigma_{t-1}^2) \\ \delta(\sigma_t^2 \,|\, \alpha + \beta x_{t-1}^2 + \gamma \sigma_{t-1}^2) \end{bmatrix}.\end{aligned} \tag{A.1.18}$$

  where the weight update, (3.2.3), is

$$w_t^{(i)} \propto \frac{1}{\sqrt{2\pi\tau^2}} \exp\left( \frac{-(y_t - x_t^{(i)})^2}{2\tau^2} \right). \tag{A.1.19}$$

- The optimal instrumental kernel

$$\begin{bmatrix} X_t \\ \sigma_t^2 \end{bmatrix} \Big| \begin{bmatrix} X_{t-1} = x_{t-1} \\ \sigma_{t-1}^2 \end{bmatrix}, (Y_t = y_t), \theta \sim \begin{bmatrix} \mathcal{N}(x_t \,|\, \sigma_t^2 y_t/(\sigma_t^2+\tau^2), \sigma_t^2\tau^2/(\sigma_t^2+\tau^2)) \\ \delta(\sigma_t^2 \,|\, \alpha + \beta x_{t-1}^2 + \gamma\sigma_{t-1}^2) \end{bmatrix}. \tag{A.1.20}$$

  where the weight update, (3.2.3), is

$$w_t^{(i)} \propto \frac{1}{\sqrt{2\pi((\sigma_t^{(i)})^2+\tau^2)}} \exp\left(\frac{-y_t^2}{2((\sigma_t^{(i)})^2+\tau^2)}\right). \tag{A.1.21}$$

In our experiments with the GARCH model, we use the optimal instrumental kernel.

The elementwise complete data loglikelihood is

$$\begin{aligned} \log p(y_t, x_t, \sigma_t^2 \,|\, x_{t-1}, \sigma_{t-1}^2, \theta) = & \\ & - \frac{\log(2\pi) + \log(\alpha + \beta x_{t-1}^2 + \gamma\sigma_{t-1}^2)}{2} \\ & - \frac{x_t^2}{2(\alpha + \beta x_{t-1}^2 + \gamma\sigma_{t-1}^2)} \\ & - 0.5\log(2\pi) - \log(\tau) - \frac{(y_t - x_t)^2}{2\tau^2}. \end{aligned} \tag{A.1.22}$$

Let $\mathcal{L}_t = \log p(y_t, x_t, \sigma_t^2 | x_{t-1}, \sigma_{t-1}^2, \theta)$ and set $C_t = \frac{x_t^2 - \sigma_t^2}{2\sigma_t^4}$. Then the gradient of the complete data loglikelihood $\nabla\mathcal{L}_t$ is

$$\begin{aligned} \nabla_\tau \mathcal{L}_t &= \frac{(y_t - x_t)^2 - \tau^2}{\tau^3}, \\ \nabla_{\log\mu} \mathcal{L}_t &= C_t \cdot (1-\phi) \cdot \mu, \\ \nabla_{\text{logit}\,\phi} \mathcal{L}_t &= C_t \cdot (\lambda x_{t-1}^2 + (1-\lambda)\sigma_{t-1}^2 - \mu) \cdot \phi(1-\phi), \\ \nabla_{\text{logit}\,\lambda} \mathcal{L}_t &= C_t \cdot (\phi x_{t-1}^2 - \phi\sigma_{t-1}^2) \cdot \lambda(1-\lambda). \end{aligned} \tag{A.1.23}$$

The SGMCMC scheme is completed by setting the prior distributions for the parameters as follows: $(\phi+1)/2 \sim \text{Beta}(10, 1.5)$, $\mu \sim \text{Uniform}(0, 2)$, $(\lambda+1)/2 \sim \text{Beta}(20, 1.5)$ and $\tau^2 \sim \mathcal{IG}(2, 0.5)$.

## A.2 Additional experiments

We first present the stochastic gradient bias when using other particle filtering methods and when varying the parameters with the LGSSM data. We then present additional SGLD results on synthetic data for the LGSSM in higher dimensions, the SVM and the GARCH models. We finally present some additional details for the SGLD experiment on the EUR-US exchange rate data.

### A.2.1 Gradient Bias with PaRIS

Figure A.2.1 compares the stochastic gradient bias of the naive PF with PaRIS on the LGSSM data in Section 3.5.2.

From Figure A.2.1 (top) and (bottom-left), we see that the naive PF (blue or solid line) performs similarly to PaRIS (red or dashed line) as $N$ varies. However, Figure A.2.1 (bottom-right) shows that the naive PF is about 10 times faster per iteration than PaRIS.

From Figure A.2.1 (top) and (bottom-left), we see that the naive PF (blue or solid line) performs similarly to PaRIS (red or dashed line) and Poyiadjis $N^2$ (green or dot-dashed line) as $N$ varies. However, Figure A.2.1 (bottom-right) shows that the naive PF is about 10 times faster per iteration than PaRIS and Poyiadjis $N^2$.

### A.2.2 Gradient Bias Varying Parameters

Figure A.2.2 compares the stochastic gradient bias for different values of $\phi \in (-0.97, 1.02)$ for the LGSSM experiment in Section 5.2 and shows the trade-off between the buffering error (II) and particle error (III) as $\phi$ (and therefore $L_\theta$) varies.

From Figure A.2.2 (left) the buffer methods are worse than using the full buffer (red) for $\phi > 1.00$ with $B = 8$ (blue), and $\phi > 1.01$ for $B = 16$ (purple). This is because the buffering error (II) decays less rapidly with $B$ as $\phi$ increases.

Comparing the naive PF ($N = 1000$) to the Kalman filter, Figure A.2.2 (left vs right), we see there is a large gap due to particle error (III) as well. Therefore, as

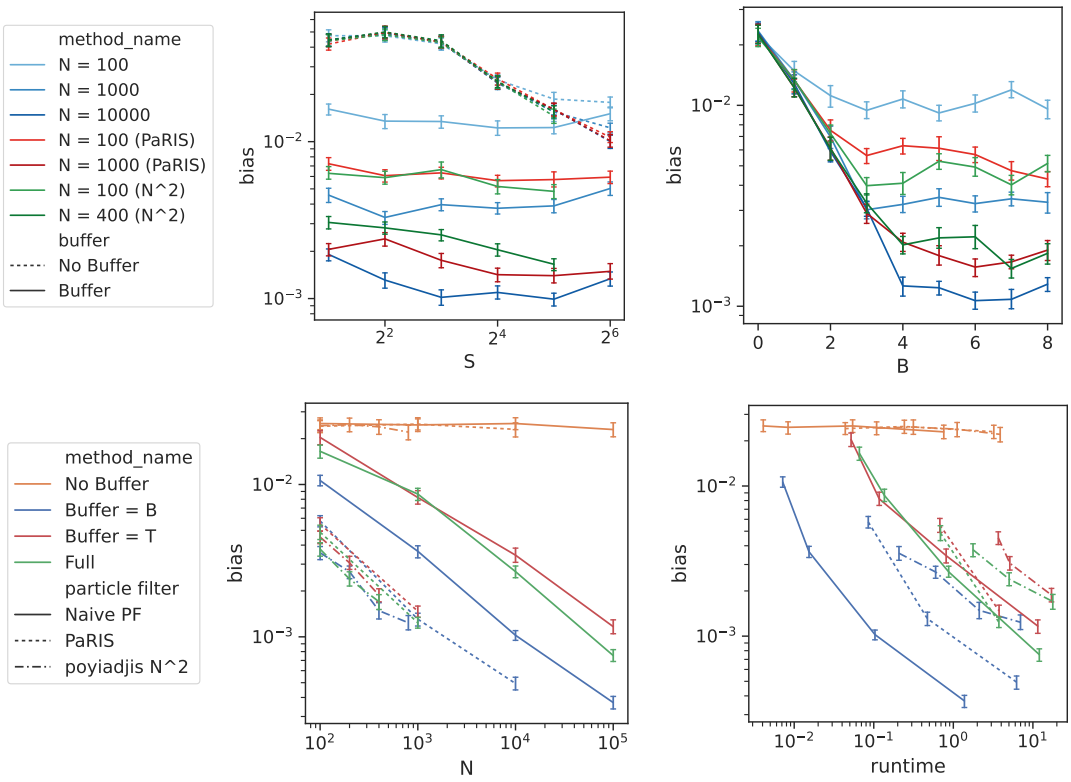



Figure A.2.1: Stochastic gradient bias varying $B, S, N$ for the naive PF and PaRIS on the LGSSM data. (Top-left) bias vs $S$, (top-right) bias vs $B$, (bottom-left) bias vs $N$, (bottom-right) bias vs runtime in seconds.

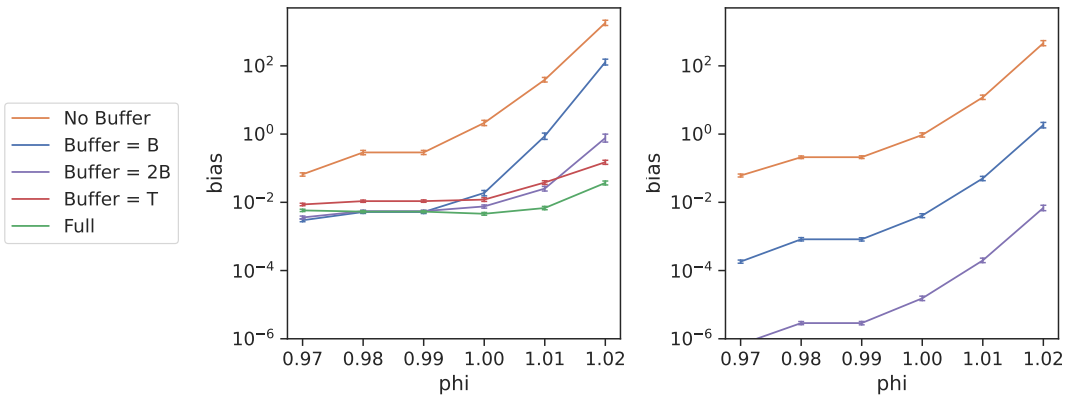



Figure A.2.2: Stochastic gradient bias varying $\phi$ with $S = 16, B = 8$ for (left) naive PF $N = 1000$, (right) Kalman filter $N = \infty$.

$\phi$ increases, both $B$ and $N$ need to increase to control bias; otherwise the buffered methods have larger bias than full sequence gradients (green)

And again, in all cases, not using a buffer (orange) has the largest bias.

### A.2.3 SGLD on Synthetic Data

#### Additional MSE Figures for LGSSM

Figure A.2.3 presents extra MSE plots for the parameters not presented in the main paper. Tables A.2.1 and A.2.2 present the full KSD results for each variable.

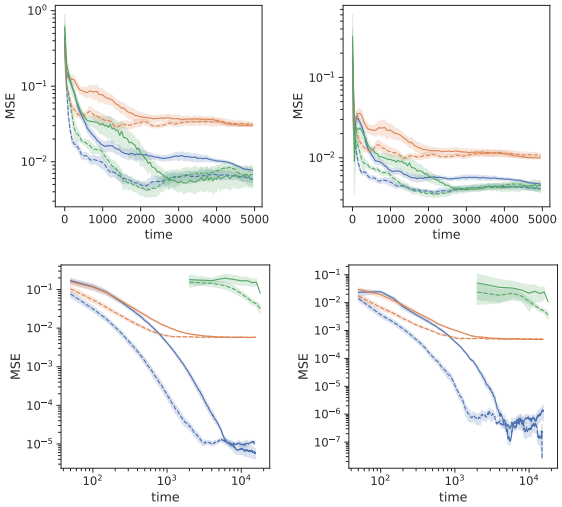



Figure A.2.3: Additional metrics for SGLD on LGSSM: (left) MSE of $\sigma$, (right) MSE of $\tau$, (top) $T = 10^3$, (bottom) $T = 10^6$.

#### Higher Dimensional LGSSM

We generate synthetic LGSSM data for $X_t, Y_t \in \mathbb{R}^d$ using $\phi = 0.9 \cdot \mathbb{I}_d$, $\sigma = 0.7 \cdot \mathbb{I}_d$, and $\tau = \mathbb{I}_d$ for dimensions $d \in \{5, 10\}$ with $T = 1000$. Figure A.2.4 presents the MSE trace plots for $d = 5$ and for $d = 10$. Table A.2.3 presents the KSD tables for both $d = 5$ and $d = 10$.

We find that the Kalman filter $N = \infty$ is able to much more rapidly mix compared to the particle filter with $N = 1000$. This is both due to the increased particle filter variance in higher dimensions and the longer computation required for sampling particles in higher dimensions. However in both cases, we again see that buffering is necessary to avoid bias.

Table A.2.1: KSD results for Synthetic LGSSM with $T = 10^3$.

| | | | $\log_{10}$KSD | | | |
|---|---|---|---|---|---|---|
| $S$ | $B$ | METHOD | $\phi$ | $\sigma$ | $\tau$ | TOTAL |
| $10^3$ | – | GIBBS | 0.09 (0.25) | -0.02 (0.01) | -0.16 (0.48) | **0.51 (0.13)** |
| | | KF | 0.01 (0.57) | 0.07 (0.09) | 0.20 (0.28) | **0.64 (0.17)** |
| | | PF | 0.38 (0.26) | 0.10 (0.16) | 0.44 (0.19) | 0.85 (0.08) |
| 40 | 0 | KF | 1.53 (0.03) | -0.08 (0.07) | -0.04 (0.16) | 1.55 (0.03) |
| | | PF | 1.55 (0.03) | -0.04 (0.13) | 0.10 (0.26) | 1.58 (0.03) |
| 40 | 10 | KF | 0.18 (0.27) | 0.02 (0.07) | 0.04 (0.44) | **0.61 (0.21)** |
| | | PF | 0.27 (0.46) | 0.09 (0.13) | -0.11 (0.53) | **0.68 (0.25)** |

### SVM

Figure A.2.5 presents the MSE plots for SGLD on the synthetic SVM data $T = 1000$ and Table A.2.4 presents the KSD for each sampled chain.

We find that buffering performs best (as measured by KSD). From Figure A.2.5 we see that not buffering leads to bias, while the full sequence method is nosier (fewer larger steps) compared to the buffer method.

### GARCH

Figure A.2.6 presents the trace plot metrics for SGLD on the synthetic GARCH data $T = 1000$ and Table A.2.5 presents the KSD for each sampled chain.

We again find that buffering performs best (as measured by KSD). From Figure A.2.6 we see that not buffering leads to bias in sampling $\mu$ and $\lambda$. The full sequence method encounters high particle error and therefore requires a much longer runtime with a much smaller stepsize to reduce bias.

Table A.2.2: KSD results for Synthetic LGSSM with $T = 10^6$.

| | | | $\log_{10}$KSD | | | |
|---|---|---|---|---|---|---|
| $S$ | $B$ | METHOD | $\phi$ | $\sigma$ | $\tau$ | TOTAL |
| $10^6$ | – | GIBBS | 3.91 (0.80) | 3.43 (1.07) | 3.52 (0.73) | 4.23 (0.74) |
| | | KF | 4.51 (0.48) | 4.21 (0.50) | 3.65 (0.55) | 4.85 (0.36) |
| | | PF | 4.77 (0.39) | 4.11 (0.57) | 3.55 (0.95) | 4.92 (0.40) |
| 40 | 0 | KF | 4.64 (0.14) | 3.25 (0.21) | 2.83 (0.61) | 4.68 (0.11) |
| | | PF | 4.64 (0.13) | 3.19 (0.35) | 3.12 (0.45) | 4.68 (0.10) |
| 40 | 10 | KF | 3.04 (0.39) | 1.57 (0.50) | 2.68 (0.20) | **3.25 (0.29)** |
| | | PF | 3.26 (0.17) | 1.70 (0.38) | 2.87 (0.33) | **3.43 (0.19)** |

Table A.2.3: KSD results for Synthetic LGSSM in higher dimensions

| | | | $\log_{10}$KSD | | | |
|---|---|---|---|---|---|---|
| DIM | GRAD EST. | N | $\phi$ | $\sigma$ | $\tau$ | TOTAL |
| 5 | NO BUFFER | 1000 | 1.78 (0.04) | 1.97 (0.26) | 1.44 (0.45) | 2.28 (0.20) |
| | | $\infty$ | 1.74 (0.01) | 2.09 (0.02) | 1.64 (0.02) | 2.35 (0.01) |
| | BUFFER | 1000 | 1.18 (0.17) | 1.74 (0.25) | 1.44 (0.03) | **2.01 (0.13)** |
| | | $\infty$ | 0.84 (0.03) | 1.97 (0.03) | 1.40 (0.05) | **2.10 (0.03)** |
| 10 | NO BUFFER | 1000 | 1.84 (0.01) | 2.40 (0.06) | 2.26 (0.13) | 2.71 (0.06) |
| | | $\infty$ | 1.79 (0.01) | 2.13 (0.04) | 2.12 (0.01) | 2.52 (0.02) |
| | BUFFER | 1000 | 1.60 (0.13) | 2.37 (0.04) | 2.20 (0.04) | 2.64 (0.04) |
| | | $\infty$ | 1.04 (0.06) | 2.08 (0.04) | 2.07 (0.01) | **2.39 (0.02)** |

### A.2.4 SGLD on Exchange Rate

The EUR-US exchange rate data was pulled from the `https://www.finam.ru` website for the time period of November 2017 to October 2018 at the minute resolution. The

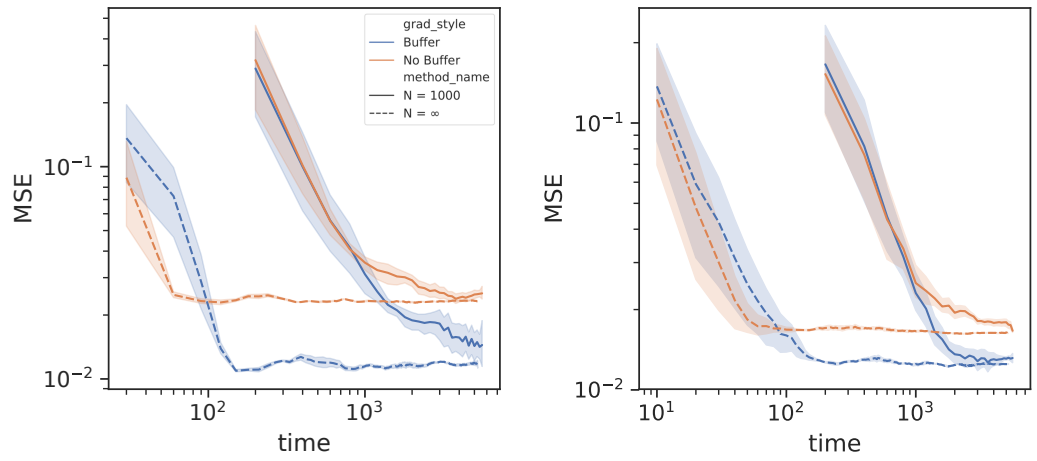



Figure A.2.4: SGLD Results for LGSSM. MSE of $\phi$ (left) for $X \in \mathbb{R}^5$, (right) $X \in \mathbb{R}^{10}$.

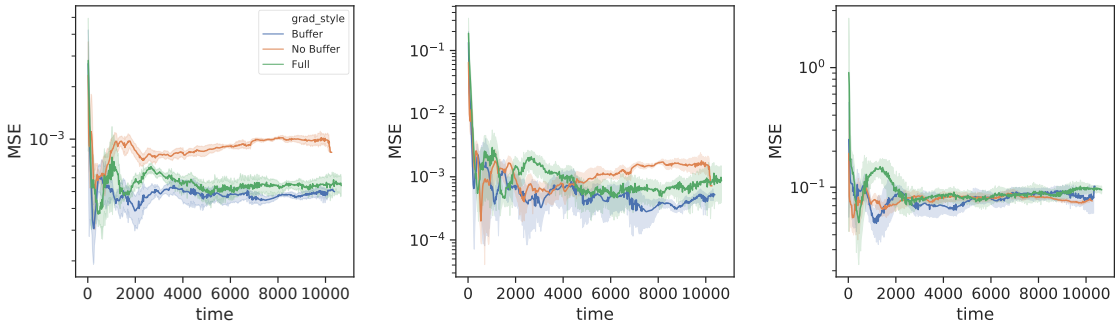



Figure A.2.5: SGLD results for synthetic SVM data: (left) MSE of $\phi$, (center) MSE of $\sigma$, (right) MSE of $\tau$.

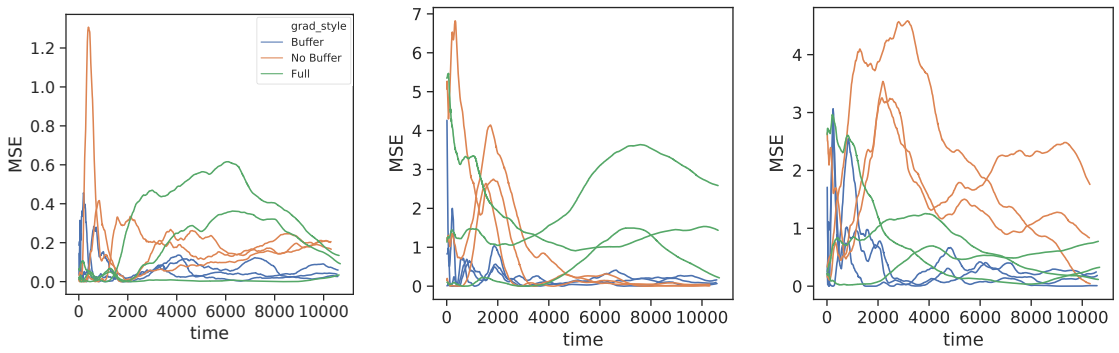



Figure A.2.6: SGLD results for synthetic GARCH data: (left) MSE of $\log(\mu)$, (center) MSE of logit $\phi$, (right) MSE of logit $\lambda$.

data is plotted in Figure A.2.7.

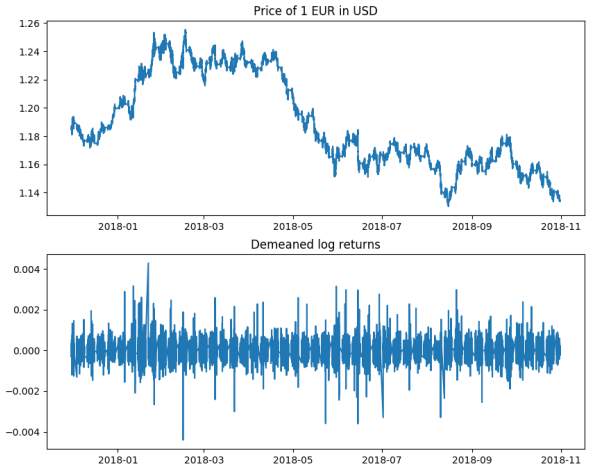



Figure A.2.7: EUR-US Exchange Rate Data (top) raw data (bottom) demeaned log-returns.

The *demeaned log-returns* are calculated by taking the difference of the log-closing price (at each minute) and removing the mean, as done in the `stochvol` package in R Kastner (2016)

$$\tilde{y}_t = \log(y_t/y_{t-1}) - \frac{1}{T}\sum_{t'} \log(y_{t'}/y_{t'-1}). \tag{A.2.1}$$

**SVM**

For the SVM, we initialized each chain at $\phi = 0.9$, $\sigma = 1.73$ and $\tau = 0.1$ for all SGLD methods. The full KSD results are presented in Table A.2.6.

**GARCH**

For the GARCH model, we initialized each chain at $\log \mu = -0.4$, $\operatorname{logit} \phi = 1.7$, $\operatorname{logit} \lambda = 2.7$ and $\tau = 0.1$ for all SGLD methods. The full KSD results are presented in Table A.2.7.

Table A.2.4: KSD results for Synthetic SVM.

| | $\log_{10}$KSD | | | |
|---|---|---|---|---|
| Grad Est. | $\phi$ | $\sigma$ | $\tau$ | Total |
| Full | 0.68 (0.28) | 0.38 (0.40) | 0.44 (0.54) | 1.12 (0.22) |
| No Buffer | 1.49 (0.05) | -0.01 (0.23) | 0.09 (0.35) | 1.53 (0.05) |
| Buffer | 0.35 (0.33) | 0.23 (0.29) | 0.21 (0.40) | **0.81 (0.22)** |

Table A.2.5: KSD results for Synthetic GARCH.

| | $\log_{10}$KSD | | | | |
|---|---|---|---|---|---|
| Grad Est. | $\log \mu$ | $\text{logit}\, \lambda$ | $\text{logit}\, \phi$ | $\tau$ | Total |
| Full | 0.29 (0.59) | 0.04 (0.03) | 0.18 (0.34) | 0.55 (0.11) | 0.97 (0.05) |
| No Buffer | 0.07 (0.08) | -0.38 (0.09) | -0.15 (0.10) | 0.56 (0.10) | 0.77 (0.08) |
| Buffer | -0.27 (0.24) | -0.72 (0.19) | -0.69 (0.17) | 0.12 (0.19) | **0.39 (0.09)** |

Table A.2.6: KSD results for SVM on exchange rate data.

| | $\log_{10}$KSD | | | |
|---|---|---|---|---|
| Grad Est. | $\phi$ | $\sigma$ | $\tau$ | Total |
| Full | 3.63 (0.30) | 3.76 (0.07) | 1.46 (0.38) | 4.03 (0.14) |
| Weekly | 3.86 (0.08) | 2.18 (0.28) | 0.67 (0.39) | 3.87 (0.08) |
| No Buffer | 4.48 (0.01) | 1.84 (0.15) | 1.21 (0.14) | 4.48 (0.01) |
| Buffer | 3.53 (0.11) | 2.32 (0.13) | 1.23 (0.05) | **3.56 (0.10)** |

Table A.2.7: KSD results for GARCH on exchange rate data.

| | $\log_{10}$KSD | | | | |
|---|---|---|---|---|---|
| Grad Est. | $\log \mu$ | $\text{logit}\, \lambda$ | $\text{logit}\, \phi$ | $\tau$ | Total |
| Full | 2.18 (0.67) | 2.18 (0.07) | 2.19 (0.61) | 2.07 (0.06) | 2.84 (0.30) |
| Weekly | 2.17 (0.51) | 2.21 (0.03) | 2.31 (0.29) | 1.85 (0.19) | 2.81 (0.21) |
| No Buffer | 1.76 (0.06) | 1.43 (0.46) | 1.31 (0.09) | 1.58 (0.08) | **2.09 (0.09)** |
| Buffer | 1.76 (0.03) | 2.01 (0.08) | 1.11 (0.07) | 1.87 (0.07) | **2.19 (0.05)** |

# Appendix B

# Appendix to Chapter 4

This Appendix is organised as follows. In Section B.1, we provide the pseudocode for the SGLD-CV, SGLD-PS and SGLD-CV-PS algorithms. In Section B.2, we provide the model specifications, gradients and Lipschitz constants for the models used in Section 3.5.1. We also describe the synthetic and real datasets used in the experiments. In Section B.3, we outline the preprocessing costs associated with implementing SGLD-CV-PS. Finally, we describe the numerical experiment set-up and provide additional figures in Sections B.4 and B.5 respectively.

## B.1 Pseudocode for algorithms

**Algorithm 7:** SGLD-CV

Input: initialise $\theta^{(1)} = \hat{\theta}$, gradients $\nabla f_i(\hat{\theta})$, batch size $n$, step-size $\epsilon$.
**for** $t = 1, 2, \ldots, T$ **do**
    Sample indices $S^t \subset \{1, \ldots, N\}$ with or without replacement.
    Calculate $\hat{g}_{cv}^{(t)}$ in (4.2.5).
    Update parameters according to (4.2.4).
**end for**
**return** $\theta^{(T+1)}$

**Algorithm 8:** SGLD with preferential subsampling (SGLD-PS)

Input: initialise $\theta^{(1)}$, weights $\mathbf{p}^{(1)}$, batch size $n$, step-size $\epsilon$.
**for** $t = 1, 2, \ldots, T$ **do**
  Update $\mathbf{p}^{(t)}$.
  Sample indices $S^t$ according to $\mathbf{p}^{(t)}$ with replacement.
  Calculate $\tilde{g}^{(t)}$ using (4.3.3).
  Update parameters $\theta^{(t+1)} \leftarrow \theta^{(t)} - \frac{\epsilon}{2} \cdot \tilde{g}^{(t)} + \mathcal{N}_d(0, \epsilon_t I_{d\times d})$
**end for**
**return** $\theta^{(T+1)}$

**Algorithm 9:** SGLD-CV with preferential subsampling (SGLD-CV-PS)

Input: initialise $\theta^{(1)}$ close to the mode $\hat{\theta}$, gradients $\nabla f_i(\hat{\theta})$, weights $\mathbf{p}^{(1)}$, batch size $n$, step-size $\epsilon$.
**for** $t = 1, 2, \ldots, T$ **do**
  Update $\mathbf{p}^{(t)}$.
  Sample indices $S^t$ according to $\mathbf{p}^{(t)}$ with replacement.
  Calculate $\tilde{g}^{(t)}$ using (4.3.7).
  Update parameters $\theta^{(t+1)} \leftarrow \theta^{(t)} - \frac{\epsilon}{2} \cdot \tilde{g}^{(t)} + \mathcal{N}_d(0, \epsilon_t I_{d\times d})$
**end for**
**return** $\theta^{(T+1)}$

## B.2 Model details

### B.2.1 Bivariate Gaussian

#### Model specification

We want to simulate independent data from:

$$X_i|\theta \sim \mathcal{N}_2(\theta, \Sigma_x) \quad \text{for } i = 1, \ldots, N.$$

It is assumed that that $\theta$ is unknown and $\Sigma_x$ is known. The likelihood for a single

observation is given by:

$$p(x_i|\theta) = \frac{1}{\sqrt{(2\pi)^2|\Sigma_x|}} \exp\left( -\frac{1}{2}(x_i-\theta)^T\Sigma_x^{-1}(x_i-\theta)\right).$$

The likelihood function for $N$ observations is

$$\begin{aligned} p(\mathbf{x}|\theta) &= \prod_{i=1}^{N} \frac{1}{\sqrt{(2\pi)^2|\Sigma_x|}} \exp\left( -\frac{1}{2}(x_i-\theta)^T\Sigma_x^{-1}(x_i-\theta)\right) \\ &\propto |\Sigma_x|^{-\frac{N}{2}} \exp\left( -\frac{1}{2}\sum_{i=1}^{N}(x_i-\theta)^T\Sigma_x^{-1}(x_i-\theta)\right). \end{aligned}$$

The loglikelihood is a quadratic form in $\theta$, and therefore the conjugate prior distribution for $\theta$ is the multivariate normal distribution. The conjugate prior for $\theta$ is set to be

$$\theta \sim \mathcal{N}_2(\mu_0, \Lambda_0).$$

The conjugate posterior that we are ultimately trying to simulate from using SGLD is known to be:

$$\pi(\theta|\mathbf{x}) \propto \exp\left( -\frac{1}{2}(\theta-\mu_1)^T\Lambda_1^{-1}(\theta-\mu_1)\right) \stackrel{D}{=} \mathcal{N}_2(\mu_1, \Lambda_1),$$

where

$$\mu_1 = (\Lambda_0^{-1} + N\Sigma_x^{-1})^{-1}(\Lambda_0^{-1}\mu_0 + N\Sigma_x^{-1}\bar{x}),$$

and

$$\Lambda_1^{-1} = \Lambda_0^{-1} + N\Sigma_x^{-1}.$$

**Model gradient**

For the prior,

$$\log p(\theta) = -\frac{1}{2}(\theta-\mu_0)^T\Lambda_0^{-1}(\theta-\mu_0)$$

Therefore,

$$\nabla f_0(\theta) = -\nabla \log p(\theta) = \Lambda_0^{-1}(\theta-\mu_0).$$

We know that for $i \in \{1, \ldots, N\}$,

$$f_i(\theta) = -\log p(x_i|\theta) = \frac{1}{2}(x_i - \theta)^T \Sigma_x^{-1}(x_i - \theta) + \text{constant}$$

Therefore,

$$\nabla f_i(\theta) = \Sigma_x^{-1}(\theta - x_i) \text{ and } \nabla^2 f_i(\theta) = \Sigma_x^{-1}.$$

**Synthetic data**

To generate the synthetic data, $N$ data points are drawn from the model with $\theta = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and $\Sigma_x = \begin{pmatrix} 1 \times 10^5 & 6 \times 10^4 \\ 6 \times 10^4 & 2 \times 10^5 \end{pmatrix}$. The hyperparameters of the prior are $\mu_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and $\Lambda_0 = \begin{pmatrix} 1 \times 10^3 & 0 \\ 0 & 1 \times 10^3 \end{pmatrix}$. Synthetic datasets of sizes $N = 10^3$ and $N = 10^4$ were generated for use in Figures 4.5.1(a) and B.5.1(a) respectively.

## B.2.2 Logistic regression

**Model specification**

Suppose we have data $x_1, \ldots, x_N$ of dimension $d$ taking values in $\mathbb{R}^d$, where each $x_i = (1, x_{i1}, \ldots, x_{ip})^T$ $(d = p + 1)$. Let us suppose that we also have the corresponding response variables $y_1, \ldots, y_N$ taking values in $\{0, 1\}$. Then, a logistic regression model with parameters $\theta = (\beta_0, \beta_1, \ldots, \beta_p)$ representing the coefficients $\beta_j$ for $j = 1, \ldots, p$ and bias $\beta_0$ will have likelihood

$$p(X, y|\theta) = \prod_{i=1}^{N} \left[\frac{1}{1 + e^{-\theta^T x_i}}\right]^{y_i} \left[1 - \frac{1}{1 + e^{-\theta^T x_i}}\right]^{1-y_i}.$$

The prior for $\theta$ is set to be $\theta \sim \mathcal{N}_d(\mu_0, \Lambda_0)$. The hyperparameters of the prior are $\mu_0 = (0, \ldots, 0)^T$ and $\Lambda_0 = \text{diag}(10, d)$.

**Model gradient and Hessian**

For the prior,

$$\log p(\theta) = -\frac{1}{2}(\theta - \mu_0)^T \Lambda_0^{-1}(\theta - \mu_0) = -\frac{1}{2}\theta^T \Lambda_0^{-1}\theta.$$

Therefore,

$$\nabla f_0(\theta) = -\log p(\theta) = \Lambda_0^{-1}\theta.$$

We know that $i \in \{1, \ldots, N\}$, the log-density

$$\begin{aligned}
\log p(y_i|x_i,\theta) &= y_i \log\left(\frac{1}{1+\exp(-\theta^T x_i)}\right) + (1-y_i)\log\left(1 - \frac{1}{1+\exp(-\theta^T x_i)}\right) \\
&= y_i \log\left(\frac{1}{1+\exp(-\theta^T x_i)}\right) + (1-y_i)\log\left(\frac{\exp(-\theta^T x_i)}{1+\exp(-\theta^T x_i)}\right) \\
&= y_i \log\left(\frac{1}{1+\exp(-\theta^T x_i)} \cdot \frac{1+\exp(-\theta^T x_i)}{\exp(-\theta^T x_i)}\right) + \log\left(\frac{\exp(-\theta^T x_i)}{1+\exp(-\theta^T x_i)}\right) \\
&= y_i \log(\exp(\theta^T x_i)) + \log\left(\frac{1}{1+\exp(\theta^T x_i)}\right) \\
&= y_i \theta^T x_i \; - \log(1+\exp(\theta^T x_i))
\end{aligned}$$

Therefore,

$$f_i(\theta) = -\log p(y_i|x_i,\theta) = \log(1+\exp(\theta^T x_i)) - y_i \theta^T x_i,$$

and the corresponding gradient vector is

$$\nabla f_i(\theta) = \frac{\exp(\theta^T x_i)}{1+\exp(\theta^T x_i)} \cdot x_i - y_i \cdot x_i = \frac{1}{1+\exp(-\theta^T x_i)} \cdot x_i - \; y_i \cdot x_i$$

The corresponding Hessian is

$$\nabla^2 f_i(\theta) = \frac{\exp(-\theta^T x_i)}{(1+\exp(-\theta^T x_i))^2} \cdot x_i x_i^T \, .$$

We know that for the function $h(a) = \log(1+\exp(-a)), h''(a) = \frac{\exp(-a)}{(1+exp(-a))^2} \leq \frac{1}{4}$. Therefore,

$$\nabla^2 f_i(\theta) = \frac{\exp(-\theta^T x_i)}{(1+\exp(-\theta^T x_i))^2} \cdot x_i x_i^T \preceq \frac{1}{4} x_i x_i^T.$$

Using results from Durmus and Moulines (2019) and Dwivedi et al. (2018), we know that $f_i(\theta)$ is $L_i$-continuous with $L_i = \frac{1}{4}\lambda_{\max}(x_i x_i^T)$.

#### Synthetic data

We used the Python module `sklearn` to produce our synthetic classification data with four features ($d = 5$). We have generated $N$ training data points and $N_{test} = 0.5N$ test data points.

Two types of synthetic data were generated:

1. Balanced classification data, where 50% of instances have $y_i = 1$. Synthetic datasets of sizes $N = 10^3$ and $N = 10^4$ were used for Figures 4.5.1(b) and 4.5.3 respectively.

2. Highly imbalanced classification data, where 95% of the instances have $y_i = 1$. Synthetic data of size $N = 10^3$ was used for Figure 4.5.1(c).

#### Real data

We used the covertype dataset (Blackard and Dean, 1998) for Figures 4.5.2(b)-(c) and B.5.1(b). The covertype dataset contains 581,012 instances and 54 features. In particular, we have used a transformed version of this dataset that is available via the LIBSVM repository[1]. We split the covertype dataset into training and test sets with 75% and 25% of the instances respectively.

### B.2.3 Linear regression

#### Model specification

Suppose we have data $x_1, \ldots, x_N$ of dimension $d$ taking values in $\mathbb{R}^d$, where each $x_i = (1, x_{i1}, \ldots, x_{ip})^T$ $(d = p + 1)$. Let us suppose that we also have the corresponding response variables $y_1, \ldots, y_N$ taking values on the real line.

We define the following linear regression model,

$$y_i = x_i^T \theta + \eta_i, \;\; \eta_i \sim \mathcal{N}(0, 1)$$

[1] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`

with parameters $\theta = (\beta_0, \beta_1, \ldots, \beta_p)$ representing the coefficients $\beta_j$ for $j = 1, \ldots, p$ and bias $\beta_0$. The regression model will thus have likelihood

$$p(X, y|\theta) = \prod_{i=1}^{N} \left[ \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{1}{2}(y_i - x_i^T\theta)^2 \right) \right].$$

The prior for $\theta$ is set to be $\theta \sim \mathcal{N}_d(\mu_0, \Lambda_0)$. The hyperparameters of the prior are $\mu_0 = (0, \ldots, 0)^T$ and $\Lambda_0 = \text{diag}(10, d)$.

**Model gradient and Hessian**

$$\log p(\theta) = -\frac{1}{2}(\theta - \mu_0)^T \Lambda_0^{-1}(\theta - \mu_0) = -\frac{1}{2}\theta^T \Lambda_0^{-1}\theta.$$

Therefore,

$$\nabla f_0(\theta) = -\nabla \log p(\theta) = \Lambda_0^{-1}\theta.$$

We know that $i \in \{1, \ldots, N\}$, the log-density

$$\log p(y_i|x_i, \theta) = -\frac{1}{2}(y_i - x_i^T\theta)^2 - \frac{1}{2}\log(2\pi).$$

Therefore,

$$f_i(\theta) = -\log p(y_i|x_i, \theta) = \frac{1}{2}(y_i - x_i^T\theta)^2 + \frac{1}{2}\log(2\pi)$$

and the corresponding gradient vector is

$$\nabla f_i(\theta) = -(y_i - x_i^T\theta) \cdot x_i.$$

The corresponding Hessian is

$$\nabla^2 f_i(\theta) = x_i x_i^T .$$

Using results from Dwivedi et al. (2018), we know that $f_i(\theta)$ is $L_i$-continuous with $L_i = \lambda_{\max}(x_i x_i^T)$.

**Real data**

We used the CASP[2] dataset from the UCI Machine Learning repository for Figures 4.5.2(a) and B.5.2. The CASP dataset contains 45,730 instances and 9 features.

## B.3 Computational cost for the SGLD-CV-PS approximate subsampling weights

Recall that the optimal subsampling weights in (4.3.9) can be approximated by the following scheme,

$$p_i \propto \sqrt{\mathrm{tr}\left(\nabla^2 f_i(\hat{\theta})\,\hat{\Sigma}\,\nabla^2 f_i(\hat{\theta})^T\right)} \text{ for } i = 1, \ldots, N.$$

Here, $\nabla^2 f_i(\cdot)$ is the Hessian matrix of $f_i(\cdot)$ and $\hat{\Sigma}$ is the covariance matrix of the Gaussian approximation to the target posterior centred at the mode.

In practice, these weights should be evaluated as a one-off preprocessing step before the SGLD-CV-PS chain is run. We now assess the total computational cost associated with this preprocessing step.

- The Hessian of each $f_i(\cdot)$ need to evaluated at the mode. For each log-density function, this step costs $O(d^2)$.
- The covariance matrix of the Gaussian approximation of the posterior needs to calculated once. This involves inverting the observed information matrix at a cost of $O(d^3)$.
- The cost of multiplying three $d \times d$ square matrices is $O(d^3)$
- The cost of calculating the trace of a $d \times d$ matrix is $O(d)$.
- The cost of calculating the square root of a scalar is $O(1)$.

[2] `https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Structure`

Therefore, the cost of calculating these weights for all $N$ data points is $O(Nd^3)$. As such, we recognise that there will be limits to where the SGLD-CV-PS algorithm can be used. The SGLD-CV-PS has been implemented with success on the covertype dataset Blackard and Dean (1998) (with 54 features and 581,012 instances and 54 features) in Section 4.5.3. We recommend that this method is not implemented for models with more than $d > 60$ parameters in practice.

## B.4 Numerical experiment set-up

### B.4.1 Step-size selection

SGLD-type algorithms do not mix well when the step-size is decreased to zero. It is therefore common (and in practice easier) to implement SGLD with a fixed step-size, as suggested by Vollmer et al. (2016). For Figures 4.5.2 - B.5.2, all samplers were run with the same step-size (with $\epsilon \approx \frac{1}{N}$). This allowed us to control for discretisation error and to independently assess the performance benefits offered by preferential subsampling. We list the step-sizes used for each experiment in Table B.4.1 below.

Table B.4.1: Step-size selection.

| **FIGURE** | **DATA** | **SIZE OF DATA** | **STEP-SIZE** |
|---|---|---|---|
| B.5.1(a) | synthetic bivariate Gaussian | $10,000$ | $1 \times 10^{-4}$ |
| B.5.1(b) | synthetic balanced logistic regression | $10,000$ | $1 \times 10^{-4}$ |
| 4.5.2(b)-(c), 4.5.3 | covertype | $581,012$ | $1 \times 10^{-6}$ |
| 4.5.2(a), B.5.2 | CASP | $45,730$ | $1 \times 10^{-5}$ |

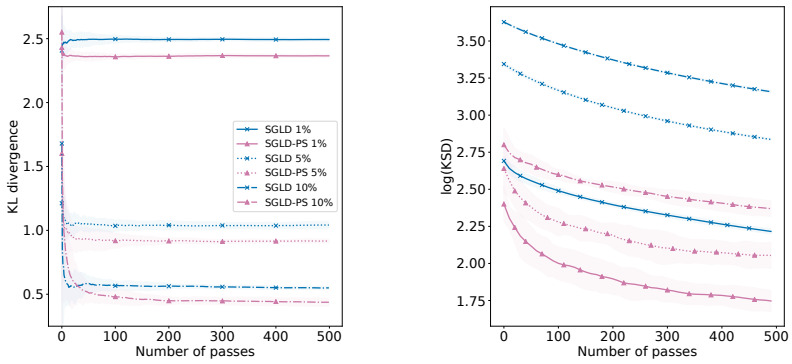
### B.4.2 Initialisation

Throughout our experiments, we were consistent in how we picked our initial start values, $\theta^{(0)}$, for SGLD and the ADAM optimiser. We sampled $\theta^{(0)}$ from the prior for

the bivariate Gaussian model. Whereas, we set $\theta^{(0)} = \mathbf{0}$ for the linear and logistic regression models.

## B.5 Additional experiments

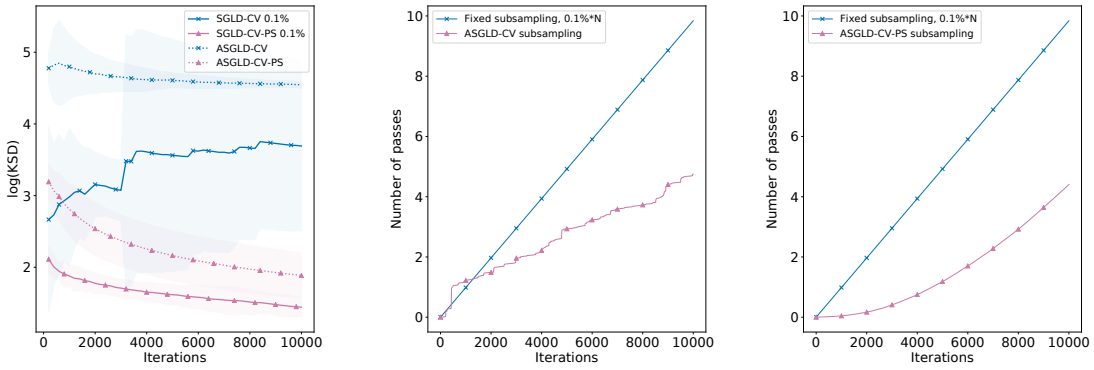
### B.5.1 Performance comparison of SGLD and SGLD-PS




(a) (b)

Figure B.5.1: Sampler performance of SGLD and SGLD-PS for 1%, 5% and 10% subsample sizes over 500 passes through the data. (a) bivariate Gaussian model on synthetic data of size $N = 10^4$ (y-axis: KL divergence); (b) logistic regression on the covertype data (y-axis: KSD).

### B.5.2 Performance of adaptive subsampling




(a) (b) (c)

Figure B.5.2: The linear regression model fitted on the CASP data. (a) KSD comparison of SGLD-CV, SGLD-CV-PS, ASGLD-CV and ASGLD-CV-PS over $10^4$ iterations; (b) the number of passes through the data achieved by fixed subsampling versus ASGLD-CV; (c) the number of passes through the data achieved by fixed subsampling versus ASGLD-CV-PS.

# Bibliography

Raja Hafiz Affandi, Alex Kulesza, Emily B. Fox, and Ben Taskar. Nystrom Approximation for Large-Scale Determinantal Processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 85–98. PMLR, 2013.

Sungjin Ahn, Babak Shahbaba, and Max Welling. Distributed Stochastic Gradient MCMC. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1044–1052. PMLR, 2014.

Christopher Aicher, Yi-An Ma, Nicholas J. Foti, and Emily B. Fox. Stochastic Gradient MCMC for State Space Models. *SIAM Journal on Mathematics of Data Science*, 1 (3):555–587, 2019.

Nima Anari, Shayan Oveis Gharan, and Alireza Rezaei. Monte Carlo Markov Chain Algorithms for Sampling Strongly Rayleigh Distributions and Determinantal Point Processes. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 103–115. PMLR, 2016.

Christophe Andrieu and G. O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, 2009.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 72(3):269–342, 05 2010.

M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

Jack Baker, Paul Fearnhead, Emily B. Fox, and Christopher Nemeth. Large-scale Stochastic Sampling from the Probability Simplex. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Jack Baker, Paul Fearnhead, Emily B. Fox, and Christopher Nemeth. Control variates for stochastic gradient MCMC. *Statistics and Computing*, 29(3):599–615, 2019a.

Jack Baker, Paul Fearnhead, Emily B. Fox, and Christopher Nemeth. sgmcmc: An R Package for Stochastic Gradient Markov Chain Monte Carlo. *Journal of Statistical Software*, 91(3):1–27, 2019b.

Rémi Bardenet, Arnaud Doucet, and Chris Holmes. On Markov chain Monte Carlo methods for tall data. *The Journal of Machine Learning Research*, 18(1):1515–1557, 2017.

Rémi Bardenet, Arnaud Doucet, and Chris Holmes. Towards scaling up Markov chain Monte carlo: an adaptive subsampling approach. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 405–413. PMLR, 2014.

Joris Bierkens, Paul Fearnhead, and G. O. Roberts. The Zig-Zag process and super-efficient sampling for Bayesian analysis of big data. *The Annals of Statistics*, 47(3): 1288 – 1320, 2019.

Jock A Blackard and Denis J Dean. Comparative Accuracies of Neural Networks and Discriminant Analysis in Predicting Forest Cover Types from Cartographic Variables. In *Second Southern Forestry GIS Conference*, pages 189–199, 1998.

Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

Alexei Borodin and Eric M Rains. Eynard–Mehta theorem, Schur process, and their Pfaffian analogs. *Journal of Statistical Physics*, 121:291–317, 2005.

Agnieszka Borowska and Ruth King. Semi-Complete Data Augmentation for Efficient State Space Model Fitting. *Journal of Computational and Graphical Statistics*, 32 (1):19–35, 2023.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization Methods for Large-Scale Machine Learning. *SIAM Review*, 60(2):223–311, 2018.

Alexandre Bouchard-Côté, Sebastian J Vollmer, and Arnaud Doucet. The Bouncy Particle Sampler: A Nonreversible Rejection-Free Markov chain Monte Carlo Method. *Journal of the American Statistical Association*, 113(522):855–867, 2018.

Daniele Calandriello, Michal Derezinski, and Michal Valko. Sampling from a $k$-DPP without looking at all items. In *Advances in Neural Information Processing Systems*, volume 33, pages 6889–6899, 2020.

Olivier Cappé, Eric Moulines, and Tobias Rydén. *Inference in Hidden Markov Models*. Springer Series in Statistics. Springer, 2005.

Ngai Hang Chan and Wilfredo Palma. State space modeling of long-memory processes. *The Annals of Statistics*, 26(2):719–740, 1998.

Niladri S Chatterji, Nicolas Flammarion, Yi-An Ma, Peter L Bartlett, and Michael I Jordan. On the Theory of Variance Reduction for Stochastic Gradient Monte Carlo. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 764–773. PMLR, 2018.

Changyou Chen, Nan Ding, and Lawrence Carin. On the Convergence of Stochastic Gradient MCMC Algorithms with High-Order Integrators. In *Advances in Neural Information Processing Systems*, volume 28, pages 2278–2286, 2015.

Changyou Chen, Wenlin Wang, Yizhe Zhang, Qinliang Su, and Lawrence Carin. A Convergence Analysis for A Class of Practical Variance-Reduction Stochastic Gradient MCMC. *Science China Information Sciences*, 62(12101), 2019.

Tianqi Chen, Emily B. Fox, and Carlos Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1683–1691. PMLR, 2014.

Seokhyun Chung, Cheng-Hao Chou, Xiaozhu Fang, Raed Al Kontar, and Chinedum Okwudire. A Multi-Stage Approach for Knowledge-Guided Predictions With Application to Additive Manufacturing. *IEEE Transactions on Automation Science and Engineering*, 19(3):1675–1687, 2022.

Maria Colombo, Alessio Figalli, and Yash Jhaveri. Lipschitz changes of variables between perturbations of log-concave measures. *Annali Scuola Normale Superiore - Classe Di Scienze*, 17(4):1491–1519, 2017.

Jeremie Coullon and Christopher Nemeth. SGMCMCJax: a lightweight JAX library for stochastic gradient Markov chain Monte Carlo algorithms. *Journal of Open Source Software*, 7(72):4113, 2022.

Dan Crisan and Joaquín Míguez. Nested particle filters for online parameter estimation in discrete-time state-space Markov models. *Bernoulli*, 24(4A):3039–3086, 2018.

Johan Dahlin, Fredrik Lindsten, and Thomas B Schön. Particle Metropolis–Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, 2015.

Arnak S Dalalyan. 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, 2017.

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

Nan Ding, Youhan Fang, Ryan Babbush, Changyou Chen, Robert D Skeel, and Hartmut Neven. Bayesian Sampling Using Stochastic Gradient Thermostats. In

*Advances in Neural Information Processing Systems*, volume 27, pages 3203–3211, 2014.

A. Doucet, M. K. Pitt, G. Deligiannidis, and R. Kohn. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika*, 102(2):295–313, 03 2015.

Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12(3):656–704, 2009.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. An Introduction to Sequential Monte Carlo methods. In *Sequential Monte Carlo Methods in Practice*, pages 3–14. Springer, 2001.

Chao Du, Jun Zhu, and Bo Zhang. Learning Deep Generative Models with Doubly Stochastic Gradient MCMC. *IEEE Transactions on Neural Networks and Learning Systems*, 29(7):3084–3096, 2018.

Kumar Avinava Dubey, Sashank J Reddi, Sinead A Williamson, Barnabas Poczos, Alexander J Smola, and Eric P Xing. Variance Reduction in Stochastic Gradient Langevin Dynamics. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Vanja Dukic, Hedibert F Lopes, and Nicholas G Polson. Tracking Epidemics With Google Flu Trends Data and a State-Space SEIR Model. *Journal of the American Statistical Association*, 107(500):1410–1426, 2012.

Alain Durmus and Éric Moulines. Nonasymptotic convergence analysis for the unadjusted Langevin algorithm. *The Annals of Applied Probability*, 27(3):1551–1587, 2017.

Alain Durmus and Eric Moulines. High-dimensional Bayesian inference via the unadjusted Langevin algorithm. *Bernoulli*, 25(4A):2854–2882, 2019.

Raaz Dwivedi, Yuansi Chen, Martin J Wainwright, and Bin Yu. Log-concave sampling: Metropolis-Hastings algorithms are fast! In *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 793–797. PMLR, 2018.

Paul Fearnhead and Hans R. Künsch. Particle Filters and Data Assimilation. *Annual Review of Statistics and Its Application*, 5:421–449, 2018.

Paul Fearnhead, Joris Bierkens, Murray Pollock, and G. O. Roberts. Piecewise Deterministic Markov Processes for Continuous-Time Monte Carlo. *Statistical Science*, 33(3):386–412, 2018.

Tianfan Fu and Zhihua Zhang. CPSG-MCMC: Clustering-Based Preprocessing method for Stochastic Gradient MCMC. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 841–850. PMLR, 2017.

Guillaume Gautier, Guillermo Polito, Rémi Bardenet, and Michal Valko. Dppy: DPP Sampling with Python. *Journal of Machine Learning Research*, 20(180):1–7, 2019.

Andrew Gelman, John B Carlin, Donald B Rubin, Aki Vehtari, David B Dunson, and Hal S Stern. *Bayesian Data Analysis*. CRC Press, third edition, 2013.

Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

Charles J Geyer. Markov Chain Monte Carlo Lecture Notes. 2005. URL `http://www.stat.umn.edu/geyer/f05/8931/n1998.pdf`.

Walter R Gilks, Sylvia Richardson, and David J Spiegelhalter. Introducing Markov chain Monte Carlo. In *Markov chain Monte Carlo in Practice*, pages 1–19. Chapman & Hall, 1996.

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

Wenbo Gong, Yingzhen Li, and José Miguel Hernández-Lobato. Meta-Learning for Stochastic Gradient MCMC. In *International Conference on Learning Representations*, 2019.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F - Radar and Signal Processing*, 140(2):107–113, 1993.

Jackson Gorham and Lester Mackey. Measuring Sample Quality with Kernels. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1292–1301. PMLR, 2017.

Jackson Gorham, Anant Raj, and Lester Mackey. Stochastic Stein Discrepancies. In *Advances in Neural Information Processing Systems*, volume 33, pages 17931–17942, 2020.

W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

Marcel Hirt and Petros Dellaportas. Scalable Bayesian Learning for State Space Models using Variational Inference with SMC Samplers. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 76–86. PMLR, 2019.

Zaijing Huang and Andrew Gelman. Sampling for Bayesian Computation with Large Datasets. Technical report, Department of Statistics, Columbia University, 2005.

Jonathan Huggins, Trevor Campbell, and Tamara Broderick. Coresets for Scalable Bayesian Logistic Regression. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Pierre E Jacob and Alexandre H Thiery. On non-negative unbiased estimators. *The Annals of Statistics*, 43(2):769–784, 2015.

Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *ASME Journal of Basic Engineering*, 82:35–45, 1960.

N. Kantas, A. Doucet, S.S. Singh, and J.M. Maciejowski. An Overview of Sequential Monte Carlo Methods for Parameter Estimation in General State-Space Models. *IFAC Proceedings Volumes*, 42(10):774–785, 2009. 15th IFAC Symposium on System Identification.

Nikolas Kantas, Arnaud Doucet, Sumeetpal S Singh, Jan Maciejowski, Nicolas Chopin, et al. On Particle Methods for Parameter Estimation in State-Space Models. *Statistical Science*, 30(3):328–351, 2015.

Gregor Kastner. Dealing with Stochastic Volatility in Time Series Using the R Package stochvol. *Journal of Statistical Software*, 69(5):1–30, 2016.

Tamás Kern and A Gyorgy. SVRG++ with Non-uniform Sampling. In *Proceedings of the 9th NIPS Workshop on Optimization for Machine Learning*, 2016.

Rafail Khasminskii. *Stochastic Stability of Differential Equations*, volume 66 of *Stochastic Modelling and Applied Probability*. Springer-Verlag Berlin, 2 edition, 2011.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2016.

Genshiro Kitagawa. Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

Genshiro Kitagawa and Seisho Sato. Monte Carlo Smoothing and Self-Organising State-Space Model. In *Sequential Monte Carlo Methods in Practice*, pages 177–195. Springer New York, 2001.

Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, 2nd corr. print. edition, 1995.

Anoop Korattikara, Yutian Chen, and Max Welling. Austerity in MCMC Land: Cutting the Metropolis-Hastings budget. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 181–189. PMLR, 2014.

Alex Kulesza and Ben Taskar. $k$-DPPs: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1193–1200, 2011.

Alex Kulesza, Ben Taskar, et al. Determinantal Point Processes for Machine Learning. *Foundations and Trends in Machine Learning*, 5(2–3):123–286, 2012.

Claire Launay, Bruno Galerne, and Agnès Desolneux. Exact Sampling of Determinantal Point Processes without Eigendecomposition. *Journal of Applied Probability*, 57(4): 1198–1221, 2020.

Bai Li, Changyou Chen, Hao Liu, and Lawrence Carin. On Connecting Stochastic Gradient MCMC and Differential Privacy. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 557–566. PMLR, 2019.

Chengtao Li, Stefanie Jegelka, and Suvrit Sra. Fast DPP sampling for Nystrom with Application to Kernel Methods. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2061–2070. PMLR, 2016a.

Ruilin Li, Xin Wang, Hongyuan Zha, and Molei Tao. Improving Sampling Accuracy of Stochastic Gradient MCMC Methods via Non-uniform Subsampling of Gradients. *Discrete and Continuous Dynamical Systems - S*, 16(2):329–360, 2021.

Wenzhe Li, Sungjin Ahn, and Max Welling. Scalable MCMC for Mixed Membership Stochastic Blockmodels. In *Proceedings of the 19th International Conference on*

*Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 723–731. PMLR, 2016b.

Jun S. Liu and Rong Chen. Sequential Monte Carlo Methods for Dynamic Systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

Qiang Liu. Importance weighted consensus Monte Carlo for distributed Bayesian inference. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 497–506. AUAI Press, 2016.

Qiang Liu, Jason Lee, and Michael Jordan. A Kernelized Stein Discrepancy for Goodness-of-fit Tests. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 276–284. PMLR, 2016.

Rui Liu, Tianyi Wu, and Barzan Mozafari. Adam with Bandit Sampling for Deep Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5393–5404, 2020.

Samuel Livingstone and Giacomo Zanella. The Barker proposal: Combining robustness and efficiency in gradient-based MCMC. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 84(2):496, 2022.

Yi-An Ma, Tianqi Chen, and Emily B. Fox. A Complete Recipe for Stochastic Gradient MCMC. In *Advances in Neural Information Processing Systems*, volume 28, pages 2917–2925, 2015.

Yi-An Ma, Nicholas J Foti, and Emily B. Fox. Stochastic Gradient MCMC Methods for Hidden Markov Models. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2265–2274. PMLR, 2017.

Yi-An Ma, Emily B. Fox, Tianqi Chen, and Lei Wu. Irreversible samplers from jump and continuous Markov processes. *Statistics and Computing*, 29:177–202, 2019.

Odile Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1):83–122, 1975.

Dougal Maclaurin and Ryan P Adams. Firefly Monte Carlo: exact MCMC with subsets of data. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 4289–4295, 2015.

Chris J Maddison, John Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Teh. Filtering Variational Objectives. In *Advances in Neural Information Processing Systems*, volume 30, pages 6573–6583, 2017.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

Sean P Meyn and Richard L Tweedie. Stability of Markovian processes II: Continuous-time processes and sampled chains. *Advances in Applied Probability*, 25(3):487–517, 1993a.

Sean P Meyn and Richard L Tweedie. Stability of Markovian processes iii: Foster–Lyapunov criteria for continuous-time processes. *Advances in Applied Probability*, 25(3):518–548, 1993b.

S.P. Meyn and R.L. Tweedie. *Markov Chains and Stochastic Stability*. Springer-Verlag, 1993c.

Grigorios Mingas, Leonardo Bottolo, and Christos-Savvas Bouganis. Particle MCMC algorithms and architectures for accelerating inference in state-space models. *International Journal of Approximate Reasoning*, 83:413–433, 2017.

Christian Naesseth, Scott Linderman, Rajesh Ranganath, and David Blei. Variational Sequential Monte Carlo. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 968–977. PMLR, 2018.

Tigran Nagapetyan, Andrew B Duncan, Leonard Hasenclever, Sebastian J Vollmer, Lukasz Szpruch, and Konstantinos Zygalakis. The true cost of stochastic gradient langevin dynamics. *arXiv preprint arXiv:1706.02692*, 2017.

Radford M Neal. MCMC using Hamiltonian Dynamics. In *Handbook of Markov Chain Monte Carlo*, chapter 5. Chapman and Hall/CRC, 2011.

Willie Neiswanger, Chong Wang, and Eric P Xing. Asymptotically exact, embarrassingly parallel MCMC. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, pages 623–632. AUAI Press, 2014.

Christopher Nemeth and Paul Fearnhead. Stochastic gradient Markov chain Monte Carlo. *Journal of the American Statistical Association*, 116(533):433–450, 2021.

Christopher Nemeth and Chris Sherlock. Merging MCMC Subposteriors through Gaussian-Process Approximations. *Bayesian Analysis*, 13(2):507 – 530, 2018.

Christopher Nemeth, Paul Fearnhead, and Lyudmila Mihaylova. Particle Approximations of the Score and Observed Information Matrix for Parameter Estimation in State–Space Models With Linear Computational Cost. *Journal of Computational and Graphical Statistics*, 25(4):1138–1157, 2016.

Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Springer, 5th edition, 2003.

Jimmy Olsson and Johan Westerborn. Efficient particle-based online smoothing in general hidden Markov models: The PaRIS algorithm. *Bernoulli*, 23(3):1951–1996, 2017.

Rihui Ou, Alexander L Young, and David B Dunson. Clustering-Enhanced Stochastic Gradient MCMC for Hidden Markov Models with Rare States. *arXiv preprint arXiv:1810.13431*, 2018.

Giorgio Parisi. Correlation functions and computer simulations. *Nuclear Physics B*, 180(3):378–384, 1981.

Sam Patterson and Yee Whye Teh. Stochastic Gradient Riemannian Langevin dynamics on the Probability Simplex. In *Advances in Neural Information Processing Systems*, volume 26, pages 3102–3110, 2013.

Taylor L Patti, Omar Shehab, Khadijeh Najafi, and Susanne F Yelin. Markov chain Monte Carlo enhanced variational quantum algorithms. *Quantum Science and Technology*, 8(1):015019, 2022.

George Poyiadjis, Arnaud Doucet, and Sumeetpal S Singh. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80, 2011.

Matias Quiroz, Mattias Villani, Robert Kohn, Minh-Ngoc Tran, and Khue-Dung Dang. Subsampling MCMC: An introduction for the survey statistician. *Sankhya A*, 80: 33–69, 2018.

Matias Quiroz, Robert Kohn, Mattias Villani, and Minh-Ngoc Tran. Speeding Up MCMC by Efficient Data Subsampling. *Journal of the American Statistical Association*, 114(526):831–843, 2019.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

Christian Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Science & Business Media, 2nd edition, 2004.

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

G. O. Roberts and Jeffrey S Rosenthal. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1:20–71, 2004.

G. O. Roberts and Richard L Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363, 1996.

G. O. Roberts, A. Gelman, and W. R. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1): 110–120, 02 1997.

Farnood Salehi, L Elisa Celis, and Patrick Thiran. Stochastic Optimization with Bandit Sampling. *arXiv preprint arXiv:1708.02544*, 2017.

Adrien Saumard and Jon A Wellner. Log-concavity and strong log-concavity: A review. *Statistics Surveys*, 8:45–114, 2014.

Mark Schmidt, Reza Babanezhad, Mohamed Ahmed, Aaron Defazio, Ann Clifton, and Anoop Sarkar. Non-Uniform Stochastic Average Gradient Method for Training Conditional Random Fields. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 819–828. PMLR, 2015.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing Finite Sums with the Stochastic Average Gradient. *Mathematical Programming*, 162:83–112, 2017.

Steven L Scott, Alexander W Blocker, Fernando V Bonassi, Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayes and Big Data: The Consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management*, 11(2):78–88, 2016.

Daniel K. Sewell and Yuguo Chen. Latent Space Models for Dynamic Networks. *Journal of the American Statistical Association*, 110(512):1646–1657, 2015.

Neil Shephard. *Stochastic Volatility: Selected Readings*. Oxford University Press, 2005.

Martin A. Tanner and Wing Hung Wong. The Calculation of Posterior Distributions by Data Augmentation. *Journal of the American Statistical Association*, 82(398): 528–540, 1987.

Yee Whye Teh, Alexandre H Thiery, and Sebastian J Vollmer. Consistency and Fluctuations For Stochastic Gradient Langevin Dynamics. *Journal of Machine Learning Research*, 17(7):1–33, 2016.

David A van Dyk and Xiao-Li Meng. The Art of Data Augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

Cédric Villani. *Optimal Transport: Old and New*, volume 338 of *A Series of Comprehensive Studies in Mathematics*. Springer Science & Business Media, first edition, 2008.

Sebastian J. Vollmer, Konstantinos C. Zygalakis, and Yee Whye Teh. Exploration of the (Non-)Asymptotic Bias and Variance of Stochastic Gradient Langevin Dynamics. *Journal of Machine Learning Research*, 17(159):1–48, 2016.

John von Neumann. Various Techniques Used in Connection With Random Digits. *Applied Math Series*, 12(36-38):3, 1951.

Callum Vyner, Christopher Nemeth, and Chris Sherlock. SwISS: A scalable Markov chain Monte Carlo divide-and-conquer strategy. *Stat*, 12(1):e523, 2022.

Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, pages 681–688, 2011.

Sinan Yıldırım, Christophe Andrieu, and Arnaud Doucet. Scalable Monte Carlo inference for state-space models. *arXiv preprint arXiv:1809.02527*, 2018.

Cheng Zhang, Hedvig Kjellstrom, and Stephan Mandt. Determinantal point processes for mini-batch diversification. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017a.

Cheng Zhang, Cengiz Öztireli, and Stephan Mandt. Diversified Mini-Batch Sampling using Repulsive Point Processes. In *Symposium on Advances in Approximate Bayesian Inference*, 2017b.

Peilin Zhao and Tong Zhang. Accelerating Minibatch Stochastic Gradient Descent using Stratified Sampling. *arXiv preprint arXiv:1405.3080*, 2014a.

Peilin Zhao and Tong Zhang. Stochastic Optimization with Importance Sampling. *arXiv preprint arXiv:1401.2753*, 2014b.

Peilin Zhao and Tong Zhang. Stochastic Optimization with Importance Sampling for Regularized Loss Minimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1–9. PMLR, 2015.