Foundations and Trends® in Machine Learning
Vol. 9, No. 2-3 (2016) 119–247

DOI: 10.1561/2200000052

Patterns of Scalable Bayesian Inference

Elaine Angelino*
UC Berkeley
elaine@eecs.berkeley.edu

Matthew James Johnson*
Harvard University
mattjj@csail.mit.edu

Ryan P. Adams
Harvard University and Twitter
rpa@seas.harvard.edu

* Authors contributed equally.

Contents

Abstract

Datasets are growing not just in size but in complexity, creating a demand for rich models and quantification of uncertainty. Bayesian methods are an excellent fit for this demand, but scaling Bayesian inference is a challenge. In response to this challenge, there has been considerable recent work based on varying assumptions about model structure, underlying computational resources, and the importance of asymptotic correctness. As a result, there is a zoo of ideas with a wide range of assumptions and applicability.

In this paper, we seek to identify unifying principles, patterns, and intuitions for scaling Bayesian inference. We review existing work on utilizing modern computing resources with both MCMC and variational approximation techniques. From this taxonomy of ideas, we characterize the general principles that have proven successful for designing scalable inference procedures and comment on the path forward.

E. Angelino, M. J. Johnson, and R. P. Adams. *Patterns of Scalable Bayesian Inference*. Foundations and Trends® in Machine Learning, vol. 9, no. 2-3, pp. 119–247, 2016.
DOI: 10.1561/2200000052.

1

Introduction

We have entered a new era of scientific discovery, in which computational insights are being integrated with large-scale statistical data analysis to enable researchers to ask both grander and more subtle questions about our natural world. This viewpoint asserts that we need not be limited to the narrow hypotheses that can be framed by traditional small-scale analysis techniques. Supporting new kinds of data-driven queries, however, requires that new methods be developed for statistical inference that can *scale up* along multiple axes — more samples, more dimensions, and greater model complexity — as well as *scale out* by taking advantage of modern parallel compute environments.

There are a variety of methodological frameworks for statistical inference; here we are concerned with the Bayesian formalism. In the Bayesian setting, inference queries are framed as interrogations of a posterior distribution over parameters, missing data, and other unknowns. By treating these unobserved quantities as random variables and conditioning on observed data, the Bayesian aims to make inferences and quantify uncertainty in a way that can coherently incorporate new data and other sources of information.

Coherently managing probabilistic uncertainty is central to Bayesian analysis, and so the computations associated with most inference tasks — estimation, prediction, hypothesis testing — are typically integrations. In some special situations it is possible to perform such integrations exactly, for example by taking advantage of tractable prior distributions and conjugacy in the prior-likelihood pair, or by using dynamic programming when the dependencies between random variables are relatively simple. Unfortunately, many inference problems are not amenable to these exact integration procedures, and so most of the interest in Bayesian computation focuses on methods of approximate inference.

There are two dominant paradigms for approximate inference in Bayesian models: Monte Carlo sampling methods and variational approximations. The Monte Carlo approach observes that integrations performed to query posterior distributions can be framed as expectations, and thus estimated with samples; such samples are most often generated via simulation from carefully designed Markov chains. Variational inference instead seeks to compute these integrals by approximating the posterior distribution with a more tractable alternative, finding the best approximation with powerful optimization algorithms.

In this paper, we examine how these techniques can be scaled up to larger problems and scaled out across parallel computational resources. This is not an exhaustive survey of a rapidly-evolving area of research; rather, we seek to identify the main ideas and themes that are emerging in this area, and articulate what we believe are some of the significant open questions and challenges.

1.1 Why be Bayesian with big data?

The Bayesian paradigm is fundamentally about integration: integration computes posterior estimates and measures of uncertainty, eliminates nuisance variables or missing data, and averages models to compute predictions or perform model comparison. While some statistical methods, such as MAP estimation, can be described from a Bayesian perspective, in which case the prior serves simply as a regularizer in an optimization problem, such methods are not inherently or exclusively Bayesian. Posterior integration is the distinguishing characteristic of Bayesian statistics, and so a defense of Bayesian ideas in the big data regime rests on the utility of integration.

The big data setting might seem to be precisely where integration isn't so important: as the dataset grows, shouldn't the posterior distribution concentrate towards a point mass? If big data means we end up making predictions using concentrated posteriors, why not focus on point estimation and avoid the specification of priors and the burden of approximate integration? These objections certainly apply to settings where the number of parameters is small and fixed ("tall data"). However, many models of interest have many parameters ("wide data"), or indeed have a number of parameters that grows along with the amount of data.

For example, an Internet company making inferences about its users' viewing and buying habits may have terabytes of data in total but only a few observations for its newest customers, the ones most important to impress with personalized recommendations. Moreover, it may wish to adapt its model in an online way as data arrive, a task that benefits from calibrated posterior uncertainties [Stern et al., 2009]. As another example, consider a healthcare company. As its dataset grows, it might hope to make more detailed and complex inferences about populations while also making careful predictions with calibrated uncertainty for each patient, even in the presence of massive missing data [Lawrence, 2015]. These scaling issues also arise in astronomy, where hundreds of billions of light sources, such as stars, galaxies, and quasars, each have latent variables that must be estimated from very weak observations, and are coupled in a large hierarchical model [Regier et al., 2015]. In Microsoft Bing's sponsored search advertising, predictive probabilities inform the pricing in the keyword auction mechanism. This problem nevertheless must be solved at scale, with tens of millions of impressions per hour [Graepel et al., 2010].

These are the regimes where big data can be small [Lawrence, 2015] and the number and complexity of statistical hypotheses grows with the data. The Bayesian inference methods we survey in this paper may provide solutions to these challenges.

1.2 The accuracy of approximate integration

Bayesian inference may be important in some modern big data regimes, but exact integration in general is computationally out of reach. While decades of research in Bayesian inference in both statistics and machine learning have produced many powerful approximate inference algorithms, the big data setting poses some new challenges. Iterative algorithms that read the entire dataset before making each update become prohibitively expensive. Sequential computation is at a significant and growing disadvantage compared to computation that can leverage parallel and distributed computing resources. Insisting on zero asymptotic bias from Monte Carlo estimates of expectations may leave us swamped in errors from high variance [Korattikara et al., 2014] or transient bias.

These challenges, and the tradeoffs that may be necessary to address them, can be viewed in terms of how accurate the integration in our approximate inference algorithms must be. Markov chain Monte Carlo (MCMC) algorithms that admit the exact posterior as a stationary distribution may be the gold standard for generically estimating posterior expectations, but if standard MCMC algorithms become intractable in the big data regime we must find alternatives and understand their tradeoffs. Indeed, someone using Bayesian methods for machine learning may be less constrained than a classical Bayesian statistician: if the ultimate goal is to form predictions that perform well according to a specific loss function, computational gains at the expense of the internal posterior representation may be worthwhile. The methods studied here cover a range of such approximate integration tradeoffs.

1.3 Outline

The remainder of this review is organized as five chapters. In Chapter 2, we provide relevant background material on exponential families, MCMC inference, mean field variational inference, and stochastic gradient optimization. The next three chapters survey recent algorithmic ideas for scaling Bayesian inference, highlighting theoretical results where possible. Each of these central technical chapters ends with a summary and discussion, identifying emergent themes and patterns as well as open questions. Chapters 3 and 4 focus on MCMC algorithms, which are inherently serial and often slow to converge; the algorithms in the first of these use various forms of data subsampling to scale up serial MCMC and in the second use a diverse array of strategies to scale out on parallel resources. In Chapter 5 we discuss two recent techniques for scaling variational mean field algorithms. Both process data in minibatches: the first applies stochastic gradient optimization methods and the second is based on incremental posterior updating. Finally, in Chapter 6 we provide an overarching discussion of the ideas we survey, focusing on challenges and open questions in large-scale Bayesian inference.

2

Background

In this chapter we summarize background material on which the ideas in subsequent chapters are based. This chapter also serves to fix some common notation. Throughout the chapter, we mostly avoid measure-theoretic definitions and instead assume that any density exists with respect to either Lebesgue measure or counting measure, depending on its context.

First, we cover some relevant aspects of exponential families. Second, we cover the foundations of Markov chain Monte Carlo (MCMC) algorithms, which are the workhorses of Bayesian statistics and are common in Bayesian machine learning. Indeed, the algorithms discussed in Chapters 3 and 4 either are MCMC algorithms or aim to approximate MCMC algorithms. Next, we describe the basics of mean field variational inference, expectation propagation, and stochastic gradient optimization, which are used extensively in Chapter 5.

2.1 Exponential families

Exponential families of densities play a key role in Bayesian analysis and many practical Bayesian methods. In particular, likelihoods that are exponential families yield natural conjugate prior families, which can provide analytical and computational advantages in both MCMC and variational inference algorithms. These ideas are useful not only to perform exact inference in some restricted settings, but also to build approximate inference algorithms for more general models. Exponential families are also particularly relevant in the context of large datasets: in a precise sense, among all families of densities in which the support does not depend on the parameter, exponential families are the only families that admit a finite-dimensional sufficient statistic. Thus only exponential families allow arbitrarily large amounts of data to be summarized with a fixed-size description.

In this section we give basic definitions, notation, and results concerning exponential families. For additional perspectives from convex analysis see Wainwright and Jordan [2008], and for perspectives from differential geometry see Amari and Nagaoka [2007].

Throughout this section we take all densities to be absolutely continuous with respect to the appropriate Lebesgue measure (when the underlying set $\mathcal{X}$ is Euclidean space) or counting measure (when $\mathcal{X}$ is discrete), and denote the Borel σ-algebra of a set $\mathcal{X}$ as $\mathcal{B}(\mathcal{X})$ (generated by Euclidean and discrete topologies, respectively). We assume measurability of all functions as necessary.

Given a statistic function $t_x : \mathcal{X} \to \mathbb{R}^n$ and a base measure $\nu_{\mathcal{X}}$, we can define an exponential family of probability densities on $\mathcal{X}$ relative to $\nu_{\mathcal{X}}$ and indexed by natural parameter $\eta_x \in \mathbb{R}^n$ by

$$p(x \,|\, \eta_x) \propto \exp\left\{\langle \eta_x,\, t_x(x)\rangle\right\}, \quad \forall \eta_x \in \mathbb{R}^n, \tag{2.1}$$

where $\langle \cdot,\, \cdot \rangle$ is the standard inner product on $\mathbb{R}^n$. We also define the partition function as

$$Z_x(\eta_x) \triangleq \int \exp\left\{\langle \eta_x,\, t_x(x)\rangle\right\} \nu_{\mathcal{X}}(dx) \tag{2.2}$$

and define $H \subseteq \mathbb{R}^n$ to be the set of all normalizable natural parameters,

$$H \triangleq \left\{\eta \in \mathbb{R}^n : Z_x(\eta) < \infty\right\}. \tag{2.3}$$

We say that an exponential family is *regular* if H is open. We can write the normalized probability density as

$$p(x \,|\, \eta) = \exp\left\{\langle \eta_x,\, t_x(x)\rangle - \log Z_x(\eta_x)\right\}. \tag{2.4}$$

Finally, when we parameterize the family with some other coordinates θ, we write the natural parameter as a continuous function $\eta_x(\theta)$ and write the density as

$$p(x \,|\, \theta) = \exp\left\{\langle \eta_x(\theta),\, t_x(x)\rangle - \log Z_x(\eta_x(\theta))\right\} \tag{2.5}$$

and take $\Theta = \eta_x^{-1}(H)$ to be the open set of parameters that correspond to normalizable densities. We summarize this notation in the following definition.

Definition 2.1 (Exponential family of densities)**.** Given a measure space $(\mathcal{X}, \mathcal{B}(\mathcal{X}), \nu_{\mathcal{X}})$, a statistic function $t_x : \mathcal{X} \to \mathbb{R}^n$, and a natural parameter function $\eta_x : \Theta \to \mathbb{R}^n$, the corresponding *exponential family of densities* relative to $\nu_{\mathcal{X}}$ is

$$p(x \,|\, \theta) = \exp\left\{\langle \eta_x(\theta),\, t_x(x)\rangle - \log Z_x(\eta_x(\theta))\right\}, \tag{2.6}$$

where

$$\log Z_x(\eta_x) \triangleq \log \int \exp\left\{\langle \eta_x,\, t_x(x)\rangle\right\} \nu_{\mathcal{X}}(dx) \tag{2.7}$$

is the log partition function.

When we write exponential families of densities for different random variables, we change the subscripts on the statistic function, natural parameter function, and log partition function to correspond to the symbol used for the random variable. When the corresponding random variable is clear from context, we drop the subscripts to simplify notation.

The statistic $t_x(x)$ is *sufficient* in the sense of the Fisher-Neyman Factorization [Keener, 2010, Theorem 3.6]. By construction,

$$p(x|\theta) \propto \exp\{\langle \eta(\theta), t_x(x)\},$$

and hence $t_x(x)$ contains all the information about x that is relevant for the parameter θ. The Koopman-Pitman-Darmois Theorem shows that among all families in which the support does not depend on the parameter, exponential families are the only families which provide this powerful summarization property, under some mild smoothness conditions [Hipp, 1974].

Exponential families have many convenient analytical and computational properties. In particular, the next proposition shows that the log partition function of an exponential family generates cumulants of the statistic.

Proposition 2.1 (Gradients of $\log Z$ and expected statistics)**.** The gradient of the log partition function of an exponential family gives the expected sufficient statistic,

$$\nabla \log Z(\eta) = \mathbb{E}_{p(x \mid \eta)} \left[t(x) \right], \tag{2.8}$$

where the expectation is over the random variable x with density $p(x \mid \eta)$. More generally, the moment generating function of $t(x)$ can be written

$$\mathrm{M}_{t(x)}(s) \triangleq \mathbb{E}_{p(x \mid \eta)} \left[e^{\langle s, t(x) \rangle} \right] = e^{\log Z(\eta + s) - \log Z(\eta)} \tag{2.9}$$

and so derivatives of $\log Z$ give cumulants of $t(x)$, where the first cumulant is the mean and the second and third cumulants are the second and third central moments, respectively.

For members of an exponential family, many quantities can be expressed generically in terms of the natural parameter, expected statistics under that parameter, and the log partition function. In particular, using a natural parameterization the Fisher information matrix of an exponential family can be computed as the Hessian matrix of its log partition function, as we summarize next.

Definition 2.2 (Score vector and Fisher information matrix)**.** Given a family of densities $p(x \mid \theta)$ indexed by a parameter θ, the *score vector* $v(x, \theta)$ is the gradient of the log density with respect to the parameter,

$$v(x, \theta) \triangleq \nabla_\theta \log p(x \mid \theta), \tag{2.10}$$

and the *Fisher information matrix* is the covariance of the score,

$$I(\theta) \triangleq \mathrm{Cov}\left[v(x, \theta) \right] = \mathbb{E}\left[v(x, \theta) v(x, \theta)^{\mathsf{T}} \right], \tag{2.11}$$

where the expectation is taken over the random variable x with density $p(x \mid \theta)$, and where we have used the identity $\mathbb{E}[v(x, \theta)] = 0$.

Proposition 2.2 (Score and Fisher information for exponential families)**.** Given an exponential family of densities $p(x \,|\, \eta)$ indexed by the natural parameter η, as in Eq. (2.4), the score with respect to the natural parameter is given by

$$v(x, \eta) = \nabla_\eta \log p(x \,|\, \eta) = t(x) - \nabla \log Z(\eta) \tag{2.12}$$

and the Fisher information matrix is given by

$$I(\eta) = \nabla^2 \log Z(\eta). \tag{2.13}$$

Finally, we summarize conjugacy properties of exponential families that are particularly useful for Bayesian inference. Given an exponential family of densities on $\mathcal{X}$ as in Definition 2.1, we can define a related exponential family of densities on Θ by defining a statistic function $t_\theta(\theta)$ in terms of the functions $\eta_x(\theta)$ and $\log Z_x(\eta_x(\theta))$.

Definition 2.3 (Natural exponential family conjugate prior)**.** Given the exponential family $p(x \,|\, \theta)$ of Definition 2.1, define the statistic function $t_\theta : \Theta \to \mathbb{R}^{n+1}$ as the concatenation

$$t_\theta(\theta) \triangleq \left(\eta_x(\theta), -\log Z_x(\eta_x(\theta))\right), \tag{2.14}$$

where the first n coordinates of $t_\theta(\theta)$ are given by $\eta_x(\theta)$ and the last coordinate is given by $-\log Z_x(\eta_x(\theta))$. We call the exponential family with statistic $t_\theta(\theta)$ the *natural exponential family conjugate prior* to the density $p(x \,|\, \theta)$ and write the density as

$$p(\theta) = \exp\left\{\langle \eta_\theta,\, t_\theta(\theta)\rangle - \log Z_\theta(\eta_\theta)\right\}, \tag{2.15}$$

where $\eta_\theta \in \mathbb{R}^{n+1}$ and the density is taken relative to some measure ν_Θ on $(\Theta, \mathcal{B}(\Theta))$.

Notice that using $t_\theta(\theta)$ we can rewrite the original density $p(x \,|\, \theta)$,

$$\begin{aligned} p(x \,|\, \theta) &= \exp\left\{\langle \eta_x(\theta),\, t_x(x)\rangle - \log Z_x(\eta_x(\theta))\right\} && (2.16) \\ &= \exp\left\{\langle t_\theta(\theta),\, (t_x(x), 1)\rangle\right\}. && (2.17) \end{aligned}$$

This relationship is useful in Bayesian inference: when the exponential family $p(x \,|\, \theta)$ is a likelihood function and the family $p(\theta)$ is used as a prior, the pair enjoy a convenient conjugacy property, as summarized in the next proposition.

Proposition 2.3 (Conjugacy). Let the densities $p(x \,|\, \theta)$ and $p(\theta)$ be defined as in Definitions 2.1 and 2.3, respectively. We have the relations

$$p(\theta, x) = \exp\left\{\langle \eta_\theta + (t_x(x), 1),\, t_\theta(\theta)\rangle - \log Z_\theta(\eta_\theta)\right\} \tag{2.18}$$

$$p(\theta \,|\, x) = \exp\left\{\langle \eta_\theta + (t_x(x), 1),\, t_\theta(\theta)\rangle - \log Z_\theta(\eta_\theta + (t_x(x), 1))\right\} \tag{2.19}$$

and hence in particular the posterior $p(\theta \,|\, x)$ is in the same exponential family as $p(\theta)$ with the natural parameter $\eta_\theta + (t_x(x), 1)$. Similarly, with multiple likelihood terms $p(x_i \,|\, \theta)$ for $i = 1, 2, \ldots, N$ we have

$$p(\theta) \prod_{i=1}^{N} p(x_i \,|\, \theta) = \exp\left\{\langle \eta_\theta + \sum_{i=1}^{N} (t_x(x_i), 1),\, t_\theta(\theta)\rangle - \log Z_\theta(\eta_\theta)\right\}. \tag{2.20}$$

Conjugate pairs are particularly useful in Bayesian analysis because as we observe data the posterior remains in the same family as the prior, with a parameter that is easy to compute in terms of sufficient statistics. In particular, if inference in the prior family is tractable, then inference in the posterior is also tractable.

2.2 Markov Chain Monte Carlo inference

Markov chain Monte Carlo (MCMC) is a class of algorithms for estimating expectations with respect to intractable probability distributions, such as most posterior distributions arising in Bayesian inference. Given a target distribution, a standard MCMC algorithm proceeds by simulating an ergodic random walk that admits the target distribution as its stationary distribution. As we develop in the following subsections, by collecting samples from the simulated trajectory and forming Monte Carlo estimates, expectations of many functions can be approximated to arbitrary accuracy. Thus MCMC is employed when samples or expectations from a distribution cannot be obtained directly, as is often the case with complex, high-dimensional systems arising across disciplines.

In this section, we first review the two underlying ideas behind MCMC algorithms: Monte Carlo methods and Markov chains. First we define the bias and variance of estimators. Next, we introduce Monte Carlo estimators based on independent and identically distributed samples. We then describe how Monte Carlo estimates can be formed using mutually dependent samples generated by a Markov chain simulation. Finally, we introduce two general MCMC algorithms commonly applied to Bayesian posterior inference, the Metropolis-Hastings and Gibbs sampling algorithms. Our exposition here mostly follows the standard treatment, such as in Brooks et al. [2011, Chapter 1], Geyer [1992], and Robert and Casella [2004].

2.2.1 Bias and variance of estimators

Notions of bias and variance are fundamental to understanding and comparing estimator performance, and much of our discussion of MCMC methods is framed in these terms.

Consider using a scalar-valued random variable $\hat{\theta}$ to estimate a fixed scalar quantity of interest θ. The bias and variance of the estimator $\hat{\theta}$ are defined as

$$\mathrm{Bias}[\hat{\theta}] = \mathbb{E}[\hat{\theta}] - \theta \tag{2.21}$$

$$\mathrm{Var}[\hat{\theta}] = \mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^2]. \tag{2.22}$$

The mean squared error $\mathbb{E}[(\hat{\theta} - \theta)^2]$ can be decomposed in terms of the variance and the square of the bias:

$$\begin{aligned}\mathbb{E}[(\hat{\theta} - \theta)^2] &= \mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}] + \mathbb{E}[\hat{\theta}] - \theta)^2] & (2.23)\\ &= \mathbb{E}[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^2] + (\mathbb{E}[\hat{\theta}] - \theta)^2 & (2.24)\\ &= \mathrm{Var}[\hat{\theta}] + \mathrm{Bias}[\hat{\theta}]^2 & (2.25)\end{aligned}$$

This decomposition provides a basic language for evaluating estimators and thinking about tradeoffs. Among unbiased estimators, those with lower variance are generally preferrable. However, when an unbiased estimator has high variance, a biased estimator that achieves low variance can have a lower overall mean squared error.

As we describe in the following sections, a substantial amount of the study of Bayesian statistical computation has focused on algorithms that produce asymptotically unbiased estimates of posterior expectations, in which the bias due to initialization is transient and is washed out relatively quickly. In this setting, the error is typically considered to be dominated by the variance term, which can be made as small as desired by increasing computation time without bound. When computation becomes expensive as in the big data setting, errors under a realistic computational budget may in fact be dominated by variance, as observed by Korattikara et al. [2014], or, as we argue in Chapter 6, transient bias. Several of the new algorithms we examine in Chapters 3 and 4 aim to adjust this tradeoff by allowing some asymptotic bias while effectively reducing the variance and transient bias contributions through more efficient computation.

2.2.2 Monte Carlo estimates from independent samples

Let X be a random variable with finite expectation $\mathbb{E}[X] = \mu$, and let $(X_i : i \in \mathbb{N})$ be a sequence of i.i.d. random variables each with the same distribution as X. The Strong Law of Large Numbers (LLN) states that the sample average converges almost surely to the expectation μ as $n \to \infty$:

$$\mathbb{P}\left(\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} X_i = \mu\right) = 1. \tag{2.26}$$

This convergence immediately suggests the Monte Carlo method: to approximate the expectation of X, which to compute exactly may involve an intractable integral, one can use i.i.d. samples and compute a sample average. In addition, because for any measurable function f the sequence $(f(X_i) : i \in \mathbb{N})$ is also a sequence of i.i.d. random variables, we can form the Monte Carlo estimate

$$\mathbb{E}[f(X)] \approx \frac{1}{n}\sum_{i=1}^{n} f(X_i). \tag{2.27}$$

Monte Carlo estimates of this form are unbiased by construction, and so the quality of a Monte Carlo estimate can be evaluated in terms of its variance as a function of the number of samples n, which in turn can be understood with the Central Limit Theorem (CLT), at least in the asymptotic regime. Indeed, the CLT provides not only the asymptotic scaling of the error but also describes the asymptotic distribution of those errors. If X is real-valued and has finite variance $\mathbb{E}[(X-\mu)^2] = \sigma^2 < \infty$, then the CLT states that the deviation $\frac{1}{n}\sum_{i=1}^{n} X_i - \mu$, rescaled appropriately, converges in distribution and is asymptotically normal:

$$\lim_{n\to\infty} \mathbb{P}\left(\frac{1}{\sqrt{n}}\sum_{i=1}^{n}(X_i-\mu) < \alpha\right) = \mathbb{P}(Z < \alpha) \tag{2.28}$$

where $Z \sim \mathcal{N}(0,\sigma^2)$. In particular, as n grows, the standard deviation of the sample average $\frac{1}{n}\sum_{i=1}^{n} X_i - \mu$ converges to zero at an asymptotic rate proportional to $\frac{1}{\sqrt{n}}$. More generally, for any real-valued measurable function f, the Monte Carlo standard error (MCSE) in the estimate (2.27) asymptotically scales as $\frac{1}{\sqrt{n}}$.

Monte Carlo estimators effectively reduce the problem of computing expectations to the problem of generating samples. However, the preceding statements require the samples used in the Monte Carlo estimate to be independent, and independent samples can be computationally difficult to generate. Instead of relying on independent samples, Markov chain Monte Carlo algorithms compute estimates using mutually dependent samples generated by simulating a Markov chain.

2.2.3 Markov chains

Let $\mathcal{X}$ be a discrete or continuous state space and let $x, x' \in \mathcal{X}$ denote states. A time-homogeneous Markov chain is a discrete-time stochastic process $(X_t : t \in \mathbb{N})$ governed by a transition operator $T(x \to x')$ that specifies the probability density of transitioning to a state x' from a given state x:

$$\mathbb{P}(X_{t+1} \in A \,|\, X_t = x) = \int_A T(x \to x')\, dx', \quad \forall t \in \mathbb{N}, \tag{2.29}$$

for all measurable sets A. A Markov chain is memoryless in the sense that given the current state its future behavior is independent of its past history.

Given an initial density $\pi_0(x)$ for X_0, a Markov chain evolves this density from one time point to the next through iterative application of the transition operator. We write the application of the transition operator to a density π_0 to yield a new density π_1 as

$$\pi_1(x') = (T\pi_0)(x') = \int_{\mathcal{X}} T(x \to x')\pi_0(x)\, dx. \tag{2.30}$$

Writing T^t to denote t repeated applications of the transition operator T, the density of X_t induced by π_0 and T is then given by $\pi_t = T^t\pi_0$.

Markov chain simulation follows this iterative definition by iteratively sampling the next state using the current state and the transition operator. That is, after first sampling X_0 from $\pi_0(\,\cdot\,)$, Markov chain simulation proceeds at time step t by sampling X_{t+1} according to the density $T(x_t \to \,\cdot\,)$ induced by the fixed sample x_t.

We are interested in Markov chains that converge in total variation to a unique stationary density $\pi(x)$ in the sense that

$$\lim_{t\to\infty} \|\pi_t - \pi\|_{\mathrm{TV}} = 0 \tag{2.31}$$

for any initial distribution π_0, where $\|\cdot\|_{\mathrm{TV}}$ denotes the total variation norm on densities:

$$\|p - q\|_{\mathrm{TV}} = \frac{1}{2}\int_{\mathcal{X}} |p(x) - q(x)|\, dx. \tag{2.32}$$

The total variation distance is also useful as an error metric for the approximate MCMC we discuss in the sequel. For a transition operator $T(x \to x')$ to admit $\pi(x)$ as a stationary density, its application must leave $\pi(x)$ invariant:

$$\pi = T\pi. \tag{2.33}$$

For a discussion of general conditions that guarantee a Markov chain converges to a unique stationary distribution, i.e., that the chain is ergodic, see Meyn and Tweedie [2009].

In some cases it is easy to show that a transition operator admits a particular stationary distribution. In particular, it is clear that π is a stationary distribution when a transition operator $T(x \to x')$ is *reversible* with respect to π, i.e., it satisfies the detailed balance (reversibility) condition with respect to a density $\pi(x)$,

$$T(x \to x')\pi(x) = T(x' \to x)\pi(x') \quad \forall x, x' \in \mathcal{X}, \tag{2.34}$$

which is a pointwise condition over $\mathcal{X} \times \mathcal{X}$. Integrating over x on both sides gives:

$$\begin{aligned}\int_{\mathcal{X}} T(x \to x')\pi(x)\,dx &= \int_{\mathcal{X}} T(x' \to x)\pi(x')\,dx \\ &= \pi(x') \int_{\mathcal{X}} T(x' \to x)\,dx \\ &= \pi(x'),\end{aligned}$$

which is precisely the required condition from (2.33). We can interpret (2.34) as stating that, for a reversible Markov chain starting from its stationary distribution, any transition $x \to x'$ is equilibrated by the corresponding reverse transition $x' \to x$. Many MCMC methods are based on deriving reversible transition operators.

For a thorough introduction to Markov chains, see Robert and Casella [2004, Chapter 6] and Meyn and Tweedie [2009].

2.2.4 Markov chain Monte Carlo (MCMC)

Markov chain Monte Carlo (MCMC) methods simulate a Markov chain for which the stationary distribution is equal to a target distribution of interest, and use the simulated samples to form Monte Carlo estimates of expectations. That is, consider simulating a Markov chain with unique stationary density $\pi(x)$, as in Section 2.2.3, and collecting its trajectory into a set of samples $\{X_i\}_{i=1}^n$. These collected samples can be used to form a Monte Carlo estimate for a function f of a random variable X with density $\pi(x)$ via

$$\mathbb{E}[f(X)] = \int_{\mathcal{X}} f(x)\pi(x)\;dx \approx \frac{1}{n}\sum_{i=1}^{n} f(X_i). \tag{2.35}$$

Even though this Markov chain Monte Carlo estimate is not constructed from independent samples, under some mild conditions it can asymptotically satisfy analogs of the Law of Large Numbers (LLN) and Central Limit Theorem (CLT) that were used to justify ordinary Monte Carlo methods in Section 2.2.2. We sketch these important results here.

The MCMC analog of the LLN states that for a chain satisfying basic recurrence conditions and admitting an invariant distribution π, for all functions f that are absolutely integrable with respect to π, i.e. all $f : \mathcal{X} \to \mathbb{R}$ that satisfy $\int_{\mathcal{X}} |f(x)|\, \pi(x)\, dx < \infty$, we have

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} f(X_i) = \int_{\mathcal{X}} f(x)\, \pi(x)\, dx \quad \text{(a.s.)}, \tag{2.36}$$

for any initial distribution π_0. This result is the basic motivation to collect samples from a Markov chain trajectory and use those samples to compute estimates of expectations with respect to the invariant distribution π. For a more detailed statement, see Meyn and Tweedie [2009, Section 17.1].

Given that Markov chain Monte Carlo estimators can satisfy a law of large numbers, we'd also like to understand the distribution of estimator errors and how the error distribution changes as we collect more samples. To quantify these errors, the analog of the CLT must take into account both the Markov dependency structure among the samples used in the Monte Carlo estimate and also the initial state in which the chain was started. However, under some additional conditions on both the Markov chain's convergence rate and the function f, the sample average for any initial distribution π_0 is asymptotically normal in distribution (with appropriate scaling):

$$\lim_{n\to\infty} \mathbb{P}\left(\frac{1}{\sqrt{n}} \sum_{n=1}^{n} (f(X_i) - \mu) < \alpha \right) = \mathbb{P}(Z < \alpha), \tag{2.37}$$

$$Z \sim \mathcal{N}\left(0, \sigma^2\right), \tag{2.38}$$

$$\sigma^2 = \mathrm{Var}_\pi[f(X_0)] + 2 \sum_{t=1}^{\infty} \mathrm{Cov}_\pi[f(X_0), f(X_t)] \tag{2.39}$$

where $\mu = \int_{\mathcal{X}} f(x)\, \pi(x)\, dx$ and where Var_π and Cov_π denote the variance and covariance with respect to the stationary distribution π. Thus standard error in the MCMC estimate also scales asymptotically as $\frac{1}{\sqrt{n}}$, with a constant that depends on the autocovariance function of the stationary version of the chain. See Meyn and Tweedie [2009, Chapter 17] and Robert and Casella [2004, Section 6.7] for precise statements of both the LLN and CLT for Markov chain Monte Carlo estimates and for conditions on the Markov chain which guarantee that these theorems hold.

These results show that the asymptotic behavior of MCMC estimates of the form (2.35) is generally comparable to that of ordinary Monte Carlo estimates as discussed in Section 2.2.2. However, in the non-asymptotic regime, MCMC estimates differ from ordinary Monte Carlo estimates in an important respect: there is a transient bias due to initializing the Markov chain out of stationarity. That is, the initial distribution π_0 from which the first iterate is sampled is generally not the chain's stationary distribution π, since if it were then ordinary Monte Carlo could be performed directly. While the marginal distribution of each Markov chain iterate converges to the stationary distribution, the effects of initialization on the initial iterates of the chain contribute an error term to Eq. (2.35) in the form of a transient bias.

This transient bias does not factor into the asymptotic behavior described by the MCMC analogs of the LLN and the CLT; asymptotically, it decreases at a rate of at least $\mathcal{O}(\frac{1}{n})$ and is hence dominated by the Monte Carlo standard error which decreases only at rate $\mathcal{O}(\frac{1}{\sqrt{n}})$. However, its effects can be significant in practice, especially in machine learning. Whenever a sampled chain seems "unmixed" because its iterates are too dependent on the initialization, errors in MCMC estimates are dominated by this transient bias.

The simulation in Figure 2.1 illustrates these error terms in MCMC estimates and how they can behave as more Markov chain samples are collected. The LLN and CLT for MCMC describe the regime on the far right of the plot: the total error can be driven arbitrarily small because the MCMC estimates are asymptotically unbiased, and the total error is asymptotically dominated by the Monte Carlo standard error. However, before reaching the asymptotic regime, the error is often dominated by the transient initialization bias. Several of the new methods we survey can be understood as attempts to alter the traditional MCMC tradeoffs, as we discuss further in Chapter 6.

Transient bias can be traded off against Monte Carlo standard error by choosing different subsets of Markov chain samples in the MCMC estimator. As an extreme choice, instead of using the MCMC estimator (2.35) with the full set of Markov chain samples $\{X_i\}_{i=1}^n$, transient bias can be minimized by forming estimates using only the last Markov

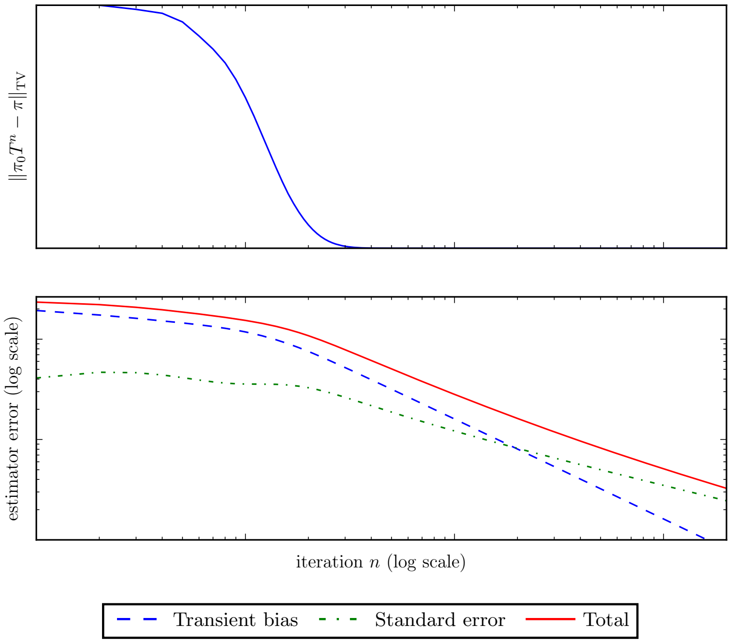


Figure 2.1: A simulation illustrating error terms in MCMC estimator (2.35) as a function of the number of Markov chain iterations (log scale). The marginal distributions of the Markov chain iterates converge to the target distribution (top panel), while the errors in MCMC estimates due to transient bias and Monte Carlo standard error are eventually driven arbitrarily small at rates of $\mathcal{O}(\frac{1}{n})$ and $\mathcal{O}(\frac{1}{\sqrt{n}})$, respectively (bottom panel). The horizontal axis is shared between the two panels.

chain sample:

$$\mathbb{E}[f(X)] \approx f(X_n). \tag{2.40}$$

However, this choice of MCMC estimator maximizes the Monte Carlo standard error, which asymptotically cannot be decreased below the posterior variance of the estimand. A practical choice is to form MCMC estimates using the last $\lceil n/2 \rceil$ Monte Carlo samples, discarding the other samples as warm-up samples, resulting in an estimator

$$\mathbb{E}[f(X)] \approx \frac{1}{\lceil n/2 \rceil} \sum_{i=\lfloor n/2 \rfloor}^{n} f(X_i). \tag{2.41}$$

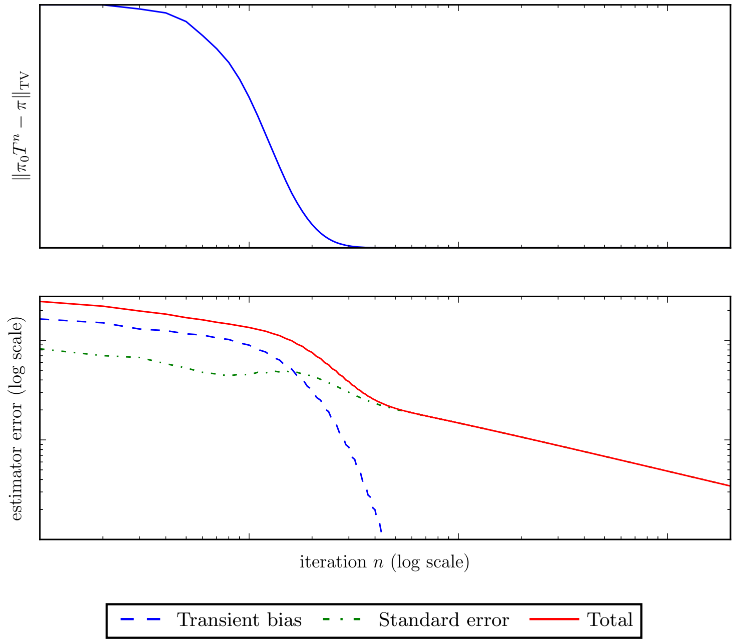


Figure 2.2: A simulation illustrating error terms in MCMC estimator (2.41) as a function of the number of Markov chain iterations (log scale). Because the first half of the Markov chain samples are not used in the estimate, the error due to transient bias is reduced much more quickly than in Figure 2.1 at the cost of shifting up the standard error curve.

With this choice, once the marginal distribution of the Markov chain iterates approaches the stationary distribution the error due to transient bias is reduced at up to exponential rates. See Figure 2.2 for an illustration. With any choice of MCMC estimator, transient bias can be asymptotically decreased at least as fast as $\mathcal{O}(\frac{1}{n})$, and potentially much faster, while MCSE can decrease only as fast as $\mathcal{O}(\frac{1}{\sqrt{n}})$.

Using these ideas, MCMC algorithms provide a general means for estimating posterior expectations of interest: first construct an algorithm to simulate an ergodic Markov chain that admits the intended posterior density as its stationary distribution, and then simply run the simulation, collect samples, and form Monte Carlo estimates from the samples. The task then is to design an algorithm to simulate from

Algorithm 1 Metropolis-Hastings for posterior sampling

```
Input: Initial state $\theta_0$, number of iterations $T$, joint density $p(\theta, \mathbf{x})$,
  proposal density $q(\theta' \,|\, \theta)$
Output: Samples $\theta_1, \dots, \theta_T$
  for $t$ in $0, \dots, T-1$ do
      $\theta' \sim q(\theta' \,|\, \theta_t)$                                          ▷ Generate proposal
      $\alpha(\theta_t, \theta') \leftarrow \min\left(1, \frac{p(\theta', \mathbf{x}) q(\theta_t \,|\, \theta')}{p(\theta_t, \mathbf{x}) q(\theta' \,|\, \theta_t)}\right)$     ▷ Acceptance probability
      $u \sim \mathrm{Unif}(0, 1)$                                     ▷ Set stochastic threshold
      if $\alpha(\theta_t, \theta') > u$ then
          $\theta_{t+1} \leftarrow \theta'$                                      ▷ Accept proposal
      else
          $\theta_{t+1} \leftarrow \theta_t$                                      ▷ Reject proposal
```

such a Markov chain with the intended stationary distribution. In the following sections, we briefly review two canonical procedures for constructing such algorithms: Metropolis-Hastings and Gibbs sampling. For a thorough treatment, see Robert and Casella [2004] and Brooks et al. [2011, Chapter 1].

2.2.5 Metropolis-Hastings (MH) sampling

In the context of Bayesian posterior inference, the Metropolis-Hastings (MH) algorithm simulates a reversible Markov chain over a state space Θ that admits the posterior density $p(\theta \,|\, x)$ as its stationary distribution. The algorithm depends on a user-specified proposal density, $q(\theta'|\theta)$, which can be evaluated numerically and sampled from efficiently, and also requires that the joint density $p(\theta, x)$ can be evaluated (up to proportionality). The MH algorithm then generates a sequence of states $\theta_1, \dots, \theta_T \in \Theta$ according to Algorithm 1.

In each iteration, a proposal for the next state θ' is drawn from the proposal distribution, conditioned on the current state θ. The proposal is stochastically accepted with probability given by the *acceptance probability*,

$$\alpha(\theta, \theta') = \min\left(1, \frac{p(\theta', x) q(\theta \,|\, \theta')}{p(\theta, x) q(\theta' \,|\, \theta)}\right), \tag{2.42}$$

via comparison to a random variate u drawn uniformly from the interval $[0, 1]$. If $u < \alpha(\theta, \theta')$, then the next state is set to the proposal, otherwise, the proposal is rejected and the next state is set to the current state. MH is a generalization of the *Metropolis algorithm* [Metropolis et al., 1953], which requires the proposal distribution to be symmetric, *i.e.*, $q(\theta' \,|\, \theta) = q(\theta \,|\, \theta')$, in which case the acceptance probability is simply

$$\alpha(\theta, \theta') = \min\left(1, \frac{p(\theta', x)}{p(\theta, x)}\right). \tag{2.43}$$

Hastings [1970] later relaxed this by showing that the proposal distribution could be arbitrary.

One can show that the stationary distribution is indeed $p(\theta \,|\, x)$ by showing that the MH transition operator satisfies detailed balance (2.34). The MH transition operator density is a two-component mixture corresponding to the 'accept' event and the 'reject' event:

$$T(\theta \to \theta') = \alpha(\theta, \theta')q(\theta' \,|\, x) + (1 - \beta(\theta))\delta_\theta(\theta') \tag{2.44}$$

$$\beta(\theta) = \int_\Theta \alpha(\theta, \theta')q(\theta' \,|\, \theta)\, d\theta'. \tag{2.45}$$

To show detailed balance, it suffices to show the two balance conditions

$$\alpha(\theta, \theta')q(\theta' \,|\, \theta)p(\theta \,|\, x) = \alpha(\theta', \theta)q(\theta \,|\, \theta')p(\theta' \,|\, x) \tag{2.46}$$

$$(1 - \beta(\theta))\delta_\theta(\theta')p(\theta \,|\, x) = (1 - \beta(\theta'))\delta_{\theta'}(\theta)p(\theta' \,|\, x). \tag{2.47}$$

To show (2.46) we write

$$\begin{aligned}
\alpha(\theta, \theta')q(\theta' \,|\, \theta)p(\theta \,|\, x) &= \min\left(1, \frac{p(\theta', x)q(\theta \,|\, \theta')}{p(\theta, x)q(\theta' \,|\, \theta)}\right) q(\theta' \,|\, \theta)p(\theta \,|\, x) \\
&= \min\left(q(\theta' \,|\, \theta)p(\theta \,|\, x), p(\theta', x)q(\theta \,|\, \theta')\frac{p(\theta \,|\, x)}{p(\theta, x)}\right) \\
&= \min\left(q(\theta' \,|\, \theta)p(\theta \,|\, x), p(\theta' \,|\, x)q(\theta \,|\, \theta')\right) \\
&= \min\left(q(\theta' \,|\, \theta)p(\theta, x)\frac{p(\theta' \,|\, x)}{p(\theta', x)}, p(\theta' \,|\, x)q(\theta \,|\, \theta')\right) \\
&= \min\left(1, \frac{p(\theta, x)q(\theta' \,|\, \theta)}{p(\theta', x)q(\theta \,|\, \theta')}\right) q(\theta \,|\, \theta')p(\theta' \,|\, x).
\end{aligned} \tag{2.48}$$

Algorithm 2 Gibbs sampling

```
Input: A collection of random variables $X = \{X_i : i \in [n]\}$ and
  subroutines to sample $X_i \mid X_{\neg i}$ for each $i \in [n]$
Output: Samples $\{\hat{x}^{(t)}\}$
  Initialize $x = (x_1, x_2, \ldots, x_n)$
  for $t = 1, 2, \ldots$ do
      for $i = 1, 2, \ldots, n$ do
          $x_i \leftarrow$ sample $X_i \mid X_{\neg i} = x_{\neg i}$
      $\hat{x}^{(t)} \leftarrow (x_1, x_2, \ldots, x_n)$
```

To show (2.47), note that if $\theta \neq \theta'$ then both sides are zero, and if $\theta = \theta'$ then both sides are trivially equal.

See Robert and Casella [2004, Section 7.3] for a more detailed treatment of the Metropolis-Hastings algorithm.

2.2.6 Gibbs sampling

Given a collection of n random variables $X = \{X_i : i \in [n]\}$, the Gibbs sampling algorithm iteratively samples each variable X_i conditioned on the sampled values of the others, $X_{\neg i} \triangleq X \setminus \{X_i\}$. The algorithm is summarized in Algorithm 2. When the random variables have a nontrivial probabilistic graphical model, the conditioning can be reduced to each variable's respective Markov blanket [Koller and Friedman, 2009, Section 12.3.1].

A variant of the *systematic scan* of Algorithm 2, in which nodes are traversed in a fixed order for each outer iteration, is the *random scan*, in which nodes are traversed according to a random permutation sampled for each outer iteration. An advantage of the random scan (and other variants) is that the chain becomes reversible and therefore simpler to analyze [Robert and Casella, 2004, Section 10.1.2].

The Gibbs sampling algorithm can be analyzed as a special case of the Metropolis-Hastings algorithm, where the proposal distribution is based on the conditional distributions and the acceptance probability is always one. If the Markov chain produced by a Gibbs sampling algorithm is ergodic, then the stationary distribution is the target distribution of X [Robert and Casella, 2004, Theorem 10.6]. The Markov chain for a Gibbs sampler can fail to be ergodic if, for example, the support of the target distribution is disconnected [Robert and Casella, 2004, Example 10.7]. A sufficient condition for Gibbs sampling to be ergodic is that all conditional densities exist and are positive everywhere [Robert and Casella, 2004, Theorem 10.8].

For a more detailed treatment of Gibbs sampling theory, see Robert and Casella [2004, Chapters 6 and 10].

2.3 Mean field variational inference

In mean field, and variational inference more generally, the task is to approximate an intractable distribution, such as a complex posterior, with a distribution from a tractable family. This tractable approximating distribution can then be used as a proxy for the intractable distribution, and we can estimate expectations of interest with respect to the approximating distribution. In this section we define the mean field optimization problem, which we revisit in Chapter 5.

To set up posterior inference as an optimization problem, we first define an objective function that measures the accuracy of an approximating distribution. The mean field variational inference approach defines an objective function using the following variational inequality.

Proposition 2.4 (Mean field variational inequality)**.** For a probability density p with respect to a base measure ν of the form

$$p(x) = \frac{1}{Z}\bar{p}(x) \quad \text{with} \quad Z \triangleq \int \bar{p}(x)\nu(dx), \tag{2.49}$$

where $\bar{p}$ is the unnormalized density, for all densities q with respect to ν we have

$$\log Z = \mathcal{L}[q] + \mathrm{KL}(q\|p) \geq \mathcal{L}[q], \tag{2.50}$$

where

$$\mathcal{L}[q] \triangleq \mathbb{E}_{q(x)}\left[\log\frac{\bar{p}(x)}{q(x)}\right] = \mathbb{E}_{q(x)}\left[\log\bar{p}(x)\right] + \mathbb{H}[q], \tag{2.51}$$

$$\mathrm{KL}(q\|p) \triangleq \mathbb{E}_{q(x)}\left[\log\frac{q(x)}{p(x)}\right], \qquad \mathbb{H}[q] \triangleq -\mathbb{E}_{q(x)}\left[q(x)\right]. \tag{2.52}$$

Proof. To show the equality, with $x \sim q$ we write

$$\mathcal{L}[q] + \mathrm{KL}(q\|p) = \mathbb{E}_{q(x)}\left[\frac{\bar{p}(x)}{q(x)}\right] + \mathbb{E}_{q(x)}\left[\log\frac{q(x)}{p(x)}\right] \tag{2.53}$$

$$= \mathbb{E}_{q(x)}\left[\log\frac{\bar{p}(x)}{p(x)}\right] \tag{2.54}$$

$$= \log Z. \tag{2.55}$$

The inequality follows from the property $\mathrm{KL}(q\|p) \geq 0$, known as Gibbs's inequality, which follows from Jensen's inequality and the fact that the logarithm is concave:

$$-\mathrm{KL}(q\|p) = \mathbb{E}_{q(x)}\left[\log\frac{p(x)}{q(x)}\right] \leq \log\int q(x)\frac{p(x)}{q(x)}\nu(dx) = 0 \tag{2.56}$$

with equality if and only if $q = p$ (ν-a.e.). □

We call $\mathcal{L}[q]$ the *variational lower bound* on the log partition function $\log Z$. Due to the statistical physics origins of variational inference methods, the negative log partition function $-\log Z$ is also called the free energy (or proportional to the free energy), and $\mathcal{L}[q]$ is also sometimes called the (negative) variational free energy [MacKay, 2003, Section 33.1]. For two densities q and p with respect to the same base measure, $\mathrm{KL}(q\|p)$ is the Kullback-Leibler divergence from q to p, used as a score of dissimilarity between pairs of densities [Amari and Nagaoka, 2007].

The variational lower bound of Proposition 2.4 is useful in inference because if we wish to approximate an intractable p with a tractable q by minimizing $\mathrm{KL}(q\|p)$, we can equivalently choose q to maximize $\mathcal{L}[q]$, which is possible to evaluate since it does not include the partition function Z. The mean field variational inference problem is then to choose the approximating distribution $q(x)$ over the family $\mathcal{Q}$ to optimize the objective $\mathcal{L}[q]$, as we summarize in the next definition.

Definition 2.4 (Mean field variational inference problem)**.** Given a target probability density $p(x) = \frac{1}{Z}\bar{p}(x)$, an approximating family $\mathcal{Q}$, the mean field variational inference problem is

$$\max_{q\in\mathcal{Q}} \mathcal{L}[q] \quad \text{or} \quad \max_{q\in\mathcal{Q}} \mathbb{E}_{q(x)}\left[\log\frac{\bar{p}(x)}{q(x)}\right]. \tag{2.57}$$

The variational family $\mathcal{Q}$ is often made tractable by enforcing factorization structure. This factorization structure in $q(x)$, along with how factors can be updated one at a time in terms of expectations with respect to the other factors, gives the mean field method its name.

In the context of Bayesian inference, p is usually an intractable posterior distribution, $\bar{p}$ is the joint distribution, and Z is the marginal likelihood, which plays a central role in Bayesian model selection and the minimum description length (MDL) criterion [MacKay, 2003, Chapter 28] [Hastie et al., 2001, Chapter 7]. Thus the value of the variational lower bound $\mathcal{L}[q]$ provides a lower bound on the log marginal likelihood, or an *evidence lower bound* (ELBO), and can serve as an approximate model selection criterion [Grosse et al., 2015].

In general mean field variational inference poses a challenging nonlinear programming problem, and standard local optimization methods, such as (block) coordinate ascent or gradient-based ascent methods, can only find local optima or stationary points. However, framing posterior inference as an optimization problem has many advantages. In particular, it allows allows us to draw on scalable optimization algorithms for inference, such as the stochastic gradient optimization methods we summarize in Section 2.5.

See Wainwright and Jordan [2008, Chapter 5] for a convex analysis perspective on mean field variational inference with exponential family graphical models, and see Bishop [2006, Chapter 10] and Murphy [2012, Chapter 22] for treatments that discuss Bayesian posterior inference and conjugacy. Koller and Friedman [2009, Section 11.5.1] develops these methods in the context of alternative variational inference approaches for probabilistic graphical models.

2.4 Expectation propagation variational inference

Expectation propagation (EP) [Minka, 2001, Murphy, 2012, Wainwright and Jordan, 2008] is another variational inference method that can be used for posterior inference in Bayesian models. It is a generalization of Assumed Density Filtering (ADF) [Maybeck, 1982, Opper and Winther, 1998] and loopy belief propagation (LBP) [Murphy et al., 1999], and can provide more accurate posterior approximations than mean field methods in some cases.

As a variational inference method, EP aims to find a tractable distribution $q(\theta)$ that approximates a target distribution $p(\theta)$ by solving an optimization problem. The EP optimization problem is derived from a variational representation of the log partition function of $p(\theta)$, using a Bethe-like entropy approximation and a relaxed set of convex constraints [Wainwright and Jordan, 2008, Section 4.3]. The EP algorithm is then a particular Lagrangian method for solving this constrained optimization problem, where local moment-matching corresponds to updating Lagrange multipliers. However, because this Lagrangian approach has no clear merit function on which to make guaranteed progress [Bertsekas, 2016, Section 5.4], the algorithm has no convergence guarantees. Alternative first-order Lagrangian methods can lead to locally-convergent algorithms [Heskes and Zoeter, 2002, Arrow et al., 1959], though we do not discuss them here.

To develop the standard EP updates heuristically, consider a target posterior distribution $p(\theta)$ with factorization structure and a corresponding factorized variational family $q(\theta)$,

$$p(\theta) \propto \prod_{k=1}^{K} f_k(\theta), \qquad\qquad q(\theta) \propto \prod_{k=1}^{K} q_k(\theta), \tag{2.58}$$

where each factor $q_k(\theta)$ is fixed to be in a parametric model class, such as a Gaussian, with parameters $\widetilde{\eta}_k$. Note that this representation is not necessarily factorized across random variables as in mean field; instead, each factor might be used to approximate a complex likelihood term on θ with a simpler form. For any factor index k, define the *cavity distribution* $q_{\neg k}(\theta)$ formed by the product of the other approximating factors as

$$q_{\neg k}(\theta) \propto \frac{q(\theta)}{q_k(\theta)}. \tag{2.59}$$

We'd like to update the factor $q_k(\theta)$ so that the approximating distribution $q(\theta) \propto q_k(\theta) q_{\neg k}(\theta)$ is closer to the *tilted distribution* $\widetilde{p}_k(\theta)$ defined by

$$\widetilde{p}_k(\theta) \propto f_k(\theta) q_{\neg k}(\theta), \tag{2.60}$$

which is likely to be a better approximation to the target distribution. Therefore we update the parameters $\widetilde{\eta}_k$ of $q_k(\theta)$ to minimize the KL divergence from the tilted distribution to the approximation, writing

$$\widetilde{\eta}_k = \arg\min_{\widetilde{\eta}_k} \mathrm{KL}(\widetilde{p}_k(\theta) \,\|\, q(\theta)) \tag{2.61}$$

$$= \arg\min_{\widetilde{\eta}_k} \mathrm{KL}\left(\tfrac{1}{Z_k} f_k(\theta) q_{\neg k}(\theta) \,\middle\|\, q_k(\theta) q_{\neg k}(\theta)\right), \tag{2.62}$$

where we have introduced the tilted distribution normalizing constant Z_k. To accomplish this update, we compute the moments of the tilted distribution $\widetilde{p}_k(\theta)$ using a method such as exact inference, MCMC, or nested EP, and then choose the parameters of $q_k(\theta)$ to approximately match the estimated moments.

We summarize the resulting algorithm in Algorithm 3. Note that defining the cavity distribution $q_{\neg k}(\theta)$ and symbolically forming the tilted distribution $\widetilde{p}_k(\theta)$ do not require real computational work. Instead, the computational effort is in computing or estimating the moments of $\widetilde{p}_k(\theta)$ and then locally optimizing the parameters $\widetilde{\eta}_k$ of $q_k(\theta)$ to approximately match those moments. Because there are many alternative choices for performing these computations, EP serves as an algorithm template or strategy, and specific EP algorithms fill in these inference and optimization steps. Expectation propagation can also be used to provide an estimate of the marginal likelihood.

2.5 Stochastic gradient optimization

In this section we briefly review some basic ideas in stochastic gradient optimization, which are used most directly in the stochastic variational inference algorithms of Chapter 5. Related stochastic gradient ideas are also used in Chapter 3, particularly in developing Stochastic Gradient Langevin Dynamics. The basic stochastic gradient optimization algorithm is given in Algorithm 4 and sufficient conditions for its convergence to a stationary point are given in Theorem 2.1.

Given a dataset $\bar{y} = \{\bar{y}^{(k)}\}_{k=1}^{K}$, where each $\bar{y}^{(k)}$ is a data *minibatch*, consider the optimization problem

$$\phi^* = \arg\max_{\phi} f(\phi) \tag{2.63}$$

Algorithm 3 Expectation propagation (EP)

```
Input: Target distribution $p(\theta) = \prod_{k=1}^{K} f_k(\theta)$, parameterized approx-
  imating family $q(\theta) = \prod_{k=1}^{K} q_k(\theta)$
Output: Approximate distribution $q(\theta)$
  Initialize parameters of each approximating factor $q_k(\theta)$
  for $t = 1, 2, \ldots$ until convergence do
      for $k = 1, 2, \ldots, K$ do
          Define cavity distribution $q_{\neg k} \propto \frac{q(\theta)}{q_k(\theta)}$
          Define tilted distribution $\widetilde{p}_k(\theta) \propto f_k(\theta) q_{\neg k}(\theta)$
          Set parameters of $q_k(\theta)$ to minimize $\mathrm{KL}(\widetilde{p}_k(\theta) \,\|\, q(\theta))$
              by computing and matching moments
          Define the updated approximation $q(\theta) \propto q_k(\theta) q_{\neg k}(\theta)$
```

where the objective function f decomposes according to

$$f(\phi) = \sum_{k=1}^{K} g(\phi, \bar{y}^{(k)}). \tag{2.64}$$

In the context of variational Bayesian inference, the objective f may be a variational lower bound on the model log evidence and ϕ may be the parameters of the variational family. Alternatively, in MAP inference, f may be the log joint density and ϕ may be the model parameters.

Using the decomposition of f, we can compute unbiased Monte Carlo estimates of its gradient. In particular, if the random index $\hat{k}$ is sampled from $\{1, 2, \ldots, K\}$, denoting the probability of sampling index k as $p_k > 0$, we have

$$\nabla_\phi f(\phi) = \sum_{k=1}^{K} p_k \frac{1}{p_k} \nabla_\phi g(\phi, \bar{y}^{(k)}) = \mathbb{E}_{\hat{k}} \left[\frac{1}{p_{\hat{k}}} \nabla_\phi g(\phi, \bar{y}^{(\hat{k})}) \right]. \tag{2.65}$$

Thus by considering a Monte Carlo approximation to the expectation over $\hat{k}$, we can generate stochastic approximate gradients of the objective f using only a single $\bar{y}^{(k)}$ at a time.

A stochastic gradient ascent algorithm uses these approximate gradients to perform updates and find a stationary point of the objective. At each iteration, such an algorithm samples a data minibatch, computes a gradient with respect to that minibatch, and takes a step in

Algorithm 4 Stochastic gradient ascent

```
Input: $f : \mathbb{R}^n \to \mathbb{R}$ of the form (2.65), sequences $\rho^{(t)}$ and $G^{(t)}$
  Initialize $\phi^{(0)} \in \mathbb{R}^n$
  for $t = 0, 1, 2, \ldots$ do
    $\hat{k}^{(t)} \leftarrow$ sample index $k$ with probability $p_k$
    $\phi^{(t+1)} \leftarrow \phi^{(t)} + \rho^{(t)} \frac{1}{p_{\hat{k}}} G^{(t)} \nabla_\phi g(\phi^{(t)}, \bar{y}^{(\hat{k}^{(t)})})$
```

that direction. In particular, for a sequence of stepsizes $\rho^{(t)}$ and a sequence of positive definite matrices $G^{(t)}$, a typical stochastic gradient ascent algorithm is given in Algorithm 4.

Stochastic gradient algorithms have very general convergence guarantees, requiring only weak conditions on the step size sequence and even the accuracy of the gradients themselves (along with standard Lipschitz smoothness of the objective). We summarize a common set of sufficient conditions in Theorem 2.1. Proofs of this result, along with more general versions, can be found in Bertsekas and Tsitsiklis [1989] and Bottou [1998]. The conditions on the step size sequence were originally developed in Robbins and Monro [1951]. Note also that while the construction here has assumed that the stochasticity in the gradients arises only from randomly subsampling a finite sum, more general versions allow for other sources of stochasticity, typically requiring only bounded variance and allowing some degree of bias [Bertsekas and Tsitsiklis, 1989, Section 7.8].

Theorem 2.1. Given a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ of the form (2.64), if

1. there exists a constant C_0 such that $f(\phi) \leq C_0$ for all $\phi \in \mathbb{R}^n$,
2. there exists a constant C_1 such that
$$\|\nabla f(\phi) - \nabla f(\phi')\|_2 \leq C_1 \|\phi - \phi'\|_2 \quad \forall \phi, \phi' \in \mathbb{R}^n,$$
3. there are positive constants C_2 and C_3 such that
$$\forall t \; C_2 I \prec G^{(t)} \prec C_3 I,$$

4. and the stepsize sequence $\rho^{(t)}$ satisfies

$$\sum_{t=0}^{\infty} \rho^{(t)} = \infty \quad \text{and} \quad \sum_{t=0}^{\infty} (\rho^{(t)})^2 < \infty,$$

then Algorithm 4 converges to a stationary point in the sense that

$$\liminf_{t \to \infty} \|\nabla f(\phi^{(t)})\| = 0 \qquad \text{(almost surely)}. \tag{2.66}$$

While stochastic optimization theory provides convergence guarantees, there is no general theory to analyze global rates of parameter convergence for nonconvex problems such as those that commonly arise in posterior inference. Indeed, the empirical rate of convergence often depends strongly on the variance of the stochastic gradient updates and on the choice of step size sequence. There are automatic methods to tune or adapt the sequence of stepsizes [Snoek et al., 2012, Ranganath et al., 2013], though we do not discuss them here.

3

MCMC with data subsets

In MCMC sampling for Bayesian inference, the task is to simulate a Markov chain that admits as its stationary distribution the posterior distribution of interest. While there are many standard procedures for constructing and simulating from such Markov chains, when the dataset is large many of these algorithms' updates become computationally expensive. This growth in complexity naturally suggests the question of whether there are MCMC procedures that can generate approximate posterior samples without using the full dataset in each update. In this chapter, we focus on recent MCMC sampling schemes that scale Bayesian inference by operating on only subsets of data at a time.

3.1 Factoring the joint density

In most Bayesian inference problems, the fundamental object of interest is the posterior density, which for fixed data is proportional to the product of the prior and the likelihood:

$$p(\theta \,|\, \mathbf{x}) \propto p(\theta, \mathbf{x}) = p(\theta) p(\mathbf{x} \,|\, \theta). \tag{3.1}$$

When the data $\mathbf{x} = \{x_n\}_{n=1}^{N}$ are conditionally independent given the model parameters θ, the likelihood can be decomposed into a product of terms:

$$p(\theta \,|\, \mathbf{x}) \propto p(\theta)p(\mathbf{x} \,|\, \theta) = p(\theta) \prod_{n=1}^{N} p(x_n \,|\, \theta). \tag{3.2}$$

When N is large, this factorization can be exploited to construct MCMC algorithms in which the updates depend only on subsets of the data.

In particular, we can use subsets of data to form an unbiased Monte Carlo estimate of the log likelihood and consequently the log joint density. The log likelihood is a sum of terms:

$$\log p(\mathbf{x} \,|\, \theta) = \sum_{n=1}^{N} \log p(x_n \,|\, \theta), \tag{3.3}$$

and we can approximate this sum using a random subset of $m < N$ terms

$$\log p(\mathbf{x} \,|\, \theta) \approx \frac{N}{m} \sum_{n=1}^{m} \log p(x_n^* \,|\, \theta), \tag{3.4}$$

where $\{x_n^*\}_{n=1}^m$ is a subset of $\{x_n\}_{n=1}^N$ sampled uniformly at random and with replacement. This approximation is an unbiased estimator and yields an unbiased estimate of the log joint density:

$$\log p(\theta)p(\mathbf{x} \,|\, \theta) \approx \log p(\theta) + \frac{N}{m} \sum_{n=1}^{m} \log p(x_n^* \,|\, \theta). \tag{3.5}$$

Several of the methods reviewed in this chapter exploit this estimator to perform MCMC updates.

3.2 Adaptive subsampling for Metropolis–Hastings

In traditional Metropolis–Hastings (MH), we evaluate the joint density to decide whether to accept or reject a proposal. As noted by Korattikara et al. [2014], because the value of the joint density depends on the full dataset, when N is large this is an unappealing amount of computation to reach a binary decision. In this section, we survey ideas for using approximate MH tests that depend on only a subset of the full dataset. The resulting approximate MCMC algorithms proceed in each iteration by reading only as much data as required to satisfy some estimated error tolerance.

While there are several variations, the common idea is to model the probability that the outcome of such an approximate MH test differs from the exact MH test. This probability model allows us to construct an approximate MCMC sampler, outlined in Section 3.2.2, where the user specifies some tolerance for the error in an MH test and the amount of data evaluated is controlled by an *adaptive stopping rule*. Different models for the MH test error lead to different stopping rules. Korattikara et al. [2014] use a normal model to construct a t-statistic hypothesis test, which we describe in Section 3.2.3. Bardenet et al. [2014] instead use concentration inequalities, which we describe in Section 3.2.4. Given an error model and resulting stopping rule, both schemes rely on an MH test based on a Monte Carlo estimate of the log joint density, which we summarize in Section 3.2.1. Our notation in this section follows Bardenet et al. [2014].

Bardenet et al. [2014] observe that similar ideas have been developed both in the context of simulated annealing[1] by the operations research community [Bulgak and Sanders, 1988, Alkhamis et al., 1999, Wang and Zhang, 2006], and in the context of MCMC inference for factor graphs [Singh et al., 2012].

3.2.1 An approximate MH test based on a data subset

In the Metropolis–Hastings algorithm (§2.2.5), the proposal is stochastically accepted when

$$\frac{p(\theta' \,|\, \mathbf{x}) q(\theta \,|\, \theta')}{p(\theta \,|\, \mathbf{x}) q(\theta' \,|\, \theta)} > u, \tag{3.6}$$

where $u \sim \text{Unif}(0, 1)$. Rearranging and using log probabilities gives

$$\log \left[\frac{p(\mathbf{x} \,|\, \theta')}{p(\mathbf{x} \,|\, \theta)} \right] > \log \left[u \frac{q(\theta' \,|\, \theta) p(\theta)}{q(\theta \,|\, \theta') p(\theta')} \right]. \tag{3.7}$$

Scaling both sides by $1/N$ gives an equivalent threshold,

$$\Lambda(\theta, \theta') > \psi(u, \theta, \theta'), \tag{3.8}$$

[1] Simulated annealing is a stochastic optimization heuristic that is operationally similar to MH.

where on the left, $\Lambda(\theta, \theta')$ is the average log likelihood ratio,

$$\Lambda(\theta, \theta') = \frac{1}{N} \sum_{n=1}^{N} \log \left[\frac{p(x_n \,|\, \theta')}{p(x_n \,|\, \theta)} \right] \equiv \frac{1}{N} \sum_{n=1}^{N} \ell_n, \tag{3.9}$$

where

$$\ell_n = \log p(x_n \,|\, \theta') - \log p(x_n \,|\, \theta), \tag{3.10}$$

and on the right,

$$\psi(u, \theta, \theta') = \frac{1}{N} \log \left[u \frac{q(\theta' \,|\, \theta) p(\theta)}{q(\theta \,|\, \theta') p(\theta')} \right]. \tag{3.11}$$

We can form an approximate threshold by subsampling the ℓ_n. Let $\{\ell_n^*\}_{n=1}^m$ be a subsample of size $m < N$, without replacement, from $\{\ell_n\}_{n=1}^N$. This gives the following approximate test:

$$\hat{\Lambda}_m(\theta, \theta') > \psi(u, \theta, \theta'), \tag{3.12}$$

where

$$\hat{\Lambda}_m(\theta, \theta') = \frac{1}{m} \sum_{n=1}^{m} \log \left[\frac{p(x_n^* \,|\, \theta')}{p(x_n^* \,|\, \theta)} \right] \equiv \frac{1}{m} \sum_{n=1}^{m} \ell_n^* \,. \tag{3.13}$$

This subsampled average log likelihood ratio $\hat{\Lambda}_m(\theta, \theta')$ is an unbiased estimate of the average log likelihood ratio $\Lambda(\theta, \theta')$. However, an error is made in the event that the approximate test (3.12) disagrees with the exact test (3.8), and the probability of such an error event depends on the distribution of $\hat{\Lambda}_m(\theta, \theta')$ and not just its mean.

Note that because the proposal θ' is usually a small perturbation of θ, we expect $\log p(x_n \,|\, \theta')$ to be similar to $\log p(x_n \,|\, \theta)$. In this case, we expect the log likelihood ratios ℓ_n have a smaller variance compared to the variance of $\log p(x_n \,|\, \theta)$ across data terms.

3.2.2 Approximate MH with an adaptive stopping rule

A nested sequence of data subsets, sampled without replacement, that converges to the complete dataset gives us a sequence of approximate MH tests that converges to the exact MH test. Modeling the error of such an approximate MH test gives us a mechanism for designing an approximate MH algorithm in which, at each iteration, we incrementally read more data until an adaptive stopping rule informs us that

Algorithm 5 Approximate MH with an adaptive stopping rule

```
Input: Initial state $\theta_0$, number of iterations $T$, data $\mathbf{x} = \{x_n\}_{n=1}^N$,
posterior $p(\theta \,|\, \mathbf{x})$, proposal $q(\theta' \,|\, \theta)$
Output: Samples $\theta_1, \dots, \theta_T$
for $t$ in $0, \dots, T-1$ do
    $\theta' \sim q(\theta' \,|\, \theta_t)$                                   ▷ Generate proposal
    $u \sim \mathrm{Unif}(0, 1)$                                ▷ Draw random number
    $\psi(u, \theta_t, \theta') \leftarrow \frac{1}{N} \log \left[ u \frac{q(\theta' \,|\, \theta_t) p(\theta_t)}{q(\theta_t \,|\, \theta') p(\theta')} \right]$
    $\hat{\Lambda}(\theta_t, \theta') \leftarrow$ AvgLogLikeRatioEstimate$(\theta_t, \theta', \psi(u, \theta_t, \theta'))$
    if $\hat{\Lambda}(\theta_t, \theta') > \psi(u, \theta_t, \theta')$ then          ▷ Approximate MH test
        $\theta_{t+1} \leftarrow \theta'$                                  ▷ Accept proposal
    else
        $\theta_{t+1} \leftarrow \theta_t$                                 ▷ Reject proposal
```

our error is less than some user-specified tolerance. Algorithm 5 outlines this approach. The function AvgLogLikeRatioEstimate computes $\hat{\Lambda}(\theta, \theta')$ according to an adaptive stopping rule that depends on an error model, *i.e.*, a way to approximate or bound the probability that the approximate outcome disagrees with the full-data outcome:

$$\mathbb{P}\left[((\hat{\Lambda}_m(\theta, \theta') > \psi(u, \theta, \theta')) \neq ((\Lambda(\theta, \theta') > \psi(u, \theta, \theta')) \right]. \tag{3.14}$$

We describe two possible error models in Sections 3.2.3 and 3.2.4.

A practical issue with adaptive subsampling is choosing the sizes of the data subsets. One approach, taken by Korattikara et al. [2014], is to use a fixed batch size b and read b more data points at a time. Bardenet et al. [2014] instead geometrically increase the total subsample size, and also discuss connections between adaptive stopping rules and related ideas such as bandit problems, racing algorithms and boosting.

3.2.3 Using a t-statistic hypothesis test

Korattikara et al. [2014] propose an approximate MH acceptance probability that uses a parametric test of significance as its error model. By assuming a normal model for the log likelihood estimate $\hat{\Lambda}(\theta, \theta')$, a t-statistic hypothesis test then provides an estimate of whether the approximate outcome agrees with the full-data outcome, *i.e.*, the expression in Equation (3.14). This leads to an adaptive framework as in Section 3.2.2 where, at each iteration, the data are processed incrementally until the t-test satisfies some user-specified tolerance ϵ.

Let us model the ℓ_n as i.i.d. from a normal distribution with bounded variance σ^2:

$$\ell_n \sim \mathcal{N}(\mu, \sigma^2)\,. \tag{3.15}$$

The mean estimate $\hat{\mu}_m$ for μ based on the subset of size m is equal to $\hat{\Lambda}_m(\theta, \theta')$:

$$\hat{\mu}_m = \hat{\Lambda}_m(\theta, \theta') = \frac{1}{m} \sum_{n=1}^{m} \ell_n^*\,. \tag{3.16}$$

The error estimate $\hat{\sigma}_m$ for σ may be derived from $s_m/\sqrt{m}$, where s_m is the empirical standard deviation of the m subsampled ℓ_n terms, *i.e.*,

$$s_m = \sqrt{\frac{m}{m-1}\left(\hat{\Lambda}_m^2(\theta, \theta') - \hat{\Lambda}_m(\theta, \theta')^2\right)}, \tag{3.17}$$

where

$$\hat{\Lambda}_m^2(\theta, \theta') = \frac{1}{m} \sum_{n=1}^{m} (\ell_n^*)^2. \tag{3.18}$$

To obtain a confidence interval, we multiply this estimate by the finite population correction, giving:

$$\hat{\sigma}_m = \frac{s_m}{\sqrt{m}} \sqrt{\frac{N-m}{N-1}}\,. \tag{3.19}$$

The test statistic

$$t = \frac{\hat{\Lambda}_m(\theta, \theta') - \psi(u, \theta, \theta')}{\hat{\sigma}_m} \tag{3.20}$$

follows a Student's t-distribution with $m-1$ degrees of freedom when $\Lambda(\theta, \theta') = \psi(u, \theta, \theta')$. The tail probability for $|t|$ then gives the probability that the approximate and actual outcomes agree, and thus

$$\rho = 1 - \phi_{m-1}(|t|) \tag{3.21}$$

Algorithm 6 Estimate of the average log likelihood ratio. The adaptive stopping rule uses a t-statistic hypothesis test.

```
**Parameters:** batch size $b$, user-defined error tolerance $\epsilon$
**function** AvgLogLikeRatioEstimate$(\theta, \theta', \psi(u, \theta, \theta'))$
    $m,\ \hat{\Lambda}(\theta, \theta'),\ \hat{\Lambda}^2(\theta, \theta') \leftarrow 0,\ 0,\ 0$
    **while** True **do**
        $c \leftarrow \min(b, N - m)$
        $\hat{\Lambda}(\theta, \theta') \leftarrow \frac{1}{m+c}\left(m\hat{\Lambda}(\theta, \theta') + \sum_{n=m+1}^{m+c} \log \frac{p(x_n \,|\, \theta')}{p(x_n \,|\, \theta)}\right)$
        $\hat{\Lambda}^2(\theta, \theta') \leftarrow \frac{1}{m+c}\left(m\hat{\Lambda}^2(\theta, \theta') + \sum_{n=m+1}^{m+c} \left[\log \frac{p(x_n \,|\, \theta')}{p(x_n \,|\, \theta)}\right]^2\right)$
        $m \leftarrow m + c$
        $s \leftarrow \sqrt{\frac{m}{m-1}\left(\hat{\Lambda}^2(\theta, \theta') - \hat{\Lambda}(\theta, \theta')^2\right)}$
        $\hat{\sigma} \leftarrow \frac{s}{\sqrt{m}}\sqrt{\frac{N-m}{N-1}}$
        $\rho \leftarrow 1 - \phi_{m-1}\left(\left|\frac{\hat{\Lambda}(\theta, \theta') - \psi(u, \theta, \theta')}{\hat{\sigma}}\right|\right)$
        **if** $\rho > \epsilon$ **or** $m = N$ **then**
            **return** $\hat{\Lambda}(\theta, \theta')$
```

is the probability that they disagree, where $\phi_{m-1}(\cdot)$ is the CDF of the Student's t-distribution with $m - 1$ degrees of freedom. The t-test thus gives an adaptive stopping rule, *i.e.*, for any user-provided tolerance $\epsilon \geq 0$, we can incrementally increase m until $\rho \leq \epsilon$. We illustrate this approach in Algorithm 6.

3.2.4 Using concentration inequalities

Bardenet et al. [2014] propose an adaptive subsampling method that is mechanically similar to using a t-test but instead uses concentration inequalities. In addition to a bound on the error (of the approximate acceptance probability) that is local to each iteration, concentration bounds yield a bound on the total variation distance between the approximate and true stationary distributions. The method is further refined with variance reduction techniques in Bardenet et al. [2015], though we only discuss the basic version here.

As in Section 3.2.3, we evaluate an approximate MH threshold based on a data subset of size m, given in Equation (3.12). We bound the probability that the approximate binary outcome is incorrect via *concentration inequalities* that characterize the quality of $\hat{\Lambda}_m(\theta, \theta')$ as an estimate for $\Lambda(\theta, \theta')$. Such a concentration inequality is a probabilistic statement that, for $\delta_m \in (0, 1)$ and some constant c_m,

$$\mathbb{P}\left(\left|\hat{\Lambda}_m(\theta, \theta') - \Lambda(\theta, \theta')\right| \leq c_m\right) \geq 1 - \delta_m. \tag{3.22}$$

For example, in Hoeffding's inequality without replacement [Serfling, 1974]

$$c_m = C_{\theta,\theta'}\sqrt{\frac{2}{m}\left(1 - \frac{m-1}{N}\right)\log\left(\frac{2}{\delta_m}\right)} \tag{3.23}$$

where

$$C_{\theta,\theta'} = \max_{1\leq n\leq N}\left|\log p(x_n \,|\, \theta') - \log p(x_n \,|\, \theta)\right| = \max_{1\leq n\leq N}|\ell_n|, \tag{3.24}$$

using ℓ_n as in Equation (3.10). Alternatively, if the empirical standard deviation s_m of the m subsampled ℓ_n^* terms is small, then the empirical Bernstein bound,

$$c_m = s_m\sqrt{\frac{2\log(3/\delta_m)}{m}} + \frac{6C_{\theta,\theta'}\log(3/\delta_m)}{m}, \tag{3.25}$$

is tighter [Audibert et al., 2009], where s_m is given in Equation (3.17). While $C_{\theta,\theta'}$ can be obtained via all the ℓ_n, this is precisely the computation we want to avoid. Therefore, the user must provide an estimate of $C_{\theta,\theta'}$.

Bardenet et al. [2014] use a concentration bound to construct an adaptive stopping rule based on a strategy called empirical Bernstein stopping [Mnih et al., 2008]. Let c_m be a concentration bound as in Equation (3.23) or (3.25) and let δ_m be the associated error. This concentration bound states that $|\hat{\Lambda}_m(\theta, \theta') - \Lambda(\theta, \theta')| \leq c_m$ with probability $1 - \delta_m$. If $|\hat{\Lambda}_m(\theta, \theta') - \psi(u, \theta, \theta')| > c_m$, then the approximate MH

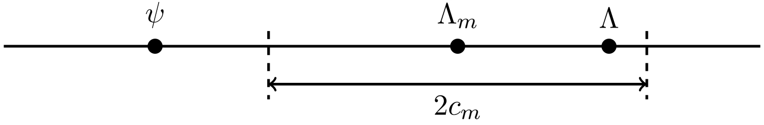


Figure 3.1: Reproduction of Figure 2 from Bardenet et al. [2014]. If $|\hat{\Lambda}_m(\theta,\theta') - \psi(u,\theta,\theta')| > c_m$, then the adaptive stopping rule using a concentration bound is satisfied and we use the approximate MH test based on $\hat{\Lambda}_m(\theta,\theta')$.

test agrees with the exact MH test with probability $1 - \delta_m$. We reproduce a helpful illustration of this scenario from Bardenet et al. [2014] in Figure 3.1. If instead $|\hat{\Lambda}_m(\theta,\theta') - \psi(u,\theta,\theta')| \leq c_m$, then we want to increase m until this is no longer the case. Let M be the *stopping time*, *i.e.*, the number of data points evaluated using this criterion,

$$M = \min\left(N, \inf_{m\geq 1} \left|\hat{\Lambda}_m(\theta,\theta') - \psi(u,\theta,\theta')\right| > c_m\right). \tag{3.26}$$

We can set δ_m according to a user-defined parameter $\epsilon \in (0,1)$ so that ϵ gives an upper bound on the error of the approximate acceptance probability. Let $p > 1$ and set

$$\delta_m = \frac{p-1}{pm^p}\epsilon, \quad \text{thus} \quad \sum_{m\geq 1} \delta_m \leq \epsilon. \tag{3.27}$$

A union bound argument gives

$$\mathbb{P}\left(\bigcap_{m\geq 1} \left\{\left|\hat{\Lambda}_m(\theta,\theta') - \Lambda(\theta,\theta')\right| \leq c_m\right\}\right) \geq 1-\epsilon, \tag{3.28}$$

under sampling without replacement. Hence, with probability $1-\epsilon$, the approximate MH test based on $\hat{\Lambda}_M(\theta,\theta')$ agrees with the exact MH test. In other words, the stopping rule for computing $\hat{\Lambda}_m(\theta,\theta')$ in Algorithm 5 is satisfied once we observe $|\hat{\Lambda}_m(\theta,\theta') - \psi(u,\theta,\theta')| > c_m$. We illustrate this approach in Algorithm 7, using Hoeffding's inequality without replacement.

In their actual implementation, Bardenet et al. [2014] modify δ_m to reflect the number of batches processed instead of the subsample

Algorithm 7 Estimate of the average log likelihood ratio. The adaptive stopping rule uses Hoeffding's inequality without replacement.

```
**Parameters:** batch size $b$, user-defined error tolerance $\epsilon$, estimate of $C_{\theta,\theta'} = \max_n |\ell_n|$, $p > 1$
function AvgLogLikeRatioEstimate($\theta, \theta', \psi(u, \theta, \theta')$)
    $m,\ \hat{\Lambda}(\theta, \theta') \leftarrow 0,\ 0$
    while True do
        $c \leftarrow \min(b, N - M)$
        $\hat{\Lambda}(\theta, \theta') \leftarrow \frac{1}{m+c}\left(m\hat{\Lambda}(\theta, \theta') + \sum_{n=m+1}^{m+c} \log \frac{p(x_n \mid \theta')}{p(x_n \mid \theta)}\right)$
        $m \leftarrow m + c$
        $\delta \leftarrow \frac{p-1}{pm^p}\epsilon$
        $c \leftarrow C_{\theta,\theta'}\sqrt{\frac{2}{m}\left(1 - \frac{m-1}{N}\right)\log\left(\frac{2}{\delta}\right)}$
        if $\left|\hat{\Lambda}(\theta, \theta') - \psi(u, \theta, \theta')\right| > c$ or $m = N$ then
            return $\hat{\Lambda}(\theta, \theta')$
```

size m. For example, suppose we use the concentration bound in Equation (3.23), *i.e.*, Hoeffding's inequality without replacement. Then after processing a subsample of size m in k batches, the adaptive stopping rule checks whether $|\hat{\Lambda}_m(\theta, \theta') - \psi(u, \theta, \theta')| > c_m$, where

$$c_m = C_{\theta,\theta'}\sqrt{\frac{2}{m}\left(1 - \frac{m-1}{N}\right)\log\left(\frac{2}{\delta_k}\right)} \tag{3.29}$$

and

$$\delta_k = \frac{p-1}{pk^p}\epsilon. \tag{3.30}$$

Also, as mentioned in Section 3.2.2, Bardenet et al. [2014] geometrically increase the subsample size by a factor γ. In their experiments, they use the empirical Bernstein-Serfling bound [Bardenet and Maillard, 2015]. For the hyperparameters, they set $p = 2$, $\gamma = 2$, and $\epsilon = 0.01$, and remark that they empirically found their algorithm to be robust to the choice of ϵ.

3.2.5 Error bounds on the stationary distribution

In this section, we reproduce some theoretical results from Korattikara et al. [2014] and Bardenet et al. [2014]. After setting up some notation, we emphasize the most general aspects of these results, which apply to pairs of transition kernels whose differences are bounded, and thus are not specific to adaptive subsampling procedures. The central theorem is an upper bound on the difference between the stationary distributions of such pairs of kernels in the case of Metropolis–Hastings. Its proof depends on the ability to bound the difference in the acceptance probabilities, at each iteration, of the two MH transition kernels.

Preliminaries and notation. Let P and Q be probability measures (distributions) with Radon–Nikodym derivatives (densities) f_P and f_Q, respectively, and absolutely continuous with respect to measure ν. The *total variation distance* between P and Q is

$$\|P - Q\|_{\mathrm{TV}} \equiv \frac{1}{2}\int_{\theta\in\Theta} |f_P(\theta) - f_Q(\theta)| d\nu(\theta). \tag{3.31}$$

Let T denote an MH transition kernel with stationary distribution π and acceptance probability $\alpha(\theta, \theta')$. Let $\tilde{T}$ denote an approximation to T, using the same proposal distribution $q(\theta' \,|\, \theta)$ but with stationary distribution $\tilde{\pi}$ and acceptance probability $\tilde{\alpha}(\theta, \theta')$. Throughout this section, we specify when $\tilde{T}$ is constructed from T via an adaptive stopping rule; some of the results are more general. Let

$$\mathcal{E}(\theta, \theta') = \tilde{\alpha}(\theta, \theta') - \alpha(\theta, \theta') \tag{3.32}$$

be the acceptance probability error of the approximate MH test, with respect to the exact test. Finally, let

$$\mathcal{E}_{\max} = \sup_{\theta,\theta'} |\mathcal{E}(\theta, \theta')| \tag{3.33}$$

be the worst case absolute acceptance probability error.

The following theorem shows that, under strong ergodicity assumptions on the exact MH chain, we have similar ergodicity guarantees for the approximate MH chain. In addition, the total variation distance between the stationary distribution of the approximate MH chain and the target distribution can also be bounded in terms of the ergodicity parameters of the exact chain and the worst case acceptance probability error given in Eq. (3.33).

Theorem 3.1 (Total variation bound under uniform geometric ergodicity [Bardenet et al., 2014])**.** Let T be uniformly geometrically ergodic, so that there exist real numbers $A < \infty$ and $h < \infty$ such that for all initial densities π_0 and integers $k > 0$,

$$\|T^k \pi_0 - \pi\|_{\mathrm{TV}} \leq A\lambda^{\lfloor k/h \rfloor}. \tag{3.34}$$

Then $\tilde{T}$ is uniformly geometrically ergodic with stationary distribution $\tilde{\pi}$, and we have for some constant $C < \infty$,

$$\begin{align} \|\tilde{T}^k \pi_0 - \tilde{\pi}\|_{\mathrm{TV}} &\leq C \left(1 - (1-\epsilon)^h (1-\lambda)\right)^{\lfloor k/h \rfloor}, \tag{3.35} \\ \|\pi - \tilde{\pi}\|_{\mathrm{TV}} &\leq \frac{Ah\mathcal{E}_{\max}}{1-\lambda}, \tag{3.36} \end{align}$$

where ϵ is the user-defined error tolerance for the algorithm and $\mathcal{E}_{\max}$ is defined in (3.33).

Instead of proving Theorem 3.1, we briefly outline a proof from Korattikara et al. [2014] of a similar theorem that exploits a slightly stronger assumption on T. Specifically, assume T satisfies the strong contraction condition,

$$\|TP - \pi\|_{\mathrm{TV}} \leq \eta \|P - \pi\|_{\mathrm{TV}}, \tag{3.37}$$

for all probability distributions P and some constant $\eta \in [0, 1)$. For approximate MH with an adaptive stopping rule, the maximum acceptance probability error $\mathcal{E}_{\max}$ directly gives an upper bound on the single-step error $\|\tilde{T}P - TP\|_{\mathrm{TV}}$. Combining the single-step error bound with the contraction condition shows that $\tilde{T}$ is also a strong contraction and yields a bound on $\|\tilde{\pi} - \pi\|_{\mathrm{TV}}$.

Finally, we note that an adaptive subsampling schemes using a concentration inequality enables an upper bound on the stopping time [Bardenet et al., 2014].

3.3 Sub-selecting data via a lower bound on the likelihood

Maclaurin and Adams [2014] introduce *Firefly Monte Carlo* (FlyMC), an auxiliary variable MCMC sampling procedure that operates on only subsets of data in each iteration. At each iteration, the algorithm dynamically selects what data to evaluate based on the random indicators included in the Markov chain state. In addition, it generates samples from the exact target posterior rather than an approximation. However, FlyMC requires a lower bound on the likelihood with a particular "collapsible" structure (essentially an exponential family lower bound) and because such bounds are not readily available or tight for many models it is not as generally applicable. The algorithm's performance depends on the tightness of the bound; it can achieve impressive gains in performance when model structure allows.

FlyMC samples from an augmented posterior that eliminates potentially many likelihood factors. Define

$$L_n(\theta) = p(x_n \,|\, \theta) \tag{3.38}$$

and let $B_n(\theta)$ be a strictly positive lower bound on $L_n(\theta)$, *i.e.*, $0 < B_n(\theta) \leq L_n(\theta)$. For each datum, we introduce a binary auxiliary variable $z_n \in \{0, 1\}$ conditionally distributed according to a Bernoulli distribution,

$$p(z_n \,|\, x_n, \theta) = \left[\frac{L_n(\theta) - B_n(\theta)}{L_n(\theta)}\right]^{z_n} \left[\frac{B_n(\theta)}{L_n(\theta)}\right]^{1-z_n}, \tag{3.39}$$

where the z_n are independent for different n. When the bound is tight, *i.e.*, $B_n(\theta) = L_n(\theta)$, then $z_n = 0$ with probability 1. More generally, a tighter bound results in a higher probability that $z_n = 0$. Augmenting the density with $\mathbf{z} = \{z_n\}_{n=1}^N$ gives:

$$\begin{aligned} p(\theta, \mathbf{z} \,|\, \mathbf{x}) &\propto p(\theta \,|\, \mathbf{x}) p(\mathbf{z} \,|\, \mathbf{x}, \theta) \\ &= p(\theta) \prod_{n=1}^{N} p(x_n \,|\, \theta) p(z_n \,|\, x_n, \theta). \end{aligned} \tag{3.40}$$

Using Equations (3.38) and (3.39), we can now write:

$$\begin{aligned}
p(\theta, \mathbf{z} \,|\, \mathbf{x}) &\propto p(\theta) \prod_{n=1}^{N} L_n(\theta) \left[\frac{L_n(\theta) - B_n(\theta)}{L_n(\theta)}\right]^{z_n} \left[\frac{B_n(\theta)}{L_n(\theta)}\right]^{1-z_n} \\
&= p(\theta) \prod_{n=1}^{N} (L_n(\theta) - B_n(\theta))^{z_n} B_n(\theta)^{1-z_n} \\
&= p(\theta) \prod_{n:z_n=1} (L_n(\theta) - B_n(\theta)) \prod_{n:z_n=0} B_n(\theta). \qquad (3.41)
\end{aligned}$$

Thus for any fixed configuration of $\mathbf{z}$ we can evaluate the joint density using only the likelihood terms $L_n(\theta)$ where $z_n = 1$ and the bound values $B_n(\theta)$ for each $n = 1, 2, \ldots, N$.

While Equation (3.41) still involves a product of N terms, if the product of the bound terms $\prod_{n:z_n=0} B_n(\theta)$ can be evaluated without reading each corresponding data point then the joint density can be evaluated reading only the data x_n for which $z_n = 1$. In particular, if the form of $B_n(\theta)$ is an exponential family density, then the product $\prod_{n:z_n=0} B_n(\theta)$ can be evaluated using only a finite-dimensional sufficient statistic for the data $\{x_n : z_n = 0\}$. Thus by exploiting lower bounds in the exponential family, FlyMC can reduce the amount of data required at each iteration of the algorithm while maintaining the exact posterior as its stationary distribution. Maclaurin and Adams [2014] show an application of this methodology to Bayesian logistic regression.

FlyMC presents three main challenges. The first is constructing a collapsible lower bound, such as an exponential family, that is sufficiently tight. The second is designing an efficient implementation. Maclaurin and Adams [2014] discuss these issues and, in particular, design a cache-like data structure for managing the relationship between the N indicator values and the data. Finally, it is likely that the inclusion of these auxiliary variables slows the mixing of the Markov chain, but Maclaurin and Adams [2014] only provide empirical evidence that this effect is small relative to the computational savings from using data subsets.

Further analysis of FlyMC, and in particular an explicit connection to pseudo-marginal techniques and an analysis of how bound tightness can affect performance, can be found in Bardenet et al. [2015, Section 4.3].

3.4 Stochastic gradients of the log joint density

In this section, we review recent efforts to develop MCMC algorithms inspired by stochastic optimization techniques. This is motivated by the existence of, first, MCMC algorithms that can be thought of as the sampling analogues of optimization algorithms, and second, scalable stochastic versions of these optimization algorithms.

Traditional gradient ascent or descent performs optimization by iteratively computing and following a local gradient [Dennis and Schnabel, 1983]. In Bayesian MAP inference, the objective function is typically a log joint density and the update rule for gradient ascent is given by

$$\theta_{t+1} = \theta_t + \frac{\epsilon_t}{2}\Big(\nabla \log p(\theta_t, \mathbf{x})\Big) \tag{3.42}$$

for $t = 1, \ldots, \infty$ and some step size sequence (ϵ_t). As discussed in Section 2.5, stochastic gradient descent (SGD) is a simple modification of gradient descent that can exploit situations where the objective function decomposes into a sum of many terms. While the traditional gradient descent update depends on all the data, *i.e.*,

$$\theta_{t+1} = \theta_t + \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \sum_{n=1}^{N} \nabla \log p(x_n \,|\, \theta_t)\right), \tag{3.43}$$

SGD forms an update based on only a data subset,

$$\theta_{t+1} = \theta_t + \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{m}\sum_{n=1}^{m} \nabla \log p(x_n \,|\, \theta_t)\right). \tag{3.44}$$

The iterates converge to a local extreme point of the log joint density in the sense that $\lim_{t\to\infty} \nabla \log p(\theta_t|\mathbf{x}) = 0$ if the step size sequence $\{\epsilon_t\}_{t=1}^{\infty}$ satisfies

$$\sum_{t=1}^{\infty} \epsilon_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \epsilon_t^2 < \infty. \tag{3.45}$$

A common choice of step size sequence is $\epsilon_t = \alpha(\beta + t)^{-\gamma}$ for some $\beta > 0$ and $\gamma \in (0.5, 1]$.

Welling and Teh [2011] propose *stochastic gradient Langevin dynamics* (SGLD), an approximate MCMC procedure that combines SGD with a simple kind of *Langevin dynamics* (*Langevin Monte Carlo*) [Neal, 1994]. They extend the *Metropolis-adjusted Langevin algorithm* (MALA) that uses noisy gradient steps to generate proposals for a Metropolis–Hastings chain [Roberts and Tweedie, 1996]. At iteration t, the MH proposal is

$$\theta' = \theta_t + \frac{\epsilon}{2}\Big(\nabla \log p(\theta_t, \mathbf{x})\Big) + \eta_t, \tag{3.46}$$

where the injected noise $\eta_t \sim \mathcal{N}(0, \epsilon)$ is Gaussian. Notice that the scale of the noise is $\sqrt{\epsilon}$, *i.e.*, is constant and set by the gradient step size parameter. The MALA proposal is thus a stochastic gradient step, constructed by adding noise to a step in the direction of the gradient.

SGLD modifies the Langevin dynamics in Equation (3.46) by using stochastic gradients based on data subsets, as in Equation (3.44), and requiring that the step size parameter satisfies Equation (3.45). Thus, at iteration t, the proposal is

$$\theta' = \theta_t + \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{m}\sum_{n=1}^{m} \nabla \log p(x_n \,|\, \theta_t)\right) + \eta_t, \tag{3.47}$$

where $\eta_t \sim \mathcal{N}(0, \epsilon_t)$. Notice that the injected noise decays with the gradient step size parameter, but at a slower rate. Specifically, if ϵ_t decays as $t^{-\gamma}$, then η_t decays as $t^{-\gamma/2}$. As in MALA, the SGLD proposal is a stochastic gradient step, where the noise comes from subsampling as well as the injected noise.

An actual Metropolis–Hastings algorithm would accept or reject the proposal in Equation (3.47) by evaluating the full (log) joint density at θ' and θ_t, but this is precisely the computation we wish to avoid. Welling and Teh [2011] observe that as $\epsilon_t \to 0$, $\theta' \to \theta_t$ in both Equations (3.46) and (3.47). In this limit, the probability of accepting the proposal converges to 1, but the chain stops completely. The authors suggest that ϵ_t can be decayed to a value that is large enough for efficient sampling, yet small enough for the acceptance probability to be high. These assumptions lead to a scheme where $\epsilon_t > \epsilon_\infty > 0$, for all t, and all proposals are accepted, therefore the acceptance probability is

Algorithm 8 Stochastic gradient Langevin dynamics (SGLD).

```
Input: Initial state $\theta_0$, number of iterations $T$, data $\mathbf{x}$, grad log prior
$\nabla \log p(\theta)$, grad log likelihood $\nabla \log p(x \mid \theta)$, batch size $m$, step size
tuning parameters ($e.g.$, $\alpha, \beta, \gamma$)
Output: Samples $\theta_1, \dots, \theta_T$
$J = N/m$
for $\tau$ in $0, \dots, T/J - 1$ do
    $\mathbf{x} \leftarrow$ PERMUTE($\mathbf{x}$)        ▷ For sampling without replacement
    for $k$ in $0, \dots, J - 1$ do
        $t = \tau J + k$
        $\epsilon_t \leftarrow \alpha(\beta + t)^{-\gamma}$        ▷ Example step size
        $\eta_t \sim \mathcal{N}(0, \epsilon_t)$        ▷ Draw noise to inject
        $\theta' \leftarrow \theta_t + \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{m}\sum_{n=km+1}^{km+m} \nabla \log p(x_n \mid \theta_t)\right) + \eta_t$
        $\theta_{t+1} \leftarrow \theta'$        ▷ Accept proposal with probability 1
```

never evaluated. We show this scheme in Algorithm 8. Without the stochastic MH acceptance step, however, asymptotic samples are no longer guaranteed to represent the target distribution.

The SGLD algorithm and its analysis have been extended in a number of directions. *Stochastic gradient Fisher scoring* modifies the injected noise to interpolate between fast, SGLD-like mixing when the posterior is close to Gaussian, and slower mixing capable of capturing non-Gaussian features [Ahn et al., 2012]. In more recent work, Patterson and Teh [2013] apply SGLD to *Riemann manifold Langevin dynamics* [Girolami and Calderhead, 2011]. There has also been some recent analysis of the asymptotic behavior of SGLD [Teh et al., 2016].

Several recent works [Chen et al., 2014, Shang et al., 2015, Ma et al., 2015] also combine the idea of SGD with Hamiltonian Monte Carlo (HMC) [Neal, 1994, 2010]. HMC improves and generalizes Langevin dynamics with coherent long-range exploration, though there is disagreement as to whether these stochastic gradient versions sacrifice the main advantages of HMC [Betancourt, 2015]. These methods are also related to recent approaches that combine the idea of ther-

Table 3.1: Summary of recent MCMC methods for Bayesian inference that operate on data subsets. Error refers to the total variation distance between the stationary distribution of the Markov chain and the target posterior distribution.

	Adaptive subsampling	**FlyMC**	**SGLD**
Approach	Approximate MH test	Auxiliary variables	Optimization plus noise
Requirements	Error model, *e.g.*, t-test	Likelihood bound	Gradients $\nabla \log p(\theta, \mathbf{x})$
Data access	Mini-batches	Random	Mini-batches
Hyperparameters	Batch size error, tolerance per iteration	None	Batch size, error tolerance, annealing schedule
Asymptotic bias	Bounded TV	None	Biased for all $\epsilon > 0$

mostats with stochastic gradients [Leimkuhler and Shang, 2016, Ding et al., 2014].

Finally, we note that all the methods in this section require gradient information, and in particular require the sampled variables to be continuous.

3.5 Summary

In this chapter, we have surveyed three recent approaches to scaling MCMC that operate on subsets of data. Below and in Table 3.1, we summarize and compare adaptive subsampling approaches (§3.2), FlyMC (§3.3), and SGLD (§3.4) along several axes.

Approaches. Adaptive subsampling approaches replace the Metropolis–Hastings (MH) test, a function of all the data, with an approximate test that depends on only a subset. FlyMC is an auxiliary variable method that stochastically replaces likelihood computations with a collapsible lower bound. Stochastic gradient Langevin dynamics (SGLD) replaces gradients in a Metropolis-adjusted Langevin algorithm (MALA) with stochastic gradients based on data subsets and eliminates the Metropolis–Hastings test.

Generality, requirements, and assumptions. Each of the methods exploits assumptions or additional problem structure. Adaptive subsampling methods require an error model that accurately represents the probability that an approximate MH test will disagree with the exact MH test. A normal model [Korattikara et al., 2014] or concentration bounds [Bardenet et al., 2014] represent natural choices; under certain conditions, tighter concentration bounds may apply. FlyMC requires a strictly positive collapsible lower bound on the likelihood, essentially an exponential family lower bound, which may not in general be available. SGLD requires the log gradients of the prior and likelihood.

Data access patterns. While all the methods use subsets of data, their access patterns differ. Adaptive subsampling and SGLD require randomization to avoid issues of bias due to data order, but this randomization can be achieved by permuting the data before each pass and hence these algorithms allow data access that is mostly sequential. In contrast, FlyMC operates on random subsets of data determined by the Markov chain itself, leading to a random access pattern. However, subsets from one iteration to the next tend to be correlated, and motivate implementation details such as the proposed cache data structure.

Hyperparameters. FlyMC does not introduce additional hyperparameters that require tuning. Both adaptive subsampling methods and SGLD introduce hyperparameters that can significantly affect performance. Both are mini-batch methods, and thus have the batch size as a tuning parameter. In adaptive subsampling methods, the stopping criterion is evaluated potentially more than once before it is satisfied. This motivates schemes that geometrically increase the amount of data processed whenever the stopping criterion is not satisfied, which introduces additional hyperparameters. Adaptive subsampling methods additionally provide a single tuning parameter that allows the user to control the error at each iteration. Finally, since these adaptive methods define an approximate MH test, they implicitly also require that the user specify a proposal distribution. For SGLD, the user must specify an annealing schedule for the step size parameter; in particular, it should converge to a small positive value so that the injected noise term dominates, while not being too large compared to the scale of the posterior distribution.

Error. FlyMC is exact in the sense that the target posterior distribution is a marginal of its augmented state space. The adaptive subsampling approaches and SGLD (for any positive step size) are approximate methods in that neither has a stationary distribution equal to the target posterior. The adaptive subsampling approaches bound the error of the MH test at each iteration, and for MH transition kernels with uniform ergodicity this one-step error bound leads to an upper bound on the total variation distance between the approximate stationary distribution and the target posterior distribution. The theoretical analysis of SGLD is less clear [Sato and Nakagawa, 2014].

3.6 Discussion

Data subsets. The methods surveyed in this chapter achieve computational gains by using data subsets in place of an entire dataset of interest. The adaptive subsampling algorithms (§3.2) are more successful when a small subsample leads to an accurate estimator for the exact MH test's accept/reject decision. Intuitively, such an estimator is easier to construct when the log posterior values at the proposed and current states are significantly different. This tends to be true far away from the mode(s) of the posterior, *e.g.*, in the tails of a distribution that decay exponentially fast, compared to the area around a mode, which is locally more flat. Thus, these algorithms tend to evaluate more data when the chain is in the vicinity of a mode, and less data when the chain is far away (which tends to be the case for an arbitrary initial condition). SGLD (§3.4) exhibits somewhat related behavior. Recall that SGLD behaves more like SGD when the update rule is dominated by the gradient term, which tends to be true during the initial execution phase. Similar to SGD, the chain progresses toward a mode at a rate that depends on the accuracy of the stochastic gradients. For a log posterior target, stochastic gradients tend to be more accurate estimators of true gradients far away from the mode(s). In contrast, the MAP-tuned version of FlyMC (§3.3) requires the fewest data evaluations when the chain is close to the MAP, since by design, the lower likelihood bounds are tightest there. Meanwhile, the untuned version of FlyMC tends to exhibit the opposite behavior.

Adaptive proposal distributions. The Metropolis-Hastings algorithm requires the user to specify a proposal distribution. Fixing proposal distribution can be problematic, because the behavior of MH is sensitive to the proposal distribution and can furthermore change as the chain converges. A common solution, employed *e.g.*, by Bardenet et al. [2014], is to use an adaptive MH scheme [Haario et al., 2001, Andrieu and Moulines, 2006]. These algorithms tune the proposal distribution during execution, using information from the samples as they are generated, in a way that provably converges asymptotically. Often, it is desirable for the proposal distribution to be close to the target. This motivates adaptive schemes that fit a distribution to the observed samples and use this fitted model as the proposal distribution. For example, a simple online procedure can update the mean μ and covariance Σ of a multidimensional Gaussian model,

$$\begin{aligned} \mu_{t+1} &= \mu_t + \gamma_{t+1}(\theta_{t+1} - \mu_t), \qquad t \geq 0, \\ \Sigma_{t+1} &= \Sigma_t + \gamma_{t+1}((\theta_{t+1} - \mu_t)(\theta_{t+1} - \mu_t)^\top - \Sigma_t), \end{aligned}$$

where t indexes the MH iterations and γ_{t+1} controls the speed with which the adaptation vanishes. An appropriate choice is $\gamma_t = t^{-\alpha}$ for $\alpha \in (1/2, 1]$. The tutorial by Andrieu and Thoms [2008] provides a review of this and other, more sophisticated, adaptive MH algorithms.

Combining methods. The subsampling-based methods in this chapter are conceptually modular, and some may be combined. For example, it might be of interest to consider a 'tunable' version of FlyMC that achieves even greater computational efficiency at the cost of its original exactness. For example, we might use an adaptive subsampling scheme (§3.2) to evaluate only a subset of terms in Equation (3.41); this subset would need to represent terms corresponding to both possible values of z_n. As another example, Korattikara et al. [2014] suggest using adaptive subsampling as a way to ‘fix up’ SGLD. Recall that the original SGLD algorithm completely eliminates the MH test and blindly accepts all proposals, in order to avoid evaluating the full posterior. A reasonable compromise is to instead evaluate a fraction of the data within the adaptive subsampling framework, since this bounds the per-iteration error.

Unbiased likelihood estimators. A basic difficulty with building MCMC algorithms that operate on data mini-batches is that while it is straightforward to construct unbiased nonnegative Monte Carlo estimators of the *log* likelihood, as in Eq. (3.4), it is not clear how to construct such estimators for the likelihood itself, which is the quantity of direct interest in many standard sampler constructions like MH. To be explicit, the natural Monte Carlo estimator of the likelihood given by

$$\exp\left\{\frac{N}{m}\sum_{n=1}^{m}\log p(x_n^* \mid \theta)\right\} \tag{3.48}$$

is not unbiased. While it is possible to transform Equation (3.48) into an unbiased likelihood estimate, *e.g.*, using a Poisson estimator [Wagner, 1987, Papaspiliopoulos, 2009, Fearnhead et al., 2010], the resulting estimators are not always non-negative, which is a requirement to incorporate the estimator into a Metropolis-Hastings algorithm. In general, we cannot derive unweighted Monte Carlo estimators that are both unbiased and nonnegative [Jacob and Thiery, 2015, Lyne et al., 2015], though it is possible if, for example, we have unbiased estimators of the log likelihood with support in a fixed interval [Jacob and Thiery, 2015, Theorem 3.1] or if we have auxiliary variables as in FlyMC [Bardenet et al., 2015, Section 4.3]. *Pseudo-marginal MCMC* algorithms [Andrieu and Roberts, 2009],[2] first introduced by Lin et al. [2000], use such unbiased nonnegative likelihood estimators, often based on importance sampling [Beaumont, 2003] or particle filters [Andrieu et al., 2010, Doucet et al., 2015], to construct exact MCMC algorithms.

[2] Pseudo-marginal MCMC is also known as *exact-approximate sampling.*

4

Parallel and distributed MCMC

MCMC procedures that take advantage of parallel computing resources form another broad approach to scaling Bayesian inference. Because the computational requirements of inference often scale with the amount of data involved, and because large datasets may not even fit on a single machine, these approaches often leverage data parallelism. Some also leverage model parallelism, distributing latent variables in the model across many machines. In this chapter we consider several approaches to scaling MCMC by exploiting parallel computation, either by adapting classical MCMC algorithms or by defining new simulation dynamics that are inherently parallel.

One way to use parallel computing resources is to run multiple sequential MCMC algorithms at once. However, running identical chains in parallel does not reduce the transient bias in MCMC estimates of posterior expectations, though it would reduce their variance. Instead of using parallel computation only to collect more MCMC samples and thus reduce only estimator variance without improving transient bias, it is often preferable to use computational resources to speed up the simulation of the chain itself. Section 4.1 surveys several methods that use parallel computation to speed up the execution of MCMC procedures, including both basic methods and more recent ideas.

Alternatively, instead of adapting serial MCMC procedures to exploit parallel resources, another approach is to design new approximate algorithms that are inherently parallel. Section 4.2 summarizes some recent ideas for simulations that can be executed in a parallel manner and have their results aggregated or corrected to represent posterior samples. Some additional parallel inference strategies that can involve MCMC inference within an expectation propagation (EP) framework are also described in Section 5.3.1.

4.1 Parallelizing standard MCMC algorithms

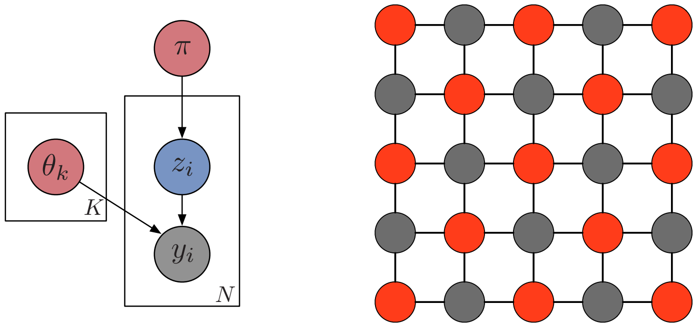
An advantage to parallelizing standard MCMC algorithms is that they retain their theoretical guarantees and analyses. Indeed, a common goal is to produce identical samples under serial and parallel execution, so that parallel resources enable speedups without introducing new approximations. This section first summarizes some basic opportunities for parallelism in MCMC and then surveys the speculative execution framework for MH.

4.1.1 Conditional independence and graph structure

The MH algorithm has a straightforward opportunity for parallelism. In particular, if the target posterior can be written as

$$\pi(\theta \,|\, \mathbf{x}) \propto \pi_0(\theta)\pi(\mathbf{x} \,|\, \theta) = \pi_0(\theta) \prod_{n=1}^{N} \pi(x_n \,|\, \theta), \tag{4.1}$$

then when the number of likelihood terms N is large it may be beneficial to parallelize the evaluation of the product of likelihoods. The communication between processors is limited to transmitting the value of the parameter and the scalar values of likelihood products. This basic parallelization, which naturally fits in a bulk synchronous parallel (BSP) computational model, exploits conditional independence in the probabilistic model, namely that the data are independent given the parameter.



(a) A mixture model

(b) An undirected grid

Figure 4.1: Graphical models and graph colorings can expose opportunities for parallelism in Gibbs samplers. (a) In this directed graphical model for a discrete mixture, each label (red) can be sampled in parallel conditioned on the parameters (blue) and data (gray) and similarly each parameter can be resampled in parallel conditioned on the labels. (b) This undirected grid has a classical "red-black" coloring, emphasizing that the variables corresponding to red nodes can be resampled in parallel given the values of black nodes and vice-versa.

Gibbs sampling algorithms can exploit more fine-grained conditional independence structure, and are thus a natural fit for graphical models which express such structure. Given a graphical model and a corresponding graph coloring with K colors that partitions the set of random variables into K groups, the random variables in each color group can be resampled in parallel while conditioning on the values in the other $K - 1$ groups [Gonzalez et al., 2011]. When models have a hierarchical exponential family structure, conditioning on a few variables can lead to many disconnected subgraphs, corresponding to inference problems to be solved separately and to have their sufficient statistics aggregated; this structure naturally lends itself to Map-Reduce parallelization [Pradier et al., 2014]. Thus graphical models provide a natural perspective on opportunities for parallelism. See Figure 4.1 for some examples.

These opportunities for parallelism, while powerful in some cases, are limited by the fact that they require frequent global synchronization and communication. Indeed, at each iteration it is often the case that every element of the dataset is read by some processor and many processors must mutually communicate. The methods we survey in the remainder of this chapter aim to mitigate these limitations by adjusting the allocation of parallel resources or by reducing communication.

4.1.2 Speculative execution and prefetching

Another class of parallel MCMC algorithms use speculative parallel execution to accelerate individual chains. This idea is called *prefetching* in some of the literature and appears to have received only limited attention.

As shown in Algorithm 1, the body of a serial MH implementation is a loop containing a single conditional statement and two associated branches. We can thus view the possible execution paths as a binary tree, illustrated in Figure 4.2. The vanilla version of parallel prefetching speculatively evaluates all paths in this binary tree on parallel processors [Brockwell, 2006]. A serial simulation corresponds to exactly one of these paths, so with J processors this approach achieves a speedup of $\log_2 J$ with respect to single core execution, ignoring communication and bookkeeping overheads.

Naïve prefetching can be improved by observing that the two branches in Algorithm 1 are not taken with equal probability. On average, the left-most branch, corresponding to a sequence of rejected proposals, tends to be more probable; the classic result for the optimal MH acceptance rate is 0.234 [Roberts et al., 1997], so most prefetching scheduling policies have been built around the expectation of rejection. Let $\alpha \leq 0.5$ be the expected probability of accepting a proposal. Byrd et al. [2008] introduced a procedure, called *speculative moves*, that speculatively evaluates only along the “reject” branch of the binary tree; in Figure 4.2, this corresponds to the left-most branch. In each round of their algorithm, only the first k out of $J - 1$ extra cores perform useful work, where k is the number of rejected proposals before the first accepted proposal, starting from the root of the tree. The expected

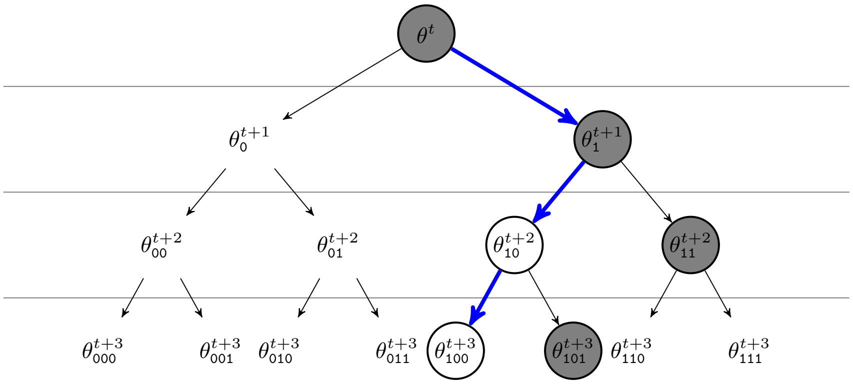


Figure 4.2: Schematic of a MH simulation superimposed on the binary tree of all possible chains. Nodes at depth d correspond to iteration $t + d$, where the root is at depth 0. Branching to the right indicates that the proposal is accepted, and branching to the left indicates that the proposal is rejected. Thick blue arrows highlight one simulated chain starting at the root θ^t; the first proposal is accepted and the next two are rejected, yielding as output: $\theta_1^{t+1}, \theta_{10}^{t+2}, \theta_{100}^{t+3}$. Each subscript is a sequence, of length d, of 0's and 1's, corresponding to the history of rejected and accepted proposals with respect to the root. Filled circles indicate states where the target density is evaluated during simulation. Those not on the chain's path correspond to rejected proposals. Their siblings are open circles on this path; since each is a copy of its parent, the target density does not need to be reevaluated to compute the next transition.

speedup is then:

$$1 + \mathbb{E}(k) < 1 + \sum_{k=0}^{\infty} k(1-\alpha)^k \alpha < 1 + \frac{1-\alpha}{\alpha} = \frac{1}{\alpha}\,.$$

Note that the first term on the left is due to the core at the root of the tree, which always performs useful computation in the prefetching scheme. When $\alpha = 0.23$, this scheme yields a maximum expected speedup of about 4.3; it achieves an expected speedup of about 4 with 16 cores. If only a few cores are available, this may be a reasonable policy, but if many cores are available, their work is essentially wasted. In contrast, the naïve prefetching policy achieves speedup that grows as the log of the number of cores. Byrd et al. [2010] later considered the special case where the evaluation of the likelihood function occurs on two timescales, slow and fast. They call this method *speculative chains*; it modifies the speculative moves approach so that whenever the evaluation of the likelihood function is slow, any available cores are used to speculatively evaluate the subsequent chain, assuming the slow step resulted in an accept.

Strid [2010] extends the naïve prefetching scheme to allocate cores according to the optimal "tree shape" with respect to various assumptions about the probability of rejecting a proposal, *i.e.*, by greedily allocating cores to nodes that maximize the depth of speculative computation expected to be correct. The author presents both static and dynamic prefetching schemes. The static scheme assumes a fixed acceptance rate; versions of this were proposed earlier in the context of simulated annealing [Witte et al., 1991]. The dynamic scheme estimates acceptance probabilities, for example, at each level of the tree by drawing empirical MH samples, or at each branch in the tree by computing $\min\{\beta, r\}$ where β is a constant ($e.g.$, $\beta = 1$) and r is an estimate of the MH ratio based on a fast approximation to the target function. Strid also proposes using the approximate target function to identify the single most likely path on which to perform speculative computation, and combines prefetching with other sources of parallelism to obtain a multiplicative effect.

In closely related work, *parallel predictive prefetching* makes efficient use of parallel resources by dynamically predicting the outcome of each MH test [Angelino et al., 2014]. In the case of Bayesian inference, these predictions can be constructed in the same manner as the approximate MH algorithms based on subsets of data, as discussed in Section 3.2.2. Furthermore, these predictions can be made in the context of an error model, *e.g.*, with the concentration inequalities used by Bardenet et al. [2014]. This yields a straightforward and rational mechanism for allocating parallel cores to computations most likely to fall along the true execution path. Algorithms 9 and 10 sketch pseudocode for an implementation of parallel predictive prefetching that follows a master-worker pattern. See the dissertation by Angelino [2014] for a formal description of the algorithm and implementation details.

Algorithm 9 Parallel predictive prefetching master process

```
repeat
    Receive message from worker $j$
    if worker $j$ wants work then
        Find highest utility node $\rho$ in tree with work left to do
        Send worker $j$ the computational state of $\rho$
    else if message contains state $\theta_\rho$ at proposal node $\rho$ then
        Record state $\theta_\rho$ at $\rho$
    else if message contains update at $\rho$ then
        Update estimate of $\pi(\theta_\rho \,|\, \mathbf{x})$ at $\rho$
        for node $\alpha$ in $\{\rho$ and its descendants$\}$ do
            Update utility of $\alpha$
            if utility of $\alpha$ below threshold and worker $k$ at $\alpha$ then
                Send worker $k$ message to stop current computation
        if posterior computation at $\rho$ and its parent complete then
            Know definitively whether to accept or reject $\theta_\rho$
            Delete subtree corresponding to branch not taken
            if node $\rho$ is the root's child then
                repeat
                    Trim old root so that new root points to child
                    Output state at root, the next state in the chain
                until posterior computation at root's child incomplete
until master has output $T$ Metropolis–Hastings chain states
Terminate all worker processes
```

4.2 Defining new parallel dynamics

In this section we survey two ideas for performing inference using new parallel dynamics. These algorithms define new dynamics in the sense that their iterates do not form ergodic Markov chains which admit the posterior distribution as an invariant distribution, and thus they do not qualify as classical MCMC schemes. Instead, while some of the updates in these algorithms resemble standard MCMC updates, the overall dynamics are designed to exploit parallel and distributed computation. A unifying theme of these new methods is to perform local computa-

Algorithm 10 Parallel predictive prefetching worker process

```
repeat
    Send master request for work
    Receive work assignment at node $\rho$ from master
    if the corresponding state $\theta_\rho$ has not yet been generated then
        Generate proposal $\theta_\rho$
    repeat
        Advance the computation of $\pi(\theta_\rho \,|\, \mathbf{x})$
        Send update at $\rho$ to master
        if receive message to stop current computation then
            break
    until computation of $\pi(\theta_\rho \,|\, \mathbf{x})$ is complete
until terminated by master
```

tion on data while controlling the amount of global synchronization or communication.

One such family of ideas involves the definition of *subposteriors*, defined using only subsets of the full dataset. Inference in the subposteriors can be performed in parallel, and the results are then globally aggregated into an approximate representation of the full posterior. Because the synchronization and communication costs—as well as the approximation quality—are determined by the aggregation step, several such aggregation procedures have been proposed. In Section 4.2.1 we summarize some of these proposals.

Another class of parallel dynamics does not define independent subposteriors but instead, motivated by Gibbs sampling, focuses on simulating from local conditional distributions with out-of-date information. In standard Gibbs sampling, updates can be parallelized in models with conditional independence structure (Section 4.1), but without such structure the Gibbs updates may depend on the full dataset and all latent variables, and thus must be performed sequentially. These sequential updates can be especially expensive with large or distributed datasets. A natural approximation to consider is to run the same local Gibbs updates in parallel with out-of-date global information and only infrequent communication. While such a procedure loses the theoretical guarantees provided by standard Gibbs sampling analysis, some empirical and theoretical results are promising. We refer to this broad class of methods as *Hogwild* Gibbs algorithms, and we survey some particular algorithms and analyses in Section 4.2.2.

4.2.1 Aggregating from subposteriors

Suppose we want to divide the evaluation of the posterior across J parallel cores. We can divide the data into J partition elements, $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(J)}$, also called *shards*, and factor the posterior into J corresponding *subposteriors*, as

$$\pi(\theta \,|\, \mathbf{x}) = \prod_{j=1}^{J} \pi^{(j)}(\theta \,|\, \mathbf{x}^{(j)}), \tag{4.2}$$

where

$$\pi^{(j)}(\theta \,|\, \mathbf{x}^{(j)}) = \pi_0(\theta)^{1/J} \prod_{x \in \mathbf{x}^{(j)}} \pi(x \,|\, \theta), \quad j = 1, \dots, J. \tag{4.3}$$

The contribution from the original prior is down-weighted so that the posterior is equal to the product of the J subposteriors, *i.e.*, $\pi(\theta \,|\, \mathbf{x}) = \prod_{j=1}^{J} \pi^{(j)}(\theta \,|\, \mathbf{x}^{(j)})$. Note that a subposterior is not the same as the posterior formed from the corresponding partition, *i.e.*,

$$\pi^{(j)}(\theta \,|\, \mathbf{x}^{(j)}) \neq \pi(\theta \,|\, \mathbf{x}^{(j)}) = \pi_0(\theta) \prod_{x \in \mathbf{x}^{(j)}} \pi(x \,|\, \theta). \tag{4.4}$$

Embarrassingly parallel consensus of subposteriors

Once a large dataset has been partitioned across multiple machines, a natural alternative is to try running MCMC inference on each partition element separately and in parallel. This yields samples from each subposterior in Equation 4.3, but there is no obvious choice for how to combine them in a coherent fashion to form approximate samples of the full posterior. In this section, we survey various proposals for forming such a *consensus* solution from the subposterior samples. Algorithm 11 outlines the structure of consensus strategies for embarrassingly parallel posterior sampling. This terminology, used by Huang and Gelman

Algorithm 11 Embarrassingly parallel consensus of subposteriors

```
Input: Initial state $\theta_0$, number of samples $T$, data partitions $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(J)}$, subposteriors $\pi^{(1)}(\theta \mid \mathbf{x}^{(1)}), \dots, \pi^{(J)}(\theta \mid \mathbf{x}^{(J)})$
Output: Approximate samples $\hat{\theta}_1, \dots, \hat{\theta}_T$
for $j = 1, 2, \dots, J$ in parallel do
    Initialize $\theta_{j,0}$
    for $t = 1, 2, \dots, T$ do
        Simulate MCMC sample $\theta_{j,t}$ from subposterior $\pi^{(j)}(\theta \mid \mathbf{x}^{(j)})$
    Collect $\theta_{j,1}, \dots, \theta_{j,T}$
$\hat{\theta}_1, \dots, \hat{\theta}_T \leftarrow$ ConsensusSamples($\{\theta_{j,1}, \dots, \theta_{j,T}\}_{j=1}^{J}$)
```

[2005] and Scott et al. [2016], invokes related notions of consensus, notably those that have existed for decades in the optimization literature on parallel algorithms in decentralized or distributed settings. We discuss this topic briefly in Section 4.2.1.

Below, we present two recent consensus strategies for combining subposterior samples, through weighted averaging and density estimation, respectively. The earlier report by Huang and Gelman [2005] proposes four consensus strategies, based either on normal approximations or importance resampling; the authors focus on Gibbs sampling for hierarchical models and do not evaluate any actual parallel implementations. Another consensus strategy is the recently proposed *variational consensus Monte Carlo* (VCMC) algorithm, which casts the consensus problem within a variational Bayes framework [Rabinovich et al., 2015].

Throughout this section, Gaussian densities provide a useful reference point and motivate some of the consensus strategies. Consider the jointly Gaussian model

$$\theta \sim \mathcal{N}(0, \Sigma_0) \tag{4.5}$$

$$\mathbf{x}^{(j)} \mid \theta \sim \mathcal{N}(\theta, \Sigma_j). \tag{4.6}$$

The joint density is:

$$\begin{aligned} p(\theta, \mathbf{x}) &= p(\theta) \prod_{j=1}^{J} p(\mathbf{x}^{(j)} \mid \theta) \\ &\propto \exp\left\{-\frac{1}{2}\theta^\top \Sigma_0^{-1}\theta\right\} \prod_{j=1}^{J} \exp\left\{-\frac{1}{2}\left(\mathbf{x}^{(j)} - \theta\right)^\top \Sigma_J^{-1}\left(\mathbf{x}^{(j)} - \theta\right)\right\} \\ &\propto \exp\left\{-\frac{1}{2}\theta^\top\left(\Sigma_0^{-1} + \sum_{j=1}^{J}\Sigma_j^{-1}\right)\theta + \left(\sum_{j=1}^{J}\Sigma_j^{-1}\mathbf{x}^{(j)}\right)^\top \theta\right\}. \end{aligned}$$

Thus the posterior is Gaussian:

$$\theta \mid \mathbf{x} \sim \mathcal{N}(\mu, \Sigma), \tag{4.7}$$

where

$$\Sigma = \left(\Sigma_0^{-1} + \sum_{j=1}^{J}\Sigma_j^{-1}\right)^{-1} \tag{4.8}$$

$$\mu = \Sigma\left(\sum_{j=1}^{J}\Sigma_j^{-1}\mathbf{x}^{(j)}\right). \tag{4.9}$$

To arrive at an expression for the subposteriors, we begin by factoring the joint distribution into an appropriate product:

$$p(\theta, \mathbf{x}) \propto \prod_{j=1}^{J} f_j(\theta), \tag{4.10}$$

where

$$\begin{aligned} f_j(\theta) &= p(\theta)^{1/J} p(\mathbf{x}^{(j)} \mid \theta) \\ &= \exp\left\{-\frac{1}{2}\theta^\top(\Sigma_0^{-1}/J)\theta\right\}\exp\left\{-\frac{1}{2}\left(\mathbf{x}^{(j)} - \theta\right)^\top \Sigma_j^{-1}\left(\mathbf{x}^{(j)} - \theta\right)\right\} \\ &\propto \exp\left\{-\frac{1}{2}\theta^\top\left(\Sigma_0^{-1}/J + \Sigma_j^{-1}\right)\theta + \left(\Sigma_j^{-1}\mathbf{x}^{(j)}\right)^\top\theta\right\}. \end{aligned}$$

Thus the subposteriors are also Gaussian:

$$\theta_j \sim \mathcal{N}\left(\tilde{\mu}_j, \tilde{\Sigma}_j\right) \propto f_j(\theta) \tag{4.11}$$

where

$$\tilde{\Sigma}_j = \left(\Sigma_0^{-1}/J + \Sigma_j^{-1}\right)^{-1} \tag{4.12}$$

$$\tilde{\mu}_j = \left(\Sigma_0^{-1}/J + \Sigma_j^{-1}\right)^{-1} \left(\Sigma_j^{-1}\mathbf{x}^{(j)}\right). \tag{4.13}$$

Weighted averaging of subposterior samples

One approach is to combine the subposterior samples via weighted averaging [Scott et al., 2016]. For simplicity, we assume that we obtain T samples in $\mathbb{R}^d$ from each subposterior, and let $\{\theta_{j,t}\}_{t=1}^T$ denote the samples from the jth subposterior. The goal is to construct T *consensus posterior* samples $\{\hat{\theta}_t\}_{t=1}^T$, that (approximately) represent the full posterior, from the JT subposterior samples, where each $\hat{\theta}_t$ combines subposterior samples $\{\theta_{j,t}\}_{j=1}^J$. We associate with each subposterior j a matrix $W_j \in \mathbb{R}^{d\times d}$ and assume that each consensus posterior sample is a weighted[1] average:

$$\hat{\theta}_t = \sum_{j=1}^J W_j \theta_{j,t}\,. \tag{4.14}$$

The challenge now is to design an appropriate set of weights.

Following Scott et al. [2016], we consider the special case of Gaussian subposteriors, as in Equation 4.11. Our presentation is slightly different, as we also account for the effect of having a prior. We also drop the subscript t from our notation for simplicity. Let $\{\theta_j\}_{j=1}^J$ be a set of draws from the J subposteriors. Each θ_j is an independent Gaussian and thus $\hat{\theta} = \sum_{j=1}^J W_j\theta_j$ is Gaussian. From Equation 4.13, its mean is

$$\begin{aligned}\mathbb{E}[\hat{\theta}] = \sum_{j=1}^J W_j \mathbb{E}[\theta_j] &= \sum_{j=1}^J W_j \tilde{\mu}_j \\ &= \sum_{j=1}^J W_j \left(\Sigma_0^{-1}/J + \Sigma_j^{-1}\right)^{-1} \left(\Sigma_j^{-1}\mathbf{x}^{(j)}\right). \end{aligned} \tag{4.15}$$

[1] Our notation differs slightly from that of Scott et al. [2016] in that our weights W_j are normalized.

Thus, if we choose

$$W_j = \Sigma\left(\Sigma_0^{-1}/J + \Sigma_j^{-1}\right) = \left(\Sigma_0^{-1} + \sum_{j=1}^{J} \Sigma_j^{-1}\right)^{-1}\left(\Sigma_0^{-1}/J + \Sigma_j^{-1}\right) \quad (4.16)$$

where Σ is the posterior covariance in Equation 4.8, then

$$\mathbb{E}[\hat{\theta}] = \Sigma\left(\sum_{j=1}^{J} \Sigma_j^{-1}\mathbf{x}^{(j)}\right) = \mu, \quad (4.17)$$

where μ is the posterior mean in Equation 4.9. A similar calculation shows that $\mathrm{Cov}(\hat{\theta}) = \Sigma$.

Thus for the Gaussian model, $\hat{\theta}$ is distributed according to the posterior distribution, indicating that Equation 4.16 gives the appropriate weights. Each weight matrix W_j is a function of Σ_0, the prior covariance, and the subposterior covariances $\{\Sigma_j\}_{j=1}^{J}$. We can form a Monte Carlo estimate of each Σ_j using the empirical sample covariance $\bar{\Sigma}_j$. Algorithm 12 summarizes this consensus approach with weighted averaging. While this weighting is optimal in the Gaussian setting, Scott et al. [2016] show it to be effective in some non-Gaussian models. Scott et al. [2016] also suggest weighting each dimension of a sample θ by the reciprocal of its marginal posterior variance, effectively restricting the weight matrices W_j to be diagonal.

Subposterior density estimation

Another consensus strategy relies on density estimation [Neiswanger et al., 2014]. A related approach, developed in Minsker et al. [2014], forms a posterior distribution estimate out of weighted samples by computing a median of subset posteriors, though we do not discuss it here. First, use the subposterior samples to separately fit a density estimator, $\tilde{\pi}^{(j)}(\theta \,|\, \mathbf{x}^{(j)})$, to each subposterior. The product of these density estimators then represents a density estimator for the full posterior target, *i.e.*,

$$\pi(\theta \,|\, \mathbf{x}) \approx \tilde{\pi}(\theta \,|\, \mathbf{x}) = \prod_{j=1}^{J} \tilde{\pi}^{(j)}(\theta \,|\, \mathbf{x}^{(j)}) \,. \quad (4.18)$$

Algorithm 12 Consensus of subposteriors via weighted averaging.

```
Parameters: Prior covariance $\Sigma_0$
function ConsensusSamples($\{\theta_{j,1}, \dots, \theta_{j,T}\}_{j=1}^{J}$)
    for $j = 1, 2, \dots, J$ do
        $\bar{\Sigma}_j \leftarrow$ Sample covariance of $\{\theta_{j,1}, \dots, \theta_{j,T}\}$
    $\Sigma \leftarrow \left( \Sigma_0^{-1} + \sum_{j=1}^{J} \bar{\Sigma}_j^{-1} \right)^{-1}$
    for $j = 1, 2, \dots, J$ do
        $W_j \leftarrow \Sigma \left( \Sigma_0^{-1}/J + \bar{\Sigma}_j^{-1} \right)$        ▷ Compute weight matrices
    for $t = 1, 2, \dots, T$ do
        $\hat{\theta}_t \leftarrow \sum_{j=1}^{J} W_j \theta_{j,t}$        ▷ Compute weighted averages
    return $\hat{\theta}_1, \dots, \hat{\theta}_T$
```

Finally, one can sample from this posterior density estimator using MCMC; ideally, this density is straightforward to obtain and sample. In general, however, density estimation can yield complex models that are not amenable to efficient sampling.

Neiswanger et al. [2014] explore three density estimation approaches of various complexities. Their first approach assumes a parametric model and is therefore approximate. Specifically, they fit a Gaussian to each set of subposterior samples, yielding

$$\tilde{\pi}(\theta \,|\, \mathbf{x}) = \prod_{j=1}^{J} \mathcal{N}(\bar{\mu}_j, \bar{\Sigma}_j), \tag{4.19}$$

where $\bar{\mu}_j$ and $\bar{\Sigma}_j$ are the empirical mean and covariance, respectively, of the samples from the jth subposterior. This product of Gaussians

Algorithm 13 Consensus of subposteriors via fits to Gaussians.

```
function ConsensusSamples($\{\theta_{j,1}, \dots, \theta_{j,T}\}_{j=1}^J$)
    for $j = 1, 2, \dots, J$ do
        $\bar{\mu}_j \leftarrow$ Sample mean of $\{\theta_{j,1}, \dots, \theta_{j,T}\}$
        $\bar{\Sigma}_j \leftarrow$ Sample covariance of $\{\theta_{j,1}, \dots, \theta_{j,T}\}$
    $\hat{\Sigma}_J \leftarrow \left( \sum_{j=1}^J \bar{\Sigma}_j^{-1} \right)^{-1}$    ▷ Covariance of product of Gaussians
    $\hat{\mu}_J \leftarrow \hat{\Sigma}_J \left( \sum_{j=1}^J \bar{\Sigma}_j^{-1} \bar{\mu}_j \right)$    ▷ Mean of product of Gaussians
    for $t = 1, 2, \dots, T$ do
        $\hat{\theta}_t \sim \mathcal{N}(\hat{\mu}_J, \hat{\Sigma}_J)$    ▷ Sample from fitted Gaussian
    return $\hat{\theta}_1, \dots, \hat{\theta}_T$
```

simplifies to a single Gaussian $\mathcal{N}(\hat{\mu}_J, \hat{\Sigma}_J)$, where

$$\hat{\Sigma}_J = \left(\sum_{j=1}^J \bar{\Sigma}_j^{-1} \right)^{-1} \tag{4.20}$$

$$\hat{\mu}_J = \hat{\Sigma}_J \left(\sum_{j=1}^J \bar{\Sigma}_j^{-1} \bar{\mu}_j \right). \tag{4.21}$$

These parameters are straightforward to compute and the overall density estimate can be sampled with reasonable efficiency and even in parallel, if desired. Algorithm 13 summarizes this consensus strategy based on fits to Gaussians.

In the case when the model is jointly Gaussian, the parametric density estimator we form is $\mathcal{N}(\hat{\mu}_J, \hat{\Sigma}_J)$, with $\hat{\mu}_J$ and $\hat{\Sigma}_J$ given in Equations 4.21 and 4.20, respectively. In this special case, the estimator exactly represents the Gaussian posterior. However, recall that we could have instead written the exact posterior directly as $\mathcal{N}(\mu, \Sigma)$, where μ and Σ are in Equations 4.9 and 4.8, respectively. Thus computing the density estimator is about as expensive as computing the exact posterior, requiring about J local matrix inversions.

The second approach proposed by Neiswanger et al. [2014] is to use a nonparametric kernel density estimate (KDE) for each subposterior. Suppose we obtain T samples $\{\theta_{j,t}\}_{t=1}^{T}$ from the jth subposterior, then its KDE with bandwidth parameter h has the following functional form:

$$\tilde{\pi}^{(j)}(\theta \,|\, \mathbf{x}^{(j)}) = \frac{1}{T}\sum_{t=1}^{T} \frac{1}{h^d} K\left(\frac{\|\theta - \theta_{j,t}\|}{h}\right), \tag{4.22}$$

i.e., the KDE is a mixture of T kernels, each centered at one of the samples. If we use T samples from each subposterior, then the density estimator for the full posterior is a complicated function with T^J terms, since it the a product of J such mixtures, and is therefore very challenging to sample from. Neiswanger et al. [2014] use a Gaussian KDE for each subposterior, and from this derive a density estimator for the full posterior that is a mixture of T^J Gaussians with unnormalized mixture weights. They also consider a third, semi-parametric approach to density estimation given by the product of a parametric (Gaussian) model and a nonparametric (Gaussian KDE) correction. As the number of samples $T \to \infty$, the nonparametric and semi-parametric density estimates exactly represent the subposterior densities and are therefore asymptotically exact. Unfortunately, their complex mixture representations grow exponentially in size, rendering them somewhat unwieldy in practice.

Weierstrass samplers

The consensus strategies surveyed so far are embarrassingly parallel. These methods obtain samples from each subposterior independently and in parallel, and from these attempt to construct samples that (approximately) represent the posterior post-hoc. The methods in this section proceed similarly, but introduce some amount of information sharing between the parallel samplers. This communication pattern is reminiscent of the alternating direction method of multipliers (ADMM) algorithm for parallel convex optimization; for a detailed treatment of ADMM, see the review by Boyd et al. [2011].

Weierstrass samplers [Wang and Dunson, 2013] are named for the Weierstrass transform [Weierstrass, 1885]: for $h > 0$, we write

$$W_h f(\theta) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}h} \exp\left\{-\frac{(\theta-\xi)^2}{2h^2}\right\} f(\xi) d\xi. \tag{4.23}$$

The transformed function $W_h f(\theta)$ is the convolution of a one-dimensional function $f(\theta)$ with a Gaussian density of standard deviation h, and so converges pointwise to $f(\theta)$ as $h \to 0$,

$$\lim_{h\to 0} W_h f(\theta) = \int_{-\infty}^{\infty} \delta(\theta-\xi) f(\xi) d\xi = f(\theta),$$

where $\delta(\tau)$ is the Dirac delta function. For $h > 0$, $W_h f(\theta)$ can be thought of as a smoothed approximation to $f(\theta)$. Equivalently, if $f(\theta)$ is the density of a random variable θ, then $W_h f(\theta)$ is the density of a noisy measurement of θ, where the noise is an additive Gaussian with zero mean and standard deviation h.

Wang and Dunson [2013] analyzes a more general class of Weierstrass transforms by defining a multivariate version and also allowing non-Gaussian kernels:

$$W_h^{(K)} f(\theta_1, \dots, \theta_d) = \int_{-\infty}^{\infty} f(\xi_1, \dots, \xi_d) \prod_{i=1}^{d} h_i^{-1} K_i\left(\frac{\theta_i - \xi_i}{h_i}\right) d\xi_i.$$

For simplicity, we restrict our attention to the one-dimensional Weierstrass transform.

Weierstrass samplers use Weierstrass transforms on subposterior densities to define an augmented model. Let $f_j(\theta)$ denote the j-th subposterior,

$$f_j(\theta) = \pi^{(j)}(\theta \,|\, \mathbf{x}^{(j)}) = \pi_0(\theta)^{1/J} \prod_{x \in \mathbf{x}^{(j)}} \pi(x \,|\, \theta), \tag{4.24}$$

so that the full posterior can be approximated as

$$\begin{aligned}\pi(\theta \,|\, \mathbf{x}) \propto \prod_{j=1}^{J} f_j(\theta) &\approx \prod_{j=1}^{J} W_h f_j(\theta) \\ &= \prod_{j=1}^{J} \int \frac{1}{\sqrt{2\pi}h} \exp\left\{ -\frac{(\theta - \xi_j)^2}{2h^2} \right\} f_j(\xi_j) d\xi_j \\ &\propto \int \prod_{j=1}^{J} \exp\left\{ -\frac{(\theta - \xi_j)^2}{2h^2} \right\} f_j(\xi_j) d\xi_j \,. \end{aligned} \tag{4.25}$$

The integrand of (4.25) defines the joint density of an augmented model that includes the $\xi = \{\xi_j\}_{j=1}^{J}$ as auxiliary variables:

$$\pi_h(\theta, \xi \,|\, \mathbf{x}) \propto \prod_{j=1}^{J} \exp\left\{ -\frac{(\theta - \xi_j)^2}{2h^2} \right\} f_j(\xi_j). \tag{4.26}$$

The posterior of interest can then be approximated by the marginal distribution of θ in the augmented model,

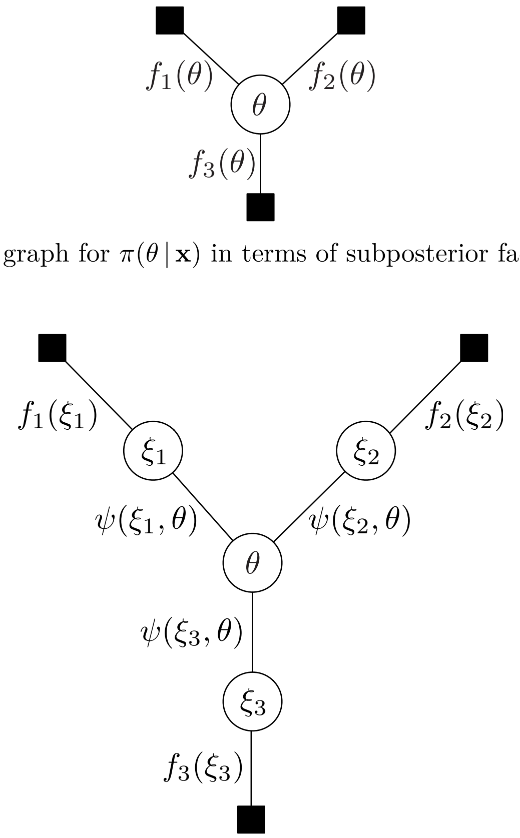
$$\int \pi_h(\theta, \xi \,|\, \mathbf{x}) d\xi \approx \pi(\theta \,|\, \mathbf{x}) \tag{4.27}$$

with pointwise equality in the limit as $h \to 0$. Thus by running MCMC in the augmented model, producing Markov chain samples of both θ and ξ, we can generate approximate samples of the posterior. Furthermore, the augmented model is more amenable to parallelization due to its conditional independence structure: conditioned on θ, the subposterior parameters ξ are rendered independent.

The same augmented model construction can be motivated without explicit reference to the Weierstrass transform of densities. Consider the factor graph model of the posterior in Figure 4.3a, which represents the definition of the posterior in terms of subposterior factors,

$$\pi(\theta \,|\, \mathbf{x}) \propto \prod_{j=1}^{J} f_j(\theta). \tag{4.28}$$

This model can be equivalently expressed as a model where each subposterior depends on an exact local copy of θ. That is, writing ξ_j as



(a) Factor graph for $\pi(\theta \,|\, \mathbf{x})$ in terms of subposterior factors $f_j(\theta)$.

(b) Factor graph for the Weierstrass augmented model $\pi(\theta, \xi \,|\, \mathbf{x})$.

Figure 4.3: Factor graphs defining the augmented model of the Weierstrass sampler.

the local copy of θ for subposterior j, the posterior is the marginal of a new augmented model given by

$$\pi(\theta, \xi \,|\, \mathbf{x}) \propto \prod_{j=1}^{J} f_j(\xi_j)\delta(\xi_j - \theta)\,. \tag{4.29}$$

This new model can be represented by the factor graph in Figure 4.3b, with potentials $\psi(\xi_j, \theta) = \delta(\xi_j - \theta)$. Finally, rather than taking the ξ_j to be exact local copies of θ, we can instead relax them to be noisy Gaussian measurements of θ:

$$\pi_h(\theta, \xi \,|\, \mathbf{x}) \propto \prod_{j=1}^{J} f_j(\xi_j)\psi_h(\xi_j, \theta) \tag{4.30}$$

$$\psi_h(\xi_j, \theta) = \exp\left\{-\frac{(\theta - \xi_j)^2}{2h^2}\right\}. \tag{4.31}$$

Thus the potentials $\psi_h(\xi_j, \theta)$ enforce some consistency across the noisy local copies of the parameter but allow them to be decoupled, where the amount of decoupling depends on h. With smaller values of h the approximate model is more accurate, but the local copies are more coupled and hence sampling in the augmented model is less efficient.

We can construct a Gibbs sampler for the joint distribution $\pi(\theta, \xi \,|\, \mathbf{x})$ in Equation 4.25 by alternately sampling from $p(\theta \,|\, \xi)$ and $p(\xi_j \,|\, \theta, \mathbf{x}^{(j)})$, for $j = 1, \ldots, J$. It follows from Equation 4.25 that

$$p(\theta \,|\, \xi_1, \ldots, \xi_j, \mathbf{x}) \propto \prod_{j=1}^{J} \exp\left\{-\frac{(\theta^2 - 2\theta\xi_j)}{2h^2}\right\}. \tag{4.32}$$

Rearranging terms gives

$$p(\theta \,|\, \xi_1, \ldots, \xi_j, \mathbf{x}) \propto \exp\left\{-\frac{(\theta - \bar{\xi})^2}{2h^2/J}\right\}, \tag{4.33}$$

where $\bar{\xi} = J^{-1} \sum_{j=1}^{J} \xi_j$. The remaining Gibbs updates follow from Equation 4.25, which directly yields

$$p(\xi_j \,|\, \theta, \mathbf{x}^{(j)}) \propto \exp\left\{-\frac{(\theta - \xi_j)^2}{2h^2}\right\} f_j(\xi_j), \quad j = 1, \ldots, J. \tag{4.34}$$

Algorithm 14 Weierstrass Gibbs sampling. For simplicity, $\theta \in \mathbb{R}$.

```
**Input:** Initial state $\theta_0$, number of samples $T$, data partitions $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(J)}$, subposteriors $f_1(\theta), \dots, f_J(\theta)$, tuning parameter $h$
**Output:** Samples $\theta_1, \dots, \theta_T$
Initialize $\theta_0$
for $t = 0, 1, \dots, T-1$ do
    Send $\theta_t$ to each processor
    for $j = 1, 2, \dots, J$ in parallel do
        $\xi_{j,t+1} \sim p(\xi_{j,t+1} \,|\, \theta_t, \mathbf{x}^{(j)}) \propto \mathcal{N}(\xi_{j,t+1} \,|\, \theta_t, h^2)\ f_j(\xi_{j,t+1})$
    Collect $\xi_{1,t+1}, \dots, \xi_{J,t+1}$
    $\bar{\xi}_{t+1} = \frac{1}{J} \sum_{j=1}^{J} \xi_{j,t+1}$
    $\theta_{t+1} \sim \mathcal{N}(\theta_{t+1} \,|\, \bar{\xi}_{t+1}, h^2/J)$
```

This Gibbs sampler allows for parallelism but requires communication at every round. A straightforward parallel implementation, shown in Algorithm 14, generates the updates for $\xi_1, \dots, \xi_J$ in parallel, but the update for θ depends on the most recent values of all the ξ_j. Wang and Dunson [2013] describes an approximate variant of the full Gibbs procedure that avoids frequent communication by only occasionally updating θ. In other efforts to exploit parallelism while avoiding communication, the authors propose alternate Weierstrass samplers based on importance sampling and rejection sampling, though the tradeoffs of these strategies require more empirical evaluation.

4.2.2 Hogwild Gibbs

Instead of designing new parallel algorithms from scratch, another approach is to take an existing MCMC algorithm and execute its updates in parallel at the expense of accuracy or theoretical guarantees. In particular, Hogwild Gibbs algorithms take a Gibbs sampling algorithm (§2.2.6) with interdependent sequential updates (*e.g.*, due to collapsed parameters or lack of graphical model structure) and simply run the updates in parallel anyway, using only occasional communication and out-of-date (stale) information from other processors. Because these strategies take existing algorithms and let the updates run 'hogwild' in the spirit of Hogwild! stochastic gradient descent in convex optimization [Recht et al., 2011], we refer to these methods as Hogwild Gibbs.

Similar approaches have a long history. Indeed, Gonzalez et al. [2011] attribute a version of this strategy, Synchronous Gibbs, to the original Gibbs sampling paper [Geman and Geman, 1984]. However, these strategies have seen renewed interest, particularly due to extensive empirical work on Approximate Distributed Latent Dirichlet Allocation (AD-LDA) [Newman et al., 2007, 2009, Asuncion et al., 2008, Liu et al., 2011, Ihler and Newman, 2012], which showed that running collapsed Gibbs sampling updates in parallel allowed for near-perfect parallelism without a loss in predictive likelihood performance. With the growing challenge of scaling MCMC both to not only big datasets but also big models, it is increasingly important to understand when and how these approaches may be useful.

In this section, we first define some variations of Hogwild Gibbs based on examples in the literature. Next, we survey the empirical results and summarize the current state of theoretical understanding.

Defining Hogwild Gibbs variants

Here we define some Hogwild Gibbs methods and related schemes, such as the stale synchronous parameter server. In particular, we consider bulk-synchronous parallel and asynchronous variations. We also fix some notation used for the remainder of the section.

For all of the Hogwild Gibbs algorithms, as with standard Gibbs sampling, we are given a collection of n random variables, $\{x_i : i \in [n]\}$ where $[n] \triangleq \{1, 2, \ldots, n\}$, and we assume that we can sample from the conditional distributions $x_i | x_{\neg i}$, where $x_{\neg i}$ denotes $\{x_j : j \neq i\}$. For the Hogwild Gibbs algorithms, we also assume we have K processors, each of which is assigned a set of variables on which to perform MCMC updates. We represent an assignment of variables to processors by fixing a partition $\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_K\}$ of $[n]$, so that the kth processor performs updates on the state values indexed by $\mathcal{I}_k$.

Algorithm 15 Bulk-synchronous parallel (BSP) Hogwild Gibbs

```
Input: Joint distribution over $x = (x_1, \ldots, x_n)$, partition $\{\mathcal{I}_1, \ldots, \mathcal{I}_K\}$
  of $\{1, 2, \ldots, n\}$, iteration schedule $q(t, k)$
  Initialize $\bar{x}^{(1)}$
  for $t = 1, 2, \ldots$ do
      for $k = 1, 2, \ldots, K$ in parallel do
          $\bar{x}_{\mathcal{I}_k}^{(t+1)} \leftarrow \textsc{LocalGibbs}(\bar{x}^{(t)}, \mathcal{I}_k, q(t, k))$
      Synchronize
  function LocalGibbs($\bar{x}$, $\mathcal{I}$, $q$)
      for $j = 1, 2, \ldots, q$ do
          for $i \in \mathcal{I}$ in order do
              $\bar{x}_i \leftarrow$ sample $x_i \mid x_{\neg i} = \bar{x}_{\neg i}$
      return $\bar{x}$
```

Bulk-synchronous parallel Hogwild Gibbs

A bulk-synchronous parallel (BSP) Hogwild Gibbs algorithm assigns variables to processors and alternates between performing parallel processor-local updates and global synchronization steps. During epoch t, the kth processor performs $q(t, k)$ MCMC updates, such as Gibbs updates, on the variables $\{x_i : i \in \mathcal{I}_k\}$ without communicating with the other processors; in particular, these updates are computed using out-of-date values for all $\{x_j : j \notin \mathcal{I}_k\}$. After all processors have completed their local updates, all processors communicate the updated state values in a global synchronization step and the system advances to the next epoch. We summarize this Hogwild Gibbs variant in Algorithm 15, in which the local MCMC updates are taken to be Gibbs updates.

Several special cases of the BSP Hogwild Gibbs scheme have been of interest. The Synchronous Gibbs scheme of Gonzalez et al. [2011] associates one variable with each processor, so that $|\mathcal{I}_k| = 1$ for each $k = 1, 2, \ldots, K$ (in which case we may take $q = 1$ since no local iterations are needed with a single variable). One may also consider the case where the partition is arbitrary and q is very large, in which case the local MCMC iterations may converge and exact block samples are drawn on each processor using old statistics from other processors for each outer iteration. Finally, note that setting $K = 1$ and $q(t, k) = 1$ reduces to standard Gibbs sampling on a single processor.

Asynchronous Hogwild Gibbs

Another Hogwild Gibbs pattern involves performing updates asynchronously. That is, processors might communicate only by sending messages to one another instead of by a global synchronization. Versions of this Hogwild Gibbs pattern has proven effective both for collapsed latent Dirichlet allocation topic model inference [Asuncion et al., 2008], and for Indian Buffet Process inference [Doshi-Velez et al., 2009]. A version was also explored in the Gibbs sampler of the Stale Synchronous Parameter (SSP) server of Ho et al. [2013], which placed an upper bound on the staleness of the entries of the state vector on each processor.

There are many possible communication strategies in the asynchronous setting, and so we follow a version of the random communication strategy employed by Asuncion et al. [2008]. In this approach, after performing some number of local updates, a processor sends its updated state information to a set of randomly-chosen processors and receives updates from other processors. The processor then updates its state representation and performs another round of local updates. A version of this asynchronous Hogwild Gibbs strategy is summarized in Algorithm 16.

Theoretical analysis

Despite its empirical successes, theoretical understanding of Hogwild Gibbs algorithms is limited. There are three settings in which some analysis has been offered: first, in a variant of AD-LDA, *i.e.*, Hogwild Gibbs applied to Latent Dirichlet Allocation models [Ihler and Newman, 2012]; second, in the jointly Gaussian case [Johnson et al., 2013, Johnson, 2014]; and third, in the discrete variable setting with sparse factor graph structure [De Sa et al., 2016].

Algorithm 16 Asynchronous Hogwild Gibbs

```
Initialize $\bar{x}^{(1)}$
for each processor $k = 1, 2, \ldots, K$ in parallel do
    for $t = 1, 2, \ldots$ do
        $\bar{x}_{\mathcal{I}_k}^{(t+1)} \leftarrow \textsc{LocalGibbs}(\bar{x}^{(t)}, \mathcal{I}_k, q(t, k))$
        Send $\bar{x}_{\mathcal{I}_k}^{(t+1)}$ to $K'$ randomly-chosen processors
        for each $k' \neq k$ do
            if update $\bar{x}_{\mathcal{I}_{k'}}$ received from processor $k'$ then
                $\bar{x}_{\mathcal{I}_{k'}}^{(t+1)} \leftarrow \bar{x}_{\mathcal{I}_{k'}}$
            else
                $\bar{x}_{\mathcal{I}_{k'}}^{(t+1)} \leftarrow \bar{x}_{\mathcal{I}_{k'}}^{(t)}$
```

The work of Ihler and Newman [2012] provides some understanding of the effectiveness of a variant of AD-LDA by bounding in terms of run-time quantities the one-step error probability induced by proceeding with sampling steps in parallel, thereby allowing an AD-LDA user to inspect the computed error bound after inference [Ihler and Newman, 2012, Section 4.2]. In experiments, the authors empirically demonstrate very small upper bounds on these one-step error probabilities, *e.g.*, a value of their parameter $\varepsilon = 10^{-4}$ meaning that at least 99.99% of samples are expected to be drawn just as if they were sampled sequentially. However, this per-sample error does not necessarily provide a direct understanding of the effectiveness of the overall algorithm because errors might accumulate over sampling steps; indeed, understanding this potential error accumulation is of critical importance in iterative systems. Furthermore, the bound is in terms of empirical run-time quantities, and thus it does not provide guidance on which other models the Hogwild strategy may be effective. Ihler and Newman [2012, Section 4.3] also provides approximate scaling analysis by estimating the order of the one-step bound in terms of a Gaussian approximation and some distributional assumptions.

The jointly Gaussian case is more tractable for analysis [Johnson et al., 2013, Johnson, 2014]. In particular, Johnson [2014, Theorem 7.6.6] shows that for the BSP Hogwild Gibbs process to be stable, *i.e.*, to form an ergodic Markov chain with a well-defined stationary distribution, for any variable partition and any iteration schedule it suffices for the model's joint Gaussian precision matrix to satisfy a generalized diagonal dominance condition. Because the precision matrix contains the coefficients of the log potentials in a Gaussian graphical model, the diagonal dominance condition captures the intuition that Hogwild Gibbs should be stable when variables do not interact too strongly. Johnson [2014, Proposition 7.6.8] gives a more refined condition for the case where the number of processor-local Gibbs iterations is large.

When a bulk-synchronous parallel Gaussian Hogwild Gibbs process defines an ergodic Markov chain and has a stationary distribution, Johnson [2014, Chapter 7] also provides an understanding of how that stationary distribution relates to the model distribution. Because both the model distribution and the Hogwild Gibbs process stationary distribution are Gaussian, accuracy can be measured in terms of the mean vector and covariance matrix. Proposition 7.6.1 of Johnson [2014] shows that the mean of a stable Gaussian Hogwild Gibbs process is always correct. Propositions 7.7.2 and 7.7.3 of Johnson [2014] identify a tradeoff in the accuracy of the process covariance matrix as a function of the number of processor-local Gibbs iterations: at least when the processor interactions are sufficiently weak, more processor-local iterations between synchronization steps increase the accuracy of the covariances among variables within each processor but decrease the accuracy of the covariances between variables on different processors. Johnson [2014, Proposition 7.7.4] also gives a more refined error bound as well as an inexpensive way to correct covariance estimates for the case where the number of processor-local Gibbs iterations is large.

Finally, De Sa et al. [2016] analyze asynchronous Hogwild Gibbs sampling on a discrete state space when the model has a sparse factor graph structure. Using a total influence condition to control the mutual dependence between variables and Dobrushin's condition, the authors bound some measures of bias of mixing time. However, their analysis (and indeed the asynchronous setting they consider) does not ensure the existence of a stationary distribution; instead, bias and mixing time are measured using the first time the distribution of the iterates comes sufficiently close to the target distribution. De Sa et al. [2016] also provide quantitative rate bounds for these measures of bias and mixing time, and show that these estimates closely track empirical simulations.

4.3 Summary

Many ideas for parallelizing MCMC have been proposed, exhibiting many tradeoffs. These ideas vary in generality, in faithfulness to the posterior, and in the parallel computation architectures for which they are best suited. Here we summarize the surveyed methods, emphasizing their relative strengths on these criteria. See Table 4.1 for an overview.

Simulating independent Markov chains Independent instances of serial MCMC algorithms can be run in an embarrassingly parallel manner, requiring only minimal communication between processors to ensure distinct initializations and to collect samples. This approach can reduce Monte Carlo variance by increasing the number of samples collected in any time budget, achieving an ideal parallel speedup, but does nothing to accelerate the warm-up period of the chains during which the transient bias is eliminated (see Section 2.2.4 and Chapter 6). That is, using parallel resources to run independent chains does nothing to improve mixing unless there is some mechanism for information sharing as in Nishihara et al. [2014]. In addition, running an independent MCMC chain on each processor requires each processor to access the full dataset, which may be problematic for especially large datasets. These considerations motivate both subposterior methods and Hogwild Gibbs.

Direct parallelization of standard updates Some MCMC algorithms applied to models with particular structure allow for straightforward parallel implementation. In particular, when the likelihood is factorized across data points, the computation of the Metropolis–Hastings acceptance probability can be parallelized. This strategy lends itself to a bulk-synchronous parallel (BSP) computational model. Parallelizing

Table 4.1: Summary of recent approaches to parallel MCMC.

	Parallel density evaluation (§4.1.1)	**Prefetching** (§4.1.2)	**Consensus** (§4.2.1)	**Weierstrass** (§4.2.1)	**Hogwild Gibbs** (§4.2.2)
Requirements	Conditional independence	None	Approximate factorization	Approximate factorization	Weak dependencies across processors
Parallel model	BSP	Speculative execution	MapReduce	BSP	BSP and asynchronous message passing variants
Communication	Each iteration	Master scheduling	Once	Tuneable	Tuneable
Design choices	None	Scheduling policy	Data partition, consensus algorithm	Data partition, synchronization frequency, Weierstrass h	Data partition, communication frequency
Computational overheads	None	Scheduling, bookkeeping	Consensus step	Auxiliary variable sampling	None
Approximation error	None	None	Depends on number of processors and consensus algorithm	Depends on number of processors and Weierstrass h	Depends on number of processors and staleness

MH in this way yields exact MCMC updates and can be effective at reducing the mixing time required by serial MH, but it requires a simple likelihood function and its implementation requires frequent synchronization and communication, mitigating parallel speedups unless the likelihood function is very expensive.

Gibbs sampling also presents an opportunity for direct parallelization for particular graphical model structures. In particular, given a graph coloring of the graphical model, variables corresponding to nodes assigned a particular color are conditionally mutually independent and can be updated in parallel without communication. However, frequent synchronization and significant communication can be required to transmit sampled values to neighbors after each update. Relaxing both the strict conditional independence requirements and synchronization requirements motivates Hogwild Gibbs.

Prefetching and speculative execution The prefetching algorithms studied in Section 4.1.2 use speculative execution to transform traditional (serial) Metropolis–Hastings into a parallel algorithm without incurring approximate updates or requiring any model structure. The implementation naturally follows a master-worker pattern, where the master allocates (possibly speculative) computational work, such as proposal generation or (partial) density evaluation, to worker processors. Ignoring overheads, basic prefetching algorithms achieve at least logarithmic speedup in the number of processors available. More sophisticated scheduling by the master, such as predictive prefetching [Angelino et al., 2014], can increase speedup significantly. While this method is very general and yields the same iterates as serial MH, the speedup can be limited.

Subposterior consensus and Weierstrass samplers Subposterior methods, such as the consensus Monte Carlo algorithms and the Weierstrass samplers of Section 4.2.1, allow for data parallelism and minimal communication because each subposterior Markov chain can be allocated to a processor and simulation can proceed independently. Communication is required only for final sample aggregation in consensus

Monte Carlo or the periodic resampling of the global parameter in the Weierstrass sampler. In consensus Monte Carlo, the quality of the approximate inference depends on both the effectiveness of the aggregation strategy and the extent to which dependencies in the posterior can be factorized into subposteriors. The Weierstrass samplers directly trade off approximation quality and the amount of decoupling between subposteriors.

The consensus Monte Carlo approach originated at Google [Scott et al., 2016] and naturally fits the MapReduce programming model, allowing it to be executed on large computational clusters. Recent work has extended consensus Monte Carlo and provides tools for designing simple consensus strategies [Rabinovich et al., 2015], but the generality and approximation quality of subposterior methods remain unclear. The Weierstrass sampler fits well into a BSP model.

Hogwild Gibbs Hogwild Gibbs of Section 4.2.2 also allows for data parallelism but avoids factorizing the posterior as in consensus Monte Carlo or instantiating coupled copies of a global parameter as in the Weierstrass sampler. Instead, processor-local sampling steps (such as local Gibbs updates) are performed with each processor treating other processors' states as fixed at stale values; processors can communicate updated states less frequently, either via synchronous or asynchronous communication. Hogwild Gibbs variants span a range of parallel computation paradigms from fully synchronous BSP to fully asynchronous message passing. While Hogwild Gibbs has proven effective in practice for several models, its applicability and approximation tradeoffs remain unclear.

4.4 Discussion

The methods studied here rely to different degrees on redundancy in the data. In particular, averaging subposterior samples relies on the subposteriors being relatively similar. Consider two extreme cases: if the subposteriors are identical then averaging their samples can produce reasonable results, but if the subposteriors have disjoint support then averaging their samples could produce iterates that look nothing like the true posterior. Furthermore, averaging may only make sense in continuous spaces that are closed under convex combinations; these averaging strategies do not apply to samples that take values in discrete spaces. Hence these subposterior methods may be most appropriate in the "tall data" regime, where the data are redundant and subposteriors can be expected to agree.

Performing density estimation on the subposteriors and forming the product of these densities avoids the problems with direct averaging, but may itself be computationally expensive and is unlikely to scale well with dimensionality. Indeed, applying this strategy to a Gaussian model provides no computational advantage to parallelization.

Unlike the subposterior methods, in Weierstrass samplers the processors try to account for the other processors' data by communicating through the shared global variable. In this way, the Weierstrass samplers are similar to the expectation propagation methods surveyed in the next chapter. However, because processors can affect each others' iterates only through the global variable bottleneck, their communication is limited. In addition, it is not clear how to extend the Weierstrass strategies to discrete variables.

Hogwild Gibbs algorithms also include influence between processors, and without an information bottleneck. Communication between processors is limited instead by an algorithm's update frequency. In addition, Hogwild Gibbs doesn't inherently rely on data redundancy: instead, it relies on variables not being too dependent across processors. For this reason, while Hogwild Gibbs has its own restrictions, it is a promising method for scaling out to big models with many latent variables rather than just the "tall data" regime. Hogwild Gibbs also readily applies to discrete variables.

5

Scaling variational algorithms

Variational inference is another paradigm for posterior inference in Bayesian models. Because variational methods pose inference as an optimization problem, ideas in scalable optimization can in principle yield scalable posterior inference algorithms. In this chapter we consider such algorithms both for mean field variational inference, which is often simply called variational Bayes, and for expectation propagation.

Variational inference is typically performed using a family of distributions that does not include the exact posterior, and as a result variational methods are inherently biased. In particular, tractable variational families typically provide only unimodal approximations, and often cannot represent some posterior correlations near particular modes. As a result, MCMC methods can in principle provide better approximations even when the Markov chain only explores a single mode in a reasonable number of iterations. However, while traditional MCMC is asymptotically unbiased, some of the scalable MCMC algorithms discussed in Chapters 3 and 4 are not, and it is unclear how to compare these biases to the biases in variational methods.

Despite its inherent bias, variational inference is widely used in machine learning because the computational advantage over MCMC can be significant, particularly when scaling inference to large datasets. The big data context may also inform the relative cost of performing inference in a constrained variational family rather than attempting to represent the posterior exactly: when the posterior is concentrated, a variational approximation, even a Gaussian approximation, may suffice [Bardenet et al., 2015]. While such questions ultimately need to be explored empirically on a case-by-case basis, the scalable variational inference methods surveyed in this chapter provide the tools for such an exploration.

In this chapter we summarize three patterns of scalable variational inference. First, in Section 5.1, we discuss the application of stochastic gradient optimization methods to mean field variational inference problems. Second, in Section 5.2, we describe an alternative approach that instead leverages the idea of incremental posterior updating to develop an inference algorithm with minibatch-based updates. Finally, in Section 5.3, we describe two scalable versions of expectation propagation (EP), one based on data parallelism and distributed computation and another based on stochastic optimization and minibatch updates.

5.1 Stochastic optimization and mean field methods

Stochastic gradient optimization is a powerful tool for scaling optimization algorithms to large datasets, and it has been applied to mean field variational inference problems to great effect. While many traditional algorithms for optimizing mean field objective functions, including both gradient-based and coordinate optimization methods, require re-reading the entire dataset in each iteration, the stochastic gradient framework allows each update to be computed with respect to minibatches of the dataset while providing very general asymptotic convergence guarantees.

In this section we first summarize the stochastic variational inference (SVI) framework of Hoffman et al. [2013], which applies to models with complete-data conjugacy. Next, we discuss alternatives and extensions which can handle more general models at the cost of updates with greater variance and, hence, slower convergence.

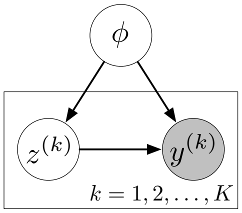


Figure 5.1: Prototypical graphical model for stochastic variational inference (SVI). The global latent variables are represented by ϕ and the local latent variables by $z^{(k)}$.

5.1.1 SVI for complete-data conjugate models

This section follows the development in Hoffman et al. [2013]. It depends on results from stochastic gradient optimization theory; see Section 2.5 for a review. For notational simplicity we consider each minibatch to consist of only a single observation; the generalization to minibatches of arbitrary sizes is immediate.

Many common probabilistic models are hierarchical: they can be written in terms of *global* latent variables (or parameters), *local* latent variables, and observations. That is, many models can be written as

$$p(\phi, z, y) = p(\phi) \prod_{k=1}^{K} p(z^{(k)} \mid \phi) p(y^{(k)} \mid z^{(k)}, \phi) \tag{5.1}$$

where ϕ denotes global latent variables, $z = \{z^{(k)}\}_{k=1}^{K}$ denotes local latent variables, and $y = \{y^{(k)}\}_{k=1}^{K}$ denotes observations. See Figure 5.1 for a graphical model. Given such a class of models, the mean field variational inference problem is to approximate the posterior $p(\phi, z \mid y)$ for fixed data y with a distribution of the form $q(\phi)q(z) = q(\phi)\prod_k q(z^{(k)})$ by finding a local minimum of the KL divergence from the approximating distribution to the posterior or, equivalently, finding a local maximum of the log marginal likelihood lower bound

$$\mathcal{L}[q(\phi)q(z)] \triangleq \mathbb{E}_{q(\phi)q(z)}\left[\log \frac{p(\phi, z, y)}{q(\phi)q(z)}\right] \leq \log p(y). \tag{5.2}$$

Hoffman et al. [2013] develops a stochastic gradient ascent algorithm for such models that leverages complete-data conjugacy. Gradients of $\mathcal{L}$ with respect to the parameters of $q(\phi)$ have a convenient form if we assume the prior $p(\phi)$ and each complete-data likelihood $p(z^{(k)}, y^{(k)} \,|\, \phi)$ are a conjugate pair of exponential family densities. That is, if we have

$$\log p(\phi) = \langle \eta_\phi,\ t_\phi(\phi) \rangle - \log Z_\phi(\eta_\phi) \tag{5.3}$$

$$\log p(z^{(k)}, y^{(k)} \,|\, \phi) = \langle \eta_{zy}(\phi),\ t_{zy}(z^{(k)}, y^{(k)}) \rangle - \log Z_{zy}(\eta_{zy}(\phi)) \tag{5.4}$$

then conjugacy identifies the statistic of the prior with the natural parameter and log partition function of the likelihood via

$$t_\phi(\phi) = (\eta_{zy}(\phi),\ -\log Z_{zy}(\eta_{zy}(\phi))\,, \tag{5.5}$$

so that

$$p(\phi, z^{(k)}, y^{(k)}) \propto \exp\left\{ \langle \eta_\phi + (t_{zy}(z^{(k)},\ y^{(k)}), 1), t_\phi(\phi) \rangle \right\}\,. \tag{5.6}$$

Conjugacy implies the optimal variational factor $q(\phi)$ has the same form as the prior; that is, without loss of generality we can write $q(\phi)$ in the same form as (5.3),

$$q(\phi) = \exp\left\{ \langle \widetilde{\eta}_\phi,\ t_\phi(\phi) \rangle - \log Z_\phi(\widetilde{\eta}_\phi) \right\}, \tag{5.7}$$

for some variational parameter $\widetilde{\eta}_\phi$.

Given this conjugacy structure, we can find a simple expression for the gradient of $\mathcal{L}$ with respect to the global variational parameter $\widetilde{\eta}_\phi$, optimizing out the local variational factor $q(z)$. That is, we write the variational objective over global parameters as

$$\mathcal{L}(\widetilde{\eta}_\phi) = \max_{q(z)} \mathcal{L}[q(\phi)q(z)]\,. \tag{5.8}$$

Writing the optimal parameters of $q(z)$ as $\widetilde{\eta}_z^*$, note that when $\widetilde{\eta}_z^*$ is partially optimized to a stationary point[1] of $\mathcal{L}$, so that $\frac{\partial \mathcal{L}}{\partial \widetilde{\eta}_z^*} = 0$ at $\widetilde{\eta}_z^*$, the chain rule implies that the gradient[2] with respect to the global

[1] More generally, when $\widetilde{\eta}_z$ is regularly constrained (typically the constraint set is linear [Wainwright and Jordan, 2008]), the same result holds because at $\widetilde{\eta}_z*$ the gradient of the objective is orthogonal to the feasible variations in $\widetilde{\eta}_z$.

[2] For a discussion of differentiability and smoothness issues that can arise when there is more than one optimizer, see Danskin [1967], Fiacco [1984, Section 2.4], and Bonnans and Shapiro [2000, Chapter 4]. Here we simply assume $\partial\widetilde{\eta}_z^*/\partial\widetilde{\eta}_\phi$ exists.

variational parameters simplifies:

$$\frac{\partial \mathcal{L}}{\partial \widetilde{\eta}_\phi}(\widetilde{\eta}_\phi) = \frac{\partial \mathcal{L}}{\partial \widetilde{\eta}_\phi}(\widetilde{\eta}_\phi, \widetilde{\eta}_z^*) + \frac{\partial \mathcal{L}}{\partial \widetilde{\eta}_z^*}\frac{\partial \widetilde{\eta}_z^*}{\partial \widetilde{\eta}_\phi}(\widetilde{\eta}_\phi, \widetilde{\eta}_z^*) \tag{5.9}$$

$$= \frac{\partial \mathcal{L}}{\partial \widetilde{\eta}_\phi}(\widetilde{\eta}_\phi, \widetilde{\eta}_z^*)\,. \tag{5.10}$$

Because a locally optimal local factor $q(z)$ can be computed with local mean field updates for a fixed value of the global variational parameter $\widetilde{\eta}_\phi$, we need only find an expression for the gradient $\nabla_{\widetilde{\eta}_\phi}\mathcal{L}(\widetilde{\eta}_\phi)$ in terms of the optimized local factors.

To find an expression for the gradient $\nabla_{\widetilde{\eta}_\phi}\mathcal{L}(\widetilde{\eta}_\phi)$ that exploits conjugacy structure, using (5.6) we can substitute

$$p(\phi, z, y) \propto \exp\left\{\langle \eta_\phi + \sum_k (t_{zy}(z^{(k)}, y^{(k)}), 1),\ t_\phi(\phi)\rangle\right\}, \tag{5.11}$$

into the definition of $\mathcal{L}$ in (5.2). Using the optimal form of $q(\phi)$, we have

$$\mathcal{L}(\widetilde{\eta}_\phi) = \mathbb{E}_{q(\phi)q(z)}\left[\langle \eta_\phi + \sum_k t_{zy}(z^{(k)}, y^{(k)}) - \widetilde{\eta}_\phi,\ t_\phi(\phi)\rangle\right]$$

$$+ \log Z_\phi(\widetilde{\eta}_\phi) + \text{const.} \tag{5.12}$$

$$= \langle \eta_\phi + \sum_k \mathbb{E}_{q(z^{(k)})}[t_{zy}(z^{(k)}, y^{(k)})] - \widetilde{\eta}_\phi,\ \mathbb{E}_{q(\phi)}[t_\phi(\phi)]\rangle$$

$$+ \log Z_\phi(\widetilde{\eta}_\phi) + \text{const}\,, \tag{5.13}$$

where the constant does not depend on $\widetilde{\eta}_\phi$. Using the identity for natural exponential families that

$$\nabla \log Z_\phi(\widetilde{\eta}_\phi) = \mathbb{E}_{q(\phi)}[t_\phi(\phi)], \tag{5.14}$$

we can write the same expression as

$$\mathcal{L}(\widetilde{\eta}_\phi) = \langle \eta_\phi + \sum_k \mathbb{E}_{q(z^{(k)})}[t_{zy}(z^{(k)}, y^{(k)})] - \widetilde{\eta}_\phi,\ \nabla \log Z_\phi(\widetilde{\eta}_\phi)\rangle$$

$$+ \log Z_\phi(\widetilde{\eta}_\phi) + \text{const}\,. \tag{5.15}$$

Thus we can compute the gradient of $\mathcal{L}(\widetilde{\eta}_\phi)$ with respect to the global variational parameters $\widetilde{\eta}_\phi$ as

$$\begin{aligned}\nabla_{\widetilde{\eta}_\phi}\mathcal{L}(\widetilde{\eta}_\phi) &= \langle \nabla^2 \log Z_\phi(\widetilde{\eta}_\phi),\ \eta_\phi + \sum_k \mathbb{E}_{q(z^{(k)})}[t_{zy}(z^{(k)}, y^{(k)})] - \widetilde{\eta}_\phi \rangle \\ &\qquad - \nabla \log Z_\phi(\widetilde{\eta}_\phi) + \nabla \log Z_\phi(\widetilde{\eta}_\phi) \qquad (5.16) \\ &= \langle \nabla^2 \log Z_\phi(\widetilde{\eta}_\phi),\ \eta_\phi + \sum_k \mathbb{E}_{q(z^{(k)})}[t_{zy}(z^{(k)}, y^{(k)})] - \widetilde{\eta}_\phi \rangle\end{aligned}$$

where the first two terms come from applying the product rule.

The matrix $\nabla^2 \log Z_\phi(\widetilde{\eta}_\phi)$ is the Fisher information of the variational family, since

$$-\mathbb{E}_{q(\phi)}\left[\nabla^2_{\widetilde{\eta}_\phi} \log q(\phi)\right] = \nabla^2 \log Z_\phi(\widetilde{\eta}_\phi). \qquad (5.17)$$

In the context of stochastic gradient ascent, we can cancel the multiplication by the matrix $\nabla^2 \log Z_\phi(\widetilde{\eta}_\phi)$ simply by choosing the sequence of positive definite matrices in Algorithm 4 to be $G^{(t)} \triangleq \nabla^2 \log Z_\phi(\widetilde{\eta}_\phi^{(t)})^{-1}$. This choice yields a stochastic *natural* gradient ascent algorithm [Amari and Nagaoka, 2007], where the updates are stochastic approximations to the natural gradient

$$\widetilde{\nabla}_{\widetilde{\eta}_\phi}\mathcal{L} = \eta_\phi + \sum_k \mathbb{E}_{q(z^{(k)})}[t_{zy}(z^{(k)}, y^{(k)})] - \widetilde{\eta}_\phi. \qquad (5.18)$$

Natural gradients effectively include a second-order quasi-Newton correction for local curvature in the variational family [Martens and Grosse, 2015, Martens, 2015], making the updates invariant to reparameterization of the variational family and thus often improving performance of the algorithm. More importantly, at least for the case of complete-data conjugate families considered here, natural gradient steps are in fact easier to compute than 'flat' gradient steps.

Therefore a stochastic natural gradient ascent algorithm on the global variational parameter $\widetilde{\eta}_\phi$ proceeds at iteration t by sampling a minibatch $y^{(k)}$ and taking a step of some size $\rho^{(t)}$ in an approximate natural gradient direction via

$$\widetilde{\eta}_\phi \leftarrow (1-\rho^{(t)})\widetilde{\eta}_\phi + \rho^{(t)}\left(\eta_\phi + K\mathbb{E}_{q(z^{(k)})}[t(z^{(k)}, y^{(k)})]\right) \qquad (5.19)$$

where we have assumed the minibatches are of equal size to simplify notation. The local variational factor $q(z^{(k)})$ is computed using a local

Algorithm 17 Stochastic Variational Inference (SVI)

```
Initialize global variational parameter $\widetilde{\eta}_\phi^{(0)}$
**for** $t = 0, 1, 2, \ldots$ until convergence **do**
    $\hat{k} \leftarrow$ sample index $k$ with probability $p_k > 0$, for $k = 1, 2, \ldots, K$
    $q(z^{(\hat{k})}) \leftarrow \textsc{LocalMeanField}(\widetilde{\eta}^{(t)}, y^{(\hat{k})})$
    $\widetilde{\eta}_\phi^{(t+1)} \leftarrow (1 - \rho^{(t)})\widetilde{\eta}_\phi^{(t)} + \rho^{(t)} \left(\eta_\phi + \frac{1}{p_{\hat{k}}}\mathbb{E}_{q(z^{(\hat{k})})}\left[t(z^{(\hat{k})}, y^{(\hat{k})})\right]\right)$
```

mean field update on the data minibatch and the global variational factor. That is, if $q(z^{(k)})$ is not further factorized in the mean field approximation, it is computed according to

$$q(z^{(k)}) \propto \exp\left\{\mathbb{E}_{q(\phi)}[\log p(z^{(k)}, y^{(k)} \,|\, \phi)]\right\}. \tag{5.20}$$

We summarize the general SVI algorithm in Algorithm 17.

To ensure asymptotic convergence to a stationary point of the objective, the step size sequence $(\rho^{(t)})_{t=0}^{\infty}$ is typically chosen to be decreasing as in Section 2.5. However, Mandt et al. [2016] analyzes the algorithm with a fixed step size and compares its stationary distribution to the SGLD inference algorithm discussed in Section 3.4.

5.1.2 Stochastic gradients with general nonconjugate models

The development of SVI in the preceding section assumes that $p(\phi)$ and $p(z, y \,|\, \phi)$ are a conjugate pair of exponential families. This assumption led to a particularly convenient form for the natural gradient of the mean field variational objective and hence an efficient stochastic gradient ascent algorithm. However, when models do not have this conjugacy structure, more general algorithms are required.

In this section we review Black Box Variational Inference (BBVI), which is a stochastic gradient algorithm for variational inference that can be applied at scale [Ranganath et al., 2014]. The "black box" name suggests its generality: while the stochastic variational inference of Section 5.1.1 requires particular model structure, BBVI only requires that the model's log joint distribution can be evaluated. It also makes few demands of the variational family, since it only requires that the family can be sampled and that the gradient of its log joint density with respect to the variational parameters can be computed efficiently. With these minimal requirements, BBVI is not only useful in the big-data setting but also a tool for handling nonconjugate variational inference more generally. Because BBVI uses Monte Carlo approximation to compute stochastic gradient updates, it fits naturally into a stochastic gradient optimization framework, and hence it has the additional benefit of yielding a scalable algorithm simply by adding minibatch sampling to its updates at the cost of increasing their variance. In this subsection we review the general BBVI algorithm and then compare it to the SVI algorithm of Section 5.1.1. For a review of Monte Carlo estimation, see Section 2.2.2.

We consider a general model $p(\theta, y) = p(\theta) \prod_{k=1}^{K} p(y^{(k)} \,|\, \theta)$ including parameters θ and observations $y = \{y^{(k)}\}_{k=1}^{K}$ divided into K minibatches. The distribution of interest is the posterior $p(\theta \,|\, y)$ and we write the variational family as $q(\theta) = q(\theta \,|\, \widetilde{\eta}_\theta)$, where we suppress the particular mean field factorization structure of $q(\theta)$ from the notation. The mean field variational lower bound is then

$$\mathcal{L} = \mathbb{E}_{q(\theta)} \left[\log \frac{p(\theta, y)}{q(\theta)} \right]. \tag{5.21}$$

Taking the gradient with respect to the variational parameter $\widetilde{\eta}_\theta$ and expanding the expectation into an integral, we have

$$\begin{aligned} \nabla_{\widetilde{\eta}_\theta} \mathcal{L} &= \nabla_{\widetilde{\eta}_\theta} \int q(\theta) \log \frac{p(\theta, y)}{q(\theta)} d\theta & (5.22) \\ &= \int \nabla_{\widetilde{\eta}_\theta} \left[\log \frac{p(\theta, y)}{q(\theta)} \right] q(\theta) d\theta + \int \log \frac{p(\theta, y)}{q(\theta)} \nabla_{\widetilde{\eta}_\theta} q(\theta) d\theta & (5.23) \end{aligned}$$

where we have moved the gradient into the integrand and applied the product rule to yield two terms. The first term is identically zero:

$$\begin{aligned} \int \nabla_{\widetilde{\eta}_\theta} \left[\log \frac{p(\theta, y)}{q(\theta)} \right] q(\theta) d\theta &= - \int \frac{1}{q(\theta)} \nabla_{\widetilde{\eta}_\theta} \left[q(\theta) \right] q(\theta) d\theta & (5.24) \\ &= - \int \nabla_{\widetilde{\eta}_\theta} q(\theta) d\theta & (5.25) \\ &= - \nabla_{\widetilde{\eta}_\theta} \int q(\theta) d\theta = 0 & (5.26) \end{aligned}$$

where we have used $\nabla_{\widetilde{\eta_\theta}} \log p(\theta, y) = 0$. To write the second term of (5.23) in a form that allows convenient Monte Carlo approximation, we first note the identity

$$\nabla_{\widetilde{\eta_\theta}} \log q(\theta) = \frac{\nabla_{\widetilde{\eta_\theta}} q(\theta)}{q(\theta)} \implies \nabla_{\widetilde{\eta_\theta}} q(\theta) = q(\theta) \nabla_{\widetilde{\eta_\theta}} \log q(\theta) \tag{5.27}$$

and hence we can write the second term of (5.23) as

$$\int \log \frac{p(\theta, y)}{q(\theta)} \nabla_{\widetilde{\eta_\theta}} q(\theta) d\theta = \int \log \frac{p(\theta, y)}{q(\theta)} \nabla_{\widetilde{\eta_\theta}} \left[\log q(\theta)\right] q(\theta) d\theta \tag{5.28}$$

$$= \mathbb{E}_{q(\theta)} \left[\log \frac{p(\theta, y)}{q(\theta)} \nabla_{\widetilde{\eta_\theta}} \log q(\theta) \right] \tag{5.29}$$

$$\approx \frac{1}{|\mathcal{S}|} \sum_{\hat{\theta} \in S} \log \frac{p(\hat{\theta}, y)}{q(\hat{\theta})} \nabla_{\widetilde{\eta_\theta}} \log q(\hat{\theta}) \tag{5.30}$$

where in the final line we have written the expectation as a Monte Carlo estimate using a set of samples $\mathcal{S}$, where $\hat{\theta} \overset{\text{iid}}{\sim} q(\theta)$ for $\hat{\theta} \in \mathcal{S}$. Notice that the gradient is written as a weighted sum of gradients of the variational log density with respect to the variational parameters, where the weights depend on the model log joint density.

The BBVI algorithm uses the Monte Carlo estimate (5.30) to compute stochastic gradient updates. This gradient estimator is also known as the score function estimator [Kleijnen and Rubinstein, 1996, Gelman and Meng, 1998] and REINFORCE gradient [Williams, 1992]. The variance of these updates, and hence the convergence of the overall stochastic gradient algorithm, depends both on the sizes of the gradients of the variational log density and on the variance of $q(\theta)$. Large variance in the gradient estimates can lead to very slow optimization, and so Ranganath et al. [2014] proposes and evaluates two variance reduction schemes, including a control variate method as well as a Rao-Blackwellization method that can exploit factorization structure in the variational family. The control variate method for variance reduction is also studied in Paisley et al. [2012].

To provide a scalable version of BBVI, gradients can be further approximated by subsampling minibatches of data. That is, using $\log p(\theta, y) = \log p(\theta) + \sum_{k=1}^{K} \log p(y^{(k)} \mid \theta)$ we write (5.29) and

Algorithm 18 Minibatch Black-Box Variational Inference (BBVI)

```
Initialize $\widetilde{\eta}_\theta^{(0)}$
for $t = 0, 1, 2, \ldots$ do
    $\mathcal{S} \leftarrow \{\hat{\theta}_s\}$ where $\hat{\theta}_s \sim q(\,\cdot\,|\,\widetilde{\eta}_\theta^{(t)})$
    $\hat{k} \sim \text{Uniform}(\{1, 2, \ldots, K\})$
    $\widetilde{\eta}_\theta^{(t+1)} \leftarrow \widetilde{\eta}_\theta^{(t)} + \frac{1}{|\mathcal{S}|} \sum_{\hat{\theta} \in \mathcal{S}} \left( \log \frac{p(\hat{\theta})}{q(\hat{\theta})} + K \log p(y^{(\hat{k})} \,|\, \hat{\theta}) \right) \nabla_{\widetilde{\eta}_\theta} \log q(\hat{\theta})$
```

(5.30) as

$$\nabla_{\widetilde{\eta}_\theta} \mathcal{L} = \mathbb{E}_{\hat{k}} \left[\mathbb{E}_{q(\theta)} \left[\left(\log \frac{p(\theta)}{q(\theta)} + K \log p(y^{(\hat{k})} \,|\, \theta) \right) \nabla_{\widetilde{\eta}_\theta} \log q(\theta) \right] \right] \tag{5.31}$$

$$\approx \frac{1}{|\mathcal{S}|} \sum_{\hat{\theta} \in \mathcal{S}} \left(\log \frac{p(\hat{\theta})}{q(\hat{\theta})} + K \mathbb{E}_{\hat{k}} \left[\log p(y^{(\hat{k})} \,|\, \hat{\theta}) \right] \right) \nabla_{\widetilde{\eta}_\theta} \log q(\hat{\theta}) \tag{5.32}$$

with the minibatch index $\hat{k}$ distributed uniformly over $\{1, 2, \ldots, K\}$ and the minibatches are assumed to be the same size for simpler notation. This subsampling over minibatches further increases the variance of the updates and thus may further limit the rate of convergence of the algorithm. We summarize this version of the BBVI algorithm in Algorithm 18.

It is instructive to compare the fully general BBVI algorithm applied to hierarchical models to the SVI algorithm of Section 5.1.1; this comparison not only shows the benefits of exploiting conjugacy structure but also suggests a potential Rao-Blackwellization scheme. Taking $\theta = (\phi, z)$ and $q(\theta) = q(\phi)q(z)$ and starting from (5.22) and (5.29), we can write the gradient as

$$\nabla_{\widetilde{\eta}_\theta} \mathcal{L} = \mathbb{E}_{q(\phi)q(z)} \left[\log \frac{p(\phi, z, y)}{q(\phi)q(z)} \nabla_{\widetilde{\eta}_\phi} \log q(\phi)q(z) \right] \tag{5.33}$$

$$\approx \frac{1}{|S|} \sum_{\hat{\phi} \in S} \left(\mathbb{E}_{q(z)} \log \frac{p(\hat{\phi}, z, y)}{q(\hat{\phi})} - \mathbb{E}_{q(z)} \log q(z) \right) \nabla_{\widetilde{\eta}_\phi} \log q(\hat{\phi}) \tag{5.34}$$

where S is a set of samples with $\hat{\phi} \overset{\text{iid}}{\sim} q(\phi)$ for $\hat{\phi} \in S$. Thus if the entropy of the local variational distribution $q(z)$ and the expectations with respect to $q(z)$ of the log density $\log p(\hat{\phi}, z, y)$ can be computed without resorting to Monte Carlo estimation, then the resulting update would likely have a lower variance than the BBVI update that requires sampling over both $q(\phi)$ and $q(z)$.

This comparison also makes clear the advantages of exploiting conjugacy in SVI: when the updates of Section 5.1.1 can be used, neither $q(\phi)$ nor $q(z)$ needs to be sampled. Furthermore, while BBVI uses stochastic gradients in its updates, the SVI algorithm of Section 5.1.1 uses stochastic natural gradients, adapting to the local curvature of the variational family. Computing stochastic natural gradients in BBVI would require both computing the Fisher information matrix of the variational family and solving a linear system with it.

The main weakness of the score function (REINFORCE) gradient estimates used in BBVI is their high variance, a weakness that scales poorly with dimensionality. Next we study an alternative gradient estimator that applies to some nonconjugate models but can yield lower variance estimates.

5.1.3 Exploiting reparameterization for some nonconjugate models

While the score function estimator of BBVI in Section 5.1.2 is sufficiently general to handle essentially any model, some nonconjugate models admit convenient stochastic gradient estimators that can have lower variance. In particular, when the latent variables are continuous (or any discrete latent variables that can be marginalized efficiently), samples from some variational distributions can be reparameterized in a way that enables an alternative stochastic gradient estimator. This technique was described in Williams [1992, Section 7.2] and is related to non-centered reparameterizations [Papaspiliopoulos et al., 2007]. In the context of variational inference it is referred to as the reparameterization trick [Kingma and Welling, 2014, Rezende et al., 2014].

Consider again the variational density $q(\theta)$ with parameter $\widetilde{\eta}_\theta$. The reparameterization trick applies when the random variable $\theta \sim q(\theta)$ can be written as a deterministic function of the parameter $\widetilde{\eta}_\theta$ and another random variable ϵ with a distribution that does not depend on $\widetilde{\eta}_\theta$,

$$\theta = f(\widetilde{\eta}_\theta, \epsilon), \tag{5.35}$$

where $\epsilon \sim p(\epsilon)$ and the partial derivative $\nabla_{\widetilde{\eta}_\theta} f(\widetilde{\eta}_\theta, \epsilon)$ exists and can be computed efficiently for almost every value of ϵ. Using this reparameterization, we can write the variational objective as

$$\mathcal{L}(\widetilde{\eta}_\theta) = \mathbb{E}_{q(\theta)}\left[\log \frac{p(\theta, y)}{q(\theta)}\right] = \mathbb{E}_{p(\epsilon)}\left[\log \frac{p(f(\widetilde{\eta}_\theta, \epsilon), y)}{q(f(\widetilde{\eta}_\theta, \epsilon))}\right]. \tag{5.36}$$

That is, the expectation does not depend on $\widetilde{\eta}_\theta$, and so when we can exchange differentiation and expectation[3] we have

$$\nabla \mathcal{L}(\widetilde{\eta}_\theta) = \mathbb{E}_{p(\epsilon)}\left[\nabla_{\widetilde{\eta}_\theta} \log \frac{p(f(\widetilde{\eta}_\theta, \epsilon), y)}{q(f(\widetilde{\eta}_\theta, \epsilon))}\right] \tag{5.38}$$

$$\approx \frac{1}{|S|} \sum_{\hat{\epsilon} \in S} \nabla_{\widetilde{\eta}_\theta} \log \frac{p(f(\widetilde{\eta}_\theta, \hat{\epsilon}), y)}{q(f(\widetilde{\eta}_\theta, \hat{\epsilon}))}, \tag{5.39}$$

where $\hat{\epsilon} \overset{\text{iid}}{\sim} p(\epsilon)$ for each $\hat{\epsilon} \in S$. This Monte Carlo approximation gives an unbiased estimate of the gradient of the variational objective, and it often has lower variance than the more general score function estimator [Kingma and Welling, 2014]. Moreover, it can be straightforward to compute using automatic differentiation tools [Kucukelbir et al., 2015, Duvenaud and Adams, 2015]. There are several proposed improvements to this estimator based on variance reduction and alternative Monte Carlo objectives [Mnih and Gregor, 2014, Burda et al., 2016, Mnih and Rezende, 2016].

As a concrete example, if $q(\theta)$ is a multivariate Gaussian density with parameters $\widetilde{\eta}_\theta = (\mu, \Sigma)$, then we can take $\epsilon \sim \mathcal{N}(0, I)$ and write the reparameterization as

$$f(\mu, \Sigma, \epsilon) = \mu + \Sigma^{\frac{1}{2}} \epsilon, \tag{5.40}$$

where $\Sigma^{\frac{1}{2}}$ is a matrix satisfying $\Sigma^{\frac{1}{2}}(\Sigma^{\frac{1}{2}})^\mathsf{T} = \Sigma$.

[3] For example, Leibniz's rule states that given a function $f : X \times \Omega \to \mathbb{R}$ where $X \subset \mathbb{R}$ is open, if $f(x, \omega)$ is a Lebesgue-integrable function of ω for each $x \in X$, the partial derivative function $\frac{\partial}{\partial x} f(x, \omega)$ exists for almost all ω, and there is an integrable function $G : \Omega \to \mathbb{R}$ such that $|\frac{\partial}{\partial x} f(x, \omega)| \leq G(\omega)$, then

$$\frac{d}{dx} \int_\Omega f(x, \omega) d\omega = \int_\Omega \frac{\partial}{\partial x} f(x, \omega) d\omega. \tag{5.37}$$

5.2 Streaming variational Bayes (SVB)

Streaming variational Bayes (SVB) provides an alternative framework in which to derive minibatch-based scalable variational inference [Broderick et al., 2013]. While the methods of Section 5.1 generally apply stochastic gradient optimization algorithms to a fixed variational mean field objective, SVB instead considers the streaming data setting, in which case there may be no fixed dataset size and hence no fixed variational objective. To handle streaming data, the SVB approach is based on the classical idea of Bayesian updating, in which a posterior is updated to reflect new data as they become available. This sequence of posteriors is approximated by a sequence of variational models, and each variational model is computed from the previous variational model via an incremental update on new data. Tank et al. [2015] develops a related streaming variational inference algorithm that applies to Bayesian nonparametric models using ideas from assumed density filtering [Minka, 2001]. Another streaming data approach, which we do not discuss here, is studied in McInerney et al. [2015].

More concretely, given a prior $p(\theta)$ over a parameter θ and a (possibly infinite) sequence of data minibatches $y^{(1)}, y^{(2)}, \ldots$, each distributed independently according to a likelihood distribution $p(y^{(k)} \mid \theta)$, we consider the sequence of posteriors

$$p(\theta \mid y^{(1)}, \ldots, y^{(t)}), \quad t = 1, 2, \ldots. \tag{5.41}$$

Given an approximation updating algorithm $\mathcal{A}$ one can compute a corresponding sequence of approximations

$$p(\theta \mid y^{(1)}, \cdots, y^{(t)}) \approx q_t(\theta) = \mathcal{A}\left(y^{(t)}, q_{t-1}(\theta)\right), \quad t = 1, 2, \ldots \tag{5.42}$$

with $q_0(\theta)p(\theta)$. This sequential updating view naturally suggests an online or one-pass algorithm in which the update (5.42) is applied successively to each of a sequence of minibatches.

A sequence of such updates may also exploit parallel or distributed computing resources. For example, the sequence of approximations may be computed as

$$p(\theta \,|\, y^{(1)}, \cdots, y^{(K_t)}) \approx q_t(\theta) \tag{5.43}$$

$$= q_{t-1}(\theta) \left(\prod_{k=K_{t-1}+1}^{K_t} \mathcal{A}\left(y^{(k)}, q_{t-1}(\theta)\right) q_{t-1}(\theta)^{-1} \right) \tag{5.44}$$

where $K_{t-1}+1, K_{t-1}+2, \ldots, K_t$ indexes a set of data minibatches for which each update is computed in parallel before being combined in the final update from $q_{t-1}(\theta)$ to $q_t(\theta)$.

This combination of partial results is especially appealing when the prior $p(\theta)$ and the family of approximating distributions $q(\theta)$ are in the same exponential family,

$$p(\theta) \propto \exp\left\{\langle \eta,\ t(\theta)\rangle\right\} \quad q_0(\theta) \propto \exp\left\{\langle \widetilde{\eta}_0,\ t(\theta)\rangle\right\} \tag{5.45}$$

$$q_t(\theta) = \mathcal{A}(y^{(k)}, q_{t-1}(\theta)) \propto \exp\left\{\langle \widetilde{\eta}_t,\ t(\theta)\rangle\right\} \tag{5.46}$$

for a prior natural parameter η and a sequence of variational parameters $\widetilde{\eta}_t$. In the exponential family case, the updates (5.44) can be written

$$p(\theta \,|\, y^{(1)}, \cdots, y^{(K_t)}) \approx q_t(\theta) \propto \exp\left\{\langle \widetilde{\eta}_t,\ t(\theta)\rangle\right\} \tag{5.47}$$

$$= \exp\left\{\langle \widetilde{\eta}_{t-1} + \sum_k (\widetilde{\eta}_k - \widetilde{\eta}_{t-1}),\ t(\theta)\rangle\right\} \tag{5.48}$$

where we may take the algorithm $\mathcal{A}$ to return an updated natural parameter, $\widetilde{\eta}_k = \mathcal{A}(y^{(k)}, \widetilde{\eta}_t)$.

Finally, similar updates can be performed in an asynchronous distributed master-worker setting. Each worker can process a minibatch and send the corresponding natural parameter increment to a master process, which updates the global variational parameter and transmits back the updated variational parameter along with a new data minibatch. In symbols, we can write that a worker operating on minibatch $y^{(k)}$ for some minibatch index k computes the update increment $\Delta\widetilde{\eta}_k$ according to

$$\Delta\widetilde{\eta}_k = \mathcal{A}(y^{(k)}, \widetilde{\eta}_{\tau(k)}) \tag{5.49}$$

Algorithm 19 Streaming Variational Bayes (SVB)

```
Initialize $\widetilde{\eta}_0$
for each worker $p = 1, 2, \ldots, P$ do
    Send task $(y^{(p)}, \widetilde{\eta}_0)$ to worker $p$
as workers send updates do
    Receive update $\Delta\widetilde{\eta}_k$ from worker $p$
    $\widetilde{\eta}_{t+1} \leftarrow \widetilde{\eta}_t + \Delta\widetilde{\eta}_k$
    Retrieve new data minibatch index $k'$
    Send new task $(y^{(k')}, \widetilde{\eta}_{t+1})$ to worker $p$
```

Algorithm 20 SVB Worker Process

```
repeat
    Receive task $(y^{(k)}, \widetilde{\eta}_t)$ from master
    $\Delta\widetilde{\eta}_k \leftarrow \mathcal{A}(y^{(k)}, \widetilde{\eta}_t) - \widetilde{\eta}_t$
    Send update $\Delta\widetilde{\eta}_k$ to master
until no tasks remain
```

where $\tau(k)$ is the index of the global variational parameter used in the worker's computation. Upon receiving an update, the master updates its global variational parameter synchronously according to

$$\widetilde{\eta}_{t+1} = \widetilde{\eta}_t + \Delta\widetilde{\eta}_k. \tag{5.50}$$

We summarize a version of this process in Algorithms 19 and 20.

A related algorithm, which we do not detail here, is Memoized Variational Inference (MVI) [Hughes and Sudderth, 2013, Hughes et al., 2015]. While this algorithm is designed for the fixed dataset setting rather than the streaming setting, the updates can be similar to those of SVB. In particular, MVI optimizes the mean field objective in the conjugate exponential family setting using the mean field coordinate descent algorithm but with an atypical update order, in which only some local variational factors are updated at a time. This update order enables minibatch-based updating but, unlike the stochastic gradient algorithms, does not optimize out the other local variational factors not included in the minibatch and instead leaves them fixed.

5.3 Scalable expectation propagation

Expectation propagation (EP) is another variational inference strategy distinct from mean field methods. As discussed in Section 2.4, the EP algorithm is a Lagrangian method and is not guaranteed to make progress on its objective or any other merit function, and as a result its stability is more difficult to analyze than that of mean field methods. However, EP can be very effective in many problems and applications [Minka, 2001, Murphy, 2012], especially in the important domain of approximate inference in Gaussian Processes (GPs), which motivates investigating new versions of EP that are particularly scalable to large datasets.

Here we summarize two recent approaches to developing scalable EP algorithms: Parallel EP (PEP) and Stochastic EP (SEP). Parallel EP [Gelman et al., 2014b, Xu et al., 2014, Hasenclever et al., 2016] distributes factors across the dataset and runs standard EP updates in parallel, making the basic strategy analogous to the parallel MCMC algorithms of Chapter 4. As a result, PEP may be especially relevant for complex models with many interactions, such as hierarchical models. Stochastic EP [Li et al., 2015] instead updates a global approximation using stochastic updates on minibatches, and is hence more analogous to the minibatch-based algorithms of Chapter 3 and Sections 5.1 and 5.2. Hence SEP provides an EP-based alternative to the stochastic-update mean field algorithms like SVI and BBVI.

5.3.1 Parallel expectation propagation (PEP)

Gelman et al. [2014b] and Xu et al. [2014] observe that EP provides a natural way to incorporate parallel computation, and in particular that for hierarchical models the cavity distribution idea allows parallel workers to efficiently communicate their inferences and inform each other. Here we develop the resulting parallel EP algorithm following Gelman et al. [2014b, Section 3] using the hierarchical model notation developed in Section 5.1.1 and Figure 5.1. Because these EP methods can use MCMC for local inference, these methods are strongly related to the parallel MCMC algorithms developed in Chapter 4.

For some dataset $y = \{y_k\}_{k=1}^K$ partitioned into K subsets, consider the hierarchical joint model

$$p(\phi, z, y) = p(\phi) \prod_{k=1}^{K} p(y_k, z_k \,|\, \phi), \tag{5.51}$$

with the target posterior distribution defined as either $p(\phi, z \,|\, y)$ or $p(\phi \,|\, y)$. Consider a corresponding approximating distribution on the global parameter, $q(\phi)$, written as

$$q(\phi) \propto p(\phi) \prod_{k=1}^{K} q_k(\phi). \tag{5.52}$$

As in Section 2.4, we can define the cavity distribution $q_{\neg k}(\phi)$ as

$$q_{\neg k}(\phi) \propto \frac{q(\phi)}{q_k(\phi)}. \tag{5.53}$$

Using the hierarchical model structure, we define the *local* tilted distribution $\widetilde{p}_k(\phi, z_k)$ on both the global parameter ϕ and the local latent variable z_k as

$$\widetilde{p}_k(\phi, z_k) \propto q_{\neg k}(\phi) p(y_k, z_k \,|\, \phi), \tag{5.54}$$

so that the tilted distribution on the global parameter, denoted $\widetilde{p}_k(\phi)$, is simply the appropriate marginal of Eq. (5.54). Note that the cavity distribution $q_{\neg k}(\phi)$ acts as a kind of local prior, summarizing the influence of the model prior $p(\phi)$ as well as the influence of the other data. The moments of $\widetilde{p}_k(\phi)$ can then be estimated with an appropriate inference method, such as MCMC or a local EP algorithm, and we can update the parameters of the factor $q_k(\phi)$ so that the resulting $q(\phi) \propto q_k(\phi) q_{\neg k}(\phi)$ matches the estimated moments.

A serial EP algorithm would run these updates for each factor $q_k(\phi)$ sequentially and, after each update, the global approximation $q(\phi)$ would be updated as $q(\phi) \propto q_k(\phi) q_{\neg k}(\phi)$. However, Gelman et al. [2014b] instead suggest running these updates in parallel, setting the global factor to $q(\phi) \propto p(\phi) \prod_k q_k(\phi)$ after each round of updates. Because the bulk of the computational effort is in estimating the moments of $\widetilde{p}_k(\phi)$ and setting the parameters of $q_k(\phi)$ so as to match those moments, this strategy parallelizes the most expensive part of inference

Algorithm 21 Parallel Expectation Propagation (PEP)

```
Input: Joint distribution $p(\phi, z, y) = p(\phi) \prod_{k=1}^{K} p(y_k, z_k \mid \phi)$, parameterized approximating family $q(\phi) = \prod_{k=1}^{K} q_k(\phi)$
Output: Approximate marginal posterior $q(\phi) \approx p(\phi \mid y)$
  Initialize parameters of each approximating factor $q_k(\phi)$
  for $t = 1, 2, \ldots$ until convergence do
      for $k = 1, 2, \ldots, K$ in parallel do
          Define cavity distribution $q_{\neg k} \propto \frac{q(\phi)}{q_k(\phi)}$
          Define local tilted distribution $\widetilde{p}_k(\phi, z_k) \propto q_{\neg k}(\phi) p(y_k, z_k \mid \phi)$
          Set parameters of $q_k(\phi)$ to minimize $\mathrm{KL}(\widetilde{p}_k(\phi) \,\|\, q(\phi))$
              by computing and matching marginal moments
      Synchronize the updated approximation $q(\phi) \propto p(\phi) \prod_{k=1}^{K} q_k(\phi)$
```

while only requiring the parameters of the $q_k(\phi)$ to be communicated. At convergence, the posteriors of interest are approximated as

$$p(\phi \mid y) \approx q(\phi) = p(\phi) \prod_{k=1}^{K} q_k(\phi) \tag{5.55}$$

$$p(\phi, z \mid y) \approx q(\phi) \prod_{k=1}^{K} p(z_k \mid \phi) p(y_k \mid z_k, \phi). \tag{5.56}$$

The algorithm is summarized in Algorithm 21.

Note that mean field algorithms offer similar opportunities for parallelization; if the mean field family is factorized as $q(\phi) \prod_k q(z_k)$ then updates to the local factors $q(z_k)$ can be performed in parallel for a fixed global factor $q(\phi)$. However, as with comparing EP with mean field methods in the sequential setting, in some cases EP can provide more accurate posterior approximations, and so it is useful to develop a corresponding parallelization strategy for EP. Gelman et al. [2014b] also study several algorithmic considerations that arise, including stability, approximate inference subroutine alternatives, and ways to reuse some parts of the computation.

While Algorithm 21 uses a global synchronization step for communication, the EP framework allows more flexible communication strategies based on message passing. Xu et al. [2014] also develop a parallel EP algorithm, called sampling via moment sharing (SMS), that uses MCMC for local inference, and the authors highlight that the processors can even communicate their local moment statistics asynchronously.

Hasenclever et al. [2016] further develop these ideas, citing that while PEP and SMS work well for simpler models like logistic regression and hierarchical linear models, they found these strategies not to be as effective on complex models like Bayesian deep neural networks. They instead propose stochastic natural-gradient EP (SNEP), which leverages exponential family structure and a locally-convergent version of EP [Heskes and Zoeter, 2002] as well as both asynchronous communication and a posterior server.

5.3.2 Stochastic expectation propagation (SEP)

While the parallel EP algorithm of the preceding section uses the structure of EP to construct a parallel algorithm, Stochastic Expectation Propagation (SEP) [Li et al., 2015] instead builds an EP algorithm with updates that can be computed using only one minibatch of data at a time. Thus SEP is closer to an EP analog of the other minibatch-based variational inference algorithms described in Sections 5.1.1 and 5.2, and has similar advantages. Specifically, parallel EP still requires the parameters of the K approximating factors to be stored in (distributed) memory, and also requires global communication to synchronize the approximating distribution $q(\phi)$. In contrast, SEP needs only an amount of memory that is constant with respect to the size of the dataset and only performs local computations. However, it also makes stronger approximating assumptions which might make it less appropriate for complex hierarchical models.

For some dataset $y = \{y_k\}_{k=1}^K$ partitioned into K minibatches, consider the joint model

$$p(\theta, y) = p(\theta) \prod_{k=1}^{K} p(y_k \,|\, \theta). \tag{5.57}$$

In standard EP, the posterior would be approximated with a factorized distribution as

$$q(\theta) \propto p(\theta) \prod_{k=1}^{K} q_k(\theta). \tag{5.58}$$

At convergence, the product of the $q_k(\theta)$ factors is meant to approximate the effect of the likelihood terms, so that $\prod_k q_k(\theta) \approx \prod_k p(y_k \,|\, \theta)$. The key idea in SEP is that we can instead directly parameterize a factor $f(\theta)$ that models the (geometric) average effect of the likelihood terms, writing

$$q(\theta) \propto p(\theta) f(\theta)^K, \tag{5.59}$$

so that $f(\theta)^K \approx \prod_k q_k(\theta)$. We can then compute stochastic updates to the factor $f(\theta)$ directly using data minibatches.

The SEP update proceeds on the tth iteration by sampling a minibatch index k, then defining a cavity distribution $q_{\neg k}(\theta) \propto p(\theta) f(\theta)^{1-\frac{1}{K}}$ and tilted distribution $\widetilde{p}_k(\theta) \propto p(y_k \,|\, \theta) q_{\neg k}(\theta)$. Next, an intermediate factor $f_k(\theta)$ parameterized like $f(\theta)$ is chosen to approximately minimize the KL divergence from the tilted distribution $\widetilde{p}_k(\theta)$ to the distribution proportional to $q_{\neg k}(\theta) f_k(\theta)$. As with the other EP algorithms, this step is carried out using an inference subroutine to compute moments of the tilted distribution and then setting the parameters of $f_k(\theta)$ to approximately match those moments. Finally, the average likelihood factor $f(\theta)$ is updated according to $f(\theta) \propto f(\theta)^{1-\epsilon_t} f_k(\theta)^{\epsilon_t}$ for some step size ϵ_t, with one natural choice being $\epsilon_t = \frac{1}{K}$. The resulting algorithm is summarized in Algorithm 22.

5.4 Summary

Stochastic gradients vs. streaming. The methods of Section 5.1 apply stochastic optimization to variational mean field inference objectives. In optimization literature and practice, stochastic gradient methods have a large body of both theoretical and empirical support, and so these methods offer a compelling framework for scalable inference. The streaming ideas surveyed in Section 5.2 are less well understood, but by treating the streaming setting, rather than the setting of a large fixed-size dataset, they may be more relevant for some applications.

Algorithm 22 Stochastic Expectation Propagation (SEP)

```
Input: Joint distribution $p(\theta, y) = p(\theta)\prod_{k=1}^{K} p(y_k \mid \theta)$, parameterized
  average likelihood factor $f(\theta)$, step size sequence $(\epsilon_t)_{t=1}^{\infty}$
Output: Approximate posterior $q(\theta) = p(\theta)f(\theta)^K$
  Initialize parameters of the average likelihood factor $f(\theta)$
  for $t = 1, 2, \ldots$ until convergence do
    Choose an observation index $k$ from $\{1, 2, \ldots, K\}$
    Define cavity distribution $q_{\neg k}(\theta) \propto p(\theta)f(\theta)^{1-\frac{1}{K}}$
    Define tilted distribution $\widetilde{p}_k(\theta) \propto p(y_k \mid \theta)q_{\neg k}(\theta)$
    Set parameters of $f_k(\theta)$ to minimize $\mathrm{KL}(\widetilde{p}_k(\theta) \,\|\, q_{\neg k}(\theta)f_k(\theta))$
        by computing and matching moments
    Update $f(\theta) \propto f(\theta)^{1-\epsilon_t} f_k(\theta)^{\epsilon_t}$
```

Minibatching vs data parallelism. All of this chapter's scalable approaches to mean field variational inference, as well as Stochastic Expectation Propagation's approach to EP inference, are based on processing minibatches of data, analogous to the MCMC methods of Chapter 3. Minibatches are motivated in three ways: through conditionally-i.i.d. models and stochastic gradient optimization (§5.1), through streaming data (§5.2), and through the factor updates of EP (§5.3.2). In both SVI and BBVI, stochastic gradients arise from randomly sampling data minibatches, while in BBVI there is additional stochasticity due to the Monte Carlo approximation required to handle nonconjugate structure. In contrast to SVI and BBVI, SVB (§5.2) processes data minibatches to drive incremental posterior updates, constructing a sequence of approximate posterior distributions that correspond to classical sequential Bayesian updating without having a single fixed objective to optimize. SEP also directly defines an algorithm with minibatch updates. The only method in this chapter that exploits data parallelism and distributed computation instead of minibatch data access is Parallel EP (§5.3.1), making it more analogous to the MCMC methods of Chapter 4.

Generality, requirements, and assumptions. As with most approaches to scaling MCMC samplers for Bayesian inference, the minibatch-based variational inference methods depend on conditionally-i.i.d. model structure. In SVI, BBVI, and SEP minibatches map to terms in a factorization of the joint probability. In SVB, minibatches map to a sequence of likelihoods to be incorporated into the variational posterior. Some of these methods further depend on and exploit exponential family and conjugacy structure. SVI is based on complete-data conjugacy, while BBVI was specifically developed for nonconjugate models. SVB is a general framework, but in the conjugate exponential family case the updates can be written in terms of simple updates to natural parameters. A direction for future research for mean field methods might be to develop new methods based on identifying and exploiting some 'middle ground' between the structural requirements of SVI and BBVI, or similarly of SVB with and without exponential family structure.

5.5 Discussion

Model complexity and data redundancy. As with many other scalable inference methods, the variational inference strategies discussed in this chapter are likely to work very well in the "tall data" regime, where the data are modeled as conditionally independent given relatively simple global parameters. However, it is less clear how these methods can be expected to work on complex models. In particular, the minibatch-based algorithms rely to some extent on data redundancy, so that a sampled minibatch can be used to make global inferences. However, the optimization perspective provided by variational inference gives us a clearer understanding of these minibatch effects than in the case of MCMC. Because variational inference algorithms generally only need to find local optima, and we don't require algorithm trajectories to satisfy the stationary distribution conditions of MCMC, the subsampling errors from minibatches can only lead to slower convergence, at least in principle. In contrast, the minibatch-based MCMC algorithms of Chapter 3 introduced new posterior approximation errors. To handle complex models with less data redundancy, the data-parallel versions of EP may provide more flexibility: instead of requiring small minibatches to be conditionally independent, these EP methods may only require the data subsets on different machines to be conditionally independent, and those data subsets may be much larger than standard minibatches.

Parallel variants of minibatch updating. The minibatch-based variational inference methods developed in this chapter suggest parallel and asynchronous variants. In the case of SVB, distributed and asynchronous versions, such as the master-worker pattern depicted by Algorithms 19 and 20, have been empirically studied [Broderick et al., 2013]. However, we lack theoretical understanding about these procedures, and it is unclear how to define and track notions of convergence or stability. Methods based on stochastic gradients, such as SVI, can naturally be extended to exploit parallel and asynchronous (or "Hogwild") variants of stochastic gradient ascent. In such parallel settings, these optimization-based techniques benefit from powerful gradient convergence results [Bertsekas and Tsitsiklis, 1989, Section 7.8], though tuning such algorithms is still a challenge. Other parallel versions of these ideas and algorithms have also been developed in Campbell and How [2014] and Campbell et al. [2015].

Inference networks and bridging the gap to MCMC. Recently, new strategies for variational inference have provided both computational efficiency and novel variational families. In stochastic variational inference, instead of optimizing the local variational factor $q(z^{(k)})$ in Algorithm 17, the parameters of a (suboptimal) local factor can be computed from the data minibatch $y^{(k)}$ using a simpler, non-iterative computation, like the application of a feed-forward neural network. This strategy of using inference networks [Rezende et al., 2014, Kingma and Welling, 2014], also referred to as amortized inference, avoids the iterative local optimization which can be expensive for general nonconjugate models and has led to many new ideas based on the variational autoencoder [Kingma and Welling, 2014]. Another new idea is to parameterize variational families in terms of a fixed number of MCMC transition steps [Salimans et al., 2015], "unrolling" the computation and computing gradients through it to optimize the approximation. These new ideas, which exploit the convenience of automatic differentiation and the expressive function approximators provided by deep neural networks, are likely to further influence scalable variational inference.

Understanding and correcting the biases in variational inference Variational methods produce biased estimates of posterior expectations when the variational family does not include the true posterior. However, the ways in which these biases affect posterior expectations of interest are often unclear. Because mean field variational inference tends to underestimate uncertainty, Giordano et al. [2015] develop the linear response variational Bayes (LRVB) method to improve posterior uncertainty estimates using a sensitivity analysis of the mean field optimization problem based on the implicit function theorem. Further research is needed into how the biases found in variational methods can be analyzed, corrected, or compared to those of approximate MCMC algorithms.

6

Challenges and questions

In this review, we have examined a variety of different views on scaling Bayesian inference up to large datasets and greater model complexity and out to parallel compute resources. Several different themes have emerged, from techniques that exploit subsets of data for computational savings to proposals for distributing inference computations across multiple machines. Progress is being made, but there remain significant open questions and outstanding challenges to be tackled as this research programme moves forward.

Trading off errors in MCMC One of the key insights underpinning much of the recent work on scaling Bayesian inference can be framed in terms of a kind of bias-variance tradeoff. Traditional MCMC theory provides asymptotically unbiased estimators for which the error can eventually be driven arbitrarily small. However, in practice, under limited computational budgets the error can be significant. This error has two components: transient bias, in which the samples produced are too dependent on the Markov chain's initialization, and Monte Carlo standard error, in which the samples collected may be too few or too highly correlated to produce good estimates.

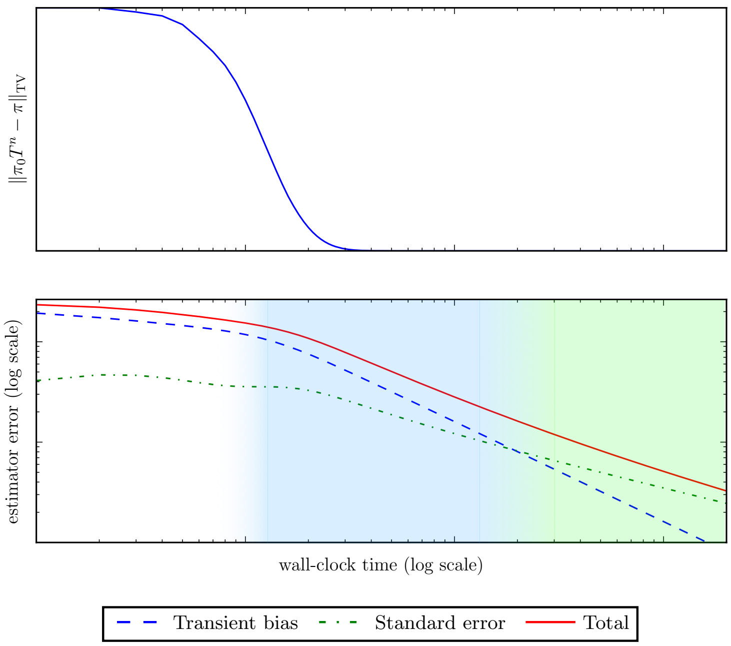


Figure 6.1: A simulation illustrating the error terms in traditional MCMC estimators as a function of wall-clock time (log scale). The marginal distributions of the Markov chain iterates converge to the target distribution (top panel), while the errors in MCMC estimates due to transient bias and Monte Carlo standard error are driven arbitrarily small.

Figure 6.1 illustrates the error regimes and tradeoffs in traditional MCMC.[1] Asymptotic analysis describes the regime on the right of the plot, after the sampler has mixed sufficiently well. In this regime, the marginal distribution of each sample is essentially equal to the target distribution, and the transient bias from initialization, which affects only the early samples in the Monte Carlo sum, is washed out rapidly at least at a $\mathcal{O}(\frac{1}{n})$ rate. The dominant source of error is due to Monte Carlo standard error, which diminishes only at a $\mathcal{O}(\frac{1}{\sqrt{n}})$ rate.

However, machine learning practitioners using MCMC often find themselves in another regime: in the middle of the plot, the error is decreasing but dominated instead by the transient bias. The challenge

[1] See also Section 2.2.4

in practice is often to get through this regime, or even to get into it at all. When the underlying Markov chain does not mix sufficiently well or when the transitions cannot be computed sufficiently quickly, getting to this regime may be infeasible for a realistic computational budget.

Several of the new MCMC techniques we have studied aim to address this challenge. In particular, the parallel predictive prefetching method of Section 4.1.2 accelerates this phase of MCMC without affecting the stationary distribution. Other methods instead introduce approximate transition operators that can be executed more efficiently. For example, the adaptive subsampling methods of Section 3.2 and the stochastic gradient sampler of Section 3.4 can execute updates more efficiently by operating only on data subsets, while the Weierstrass and Hogwild Gibbs samplers of Sections 4.2.1 and 4.2.2, respectively, execute more quickly by leveraging parallelism. These transition operators are approximate in that they do not admit the exact target distribution as a stationary distribution: instead, the stationary distribution (when it exists and is unique) is only intended to be close to the target. Framed in terms of Monte Carlo estimates, these approximations effectively accelerate the execution of the chain at the cost of introducing an asymptotic bias. Figure 6.2 illustrates this new tradeoff.

Allowing some asymptotic bias to reduce transient bias or even Monte Carlo variance is likely to enable MCMC inference at a new scale. However, both the amount of asymptotic bias introduced by these methods and the ways in which it depends on model and algorithm parameters remain unclear. More theoretical understanding and empirical study is necessary to guide machine learning practice.

Scaling limits of Bayesian inference Scalability in the context of Bayesian inference is ultimately about spending computational resources to better interrogate posterior distributions. It is therefore important to consider whether there are fundamental limits to what can be achieved by, for example, spending more money on Amazon EC2, for either faster computers or more of them.

In parallel systems, linear scaling is ideal: twice as much computational power yields twice as much useful work. Unfortunately, even

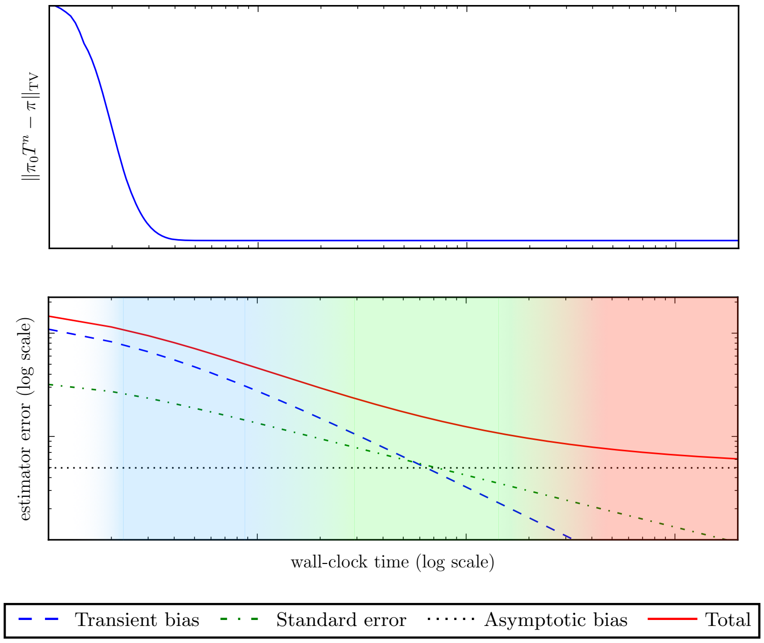


Figure 6.2: A simulation illustrating the new tradeoffs in some proposed scalable MCMC methods. Compare to Figure 6.1. As a function of wall-clock time (log scale), the Markov chain iterations execute more than an order of magnitude faster; however, because the stationary distribution is not the target distribution, an asymptotic bias remains (top panel). Correspondingly, MCMC estimator error, particularly the transient bias, can be driven to a small value more rapidly, but there is an error floor due to the introduction of the asymptotic bias (bottom panel).

if this lofty parallel speedup goal is achieved, the asymptotic picture for MCMC is dim: in the asymptotic regime, doubling the number of samples collected can only reduce the Monte Carlo standard error by a factor of $\sqrt{2}$. This scaling means that there are diminishing returns to purchasing additional computational resources, even if those resources provide linear speedup in terms of accelerating the execution of the MCMC algorithm.

Interestingly, variational methods may not suffer from such intrinsic limits. Because expectations are computed against the variational distribution, rather than against collected samples, the rate at which such approximate expectations can improve is not limited by Monte Carlo effects.

Measuring performance With all the ideas surveyed here, one thing is clear: there are many alternatives for how to scale Bayesian inference. How should we compare these alternative algorithms? Can we tell when any of these algorithms work well in an absolute sense?

One standard approach for evaluating MCMC procedures is to define a set of scalar-valued test functions (or estimands of interest) and compute effective sample size [Gelman et al., 2014a, Section 11.5][Kong et al., 1994] as a function of wall-clock time. However, in complex models designing an appropriately comprehensive set of test functions may be difficult. Furthermore, many such measures require the Markov chain to mix and do not account for any asymptotic bias [Gorham and Mackey, 2015], hence limiting their applicability to measuring the performance of many of the new inference methods studied here.

To confront these challenges, one recently-proposed approach [Gorham and Mackey, 2015] draws on Stein's method, classically used as an analytical tool, to design an efficiently-computable measure of discrepancy between a target distribution and a set of samples. A natural measure of discrepancy between a target density $p(x)$ and a (weighted) sample distribution $q(x)$, where $q(x) = \sum_{i=1}^{n} w_i \delta_{x_i}(x)$ for some set of samples $\{x_i\}_{i=1}^{n}$ and weights $\{w_i\}_{i=1}^{n}$, is to consider their largest absolute difference across a large class of test functions:

$$d_{\mathcal{H}}(q, p) = \sup_{h \in \mathcal{H}} |\mathbb{E}_q h(X) - \mathbb{E}_p h(X)| \tag{6.1}$$

where $\mathcal{H}$ is the class of test functions. While expectations with respect to the target density p may be difficult to compute, by designing $\mathcal{H}$ such that $\mathbb{E}_p h(X) = 0$ for every $h \in \mathcal{H}$, we need only compute expectations with respect to the sample distribution q. To meet this requirement, instead of designing $\mathcal{H}$ directly, we can instead choose $\mathcal{H}$ to be the image of another function class $\mathcal{G}$ under an operator $\mathcal{T}_p$ that may depend on p, so that $\mathcal{H} = \mathcal{T}_p \mathcal{G}$ and the requirement becomes $\mathbb{E}_p(\mathcal{T}_p g)(x) = 0$ and the discrepancy measure becomes

$$d_{\mathcal{T}_p\mathcal{G}}(q,p) = \sup_{g\in\mathcal{G}} |\mathbb{E}_q(\mathcal{T}_p g)(X)|. \tag{6.2}$$

Such operators $\mathcal{T}_p$ can be designed using infinitessimal generators from continuous-time ergodic Markov processes, and Gorham and Mackey [2015] suggest using the operator

$$(\mathcal{T}_p g)(x) \triangleq \langle g(x), \nabla \log p(x)\rangle + \langle \nabla, \nabla g(x)\rangle \tag{6.3}$$

which requires computing only the gradient of the target log density. Furthermore, while the optimization in (6.2) is infinite-dimensional in general and might have infinitely many smoothness constraints from $\mathcal{G}$, Gorham and Mackey [2015] shows that for the sample distribution q the test function g need only be evaluated at the finitely-many sample points $\{x_i\}_{i=1}^n$ and that only a small number of constraints must be enforced. This new performance metric does not require assumptions on whether the samples are generated from an unbiased, stationary Markov chain, and so it may provide clear ways to compare across a broad spectrum sampling-based approximate inference algorithms.

Another recently-proposed approach attempts to estimate or bound the KL divergence from an algorithm's approximate posterior representation to the true posterior, at least when applied to synthetic data. This approach, called bidirectional Monte Carlo (BDMC) [Grosse et al., 2015], can be applied to measure the performance of both variational mean field algorithms as well as annealed importance sampling (AIS) and sequential Monte Carlo (SMC) algorithms. By rearranging the variational identity (2.50), we can write the KL divergence $\mathrm{KL}(q\|p)$ from an approximating distribution $q(z,\theta)$ to a target posterior $p(z,\theta\,|\,\bar{y})$ in terms of the log marginal likelihood $\log p(\bar{y})$ and an expectation with respect to $q(z,\theta)$:

$$\mathrm{KL}(q\|p) = \log p(\bar{y}) - \mathbb{E}_{q(z,\theta)}\left[\log \frac{p(z,\theta\,|\,\bar{y})}{q(z,\theta)}\right]. \tag{6.4}$$

Because the expectation can be readily computed in a mean field setting or stochastically lower-bounded when using AIS [Grosse et al., 2015, Section 4.1], with a stochastic upper bound on $\log p(\bar{y})$ we can use (6.4) to compute a stochastic upper bound on the KL divergence $\mathrm{KL}(q\|p)$. BDMC provides a method to compute such stochastic upper bounds on $\log p(\bar{y})$ for synthetic datasets $\bar{y}$, and so may enable new performance metrics that apply to both sampling-based algorithms as well as variational mean field algorithms. However, while MCMC transition operators are used to construct AIS algorithms, BDMC does not directly apply to evaluating the performance of such transition operators in standard MCMC inference.

Developing performance metrics and evaluation procedures is critical to making progress. As observed in Grosse et al. [2015],

> In many application areas of machine learning, especially supervised learning, benchmark datasets have spurred rapid progress in developing new algorithms and clever refinements to existing algorithms. [. . .] So far, the lack of quantitative performance evaluations in marginal likelihood estimation, and in sampling-based inference more generally, has left us fumbling around in the dark.

By developing better ways to measure the performance of these Bayesian inference algorithms, we will be much better equipped to compare, improve, and extend them.

Acknowledgements

This work was funded in part by NSF IIS-1421780 and the Alfred P. Sloan Foundation. E.A. is supported by the Miller Institute for Basic Research in Science, University of California, Berkeley. M.J. is supported by a fellowship from the Harvard/MIT Joint Grants program.

References

Sungjin Ahn, Anoop Korattikara Balan, and Max Welling. Bayesian posterior sampling via stochastic gradient Fisher scoring. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Talal M. Alkhamis, Mohamed A. Ahmed, and Vu Kim Tuan. Simulated annealing for discrete optimization with estimation. *European Journal of Operational Research*, 116(3):530–544, 1999.

Shun-ichi Amari and Hiroshi Nagaoka. *Methods of Information Geometry*. American Mathematical Society, 2007.

Christophe Andrieu and Eric Moulines. On the ergodicity properties of some adaptive MCMC algorithms. *The Annals of Applied Probability*, 16(3): 1462–1505, 2006.

Christophe Andrieu and Gareth O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *Annals of Statistics*, pages 697–725, 2009.

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

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society Series B*, 72(3):269–342, 2010.

Elaine Angelino. *Accelerating Markov chain Monte Carlo via parallel predictive prefetching*. PhD thesis, School of Engineering and Applied Sciences, Harvard University, 2014.

Elaine Angelino, Eddie Kohler, Amos Waterland, Margo Seltzer, and Ryan P. Adams. Accelerating MCMC via parallel predictive prefetching. In *30th Conference on Uncertainty in Artificial Intelligence*, pages 22–31, 2014.

Kenneth J. Arrow, Leonid Hurwicz, Hirofumi Uzawa, H.B. Chenery, S.M. Johnson, S. Karlin, T. Marschak, and R.M. Solow. *Studies in linear and non-linear programming.* Stanford University Press, John Wiley & Sons, 1959.

Arthur U. Asuncion, Padhraic Smyth, and Max Welling. Asynchronous distributed learning of topic models. In *Advances in Neural Information Processing Systems 21*, pages 81–88, 2008.

Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

Rémi Bardenet and Odalric-Ambrym Maillard. Concentration inequalities for sampling without replacement. *Bernoulli*, 20(3):1361–1385, 2015.

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*, 2014.

Rémi Bardenet, Arnaud Doucet, and Chris Holmes. On Markov chain Monte Carlo methods for tall data. *arXiv preprint 1505.02827*, 2015.

Mark A. Beaumont. Estimation of population growth or decline in genetically monitored populations. *Genetics*, 164(3):1139–60, 2003.

Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 2016.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

Michael Betancourt. The fundamental incompatibility of scalable Hamiltonian Monte Carlo and naive data subsampling. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 533–540, 2015.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

J. Frederic Bonnans and Alexander Shapiro. *Perturbation Analysis of Optimization Problems.* Springer Science & Business Media, 2000.

Léon Bottou. On-line learning and stochastic approximations. In David Saad, editor, *On-line Learning in Neural Networks*, pages 9–42. Cambridge University Press, New York, NY, USA, 1998.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, January 2011.

A. E. Brockwell. Parallel Markov chain Monte Carlo simulation by prefetching. *Journal of Computational and Graphical Statistics*, 15(1):246–261, March 2006.

Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C. Wilson, and Michael I. Jordan. Streaming variational Bayes. In *Advances in Neural Information Processing Systems 26*, pages 1727–1735, 2013.

Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC press, 2011.

Akif Asil Bulgak and Jerry L. Sanders. Integrating a modified simulated annealing algorithm with the simulation of a manufacturing system to optimize buffer sizes in automatic assembly systems. In *Proceedings of the 20th Conference on Winter Simulation*, pages 684–690, New York, NY, USA, 1988. ACM.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *International Conference on Learning Representations*, 2016.

Jonathan M. R. Byrd, Stephen A. Jarvis, and Abhir H. Bhalerao. Reducing the run-time of MCMC programs by multithreading on SMP architectures. In *IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8, 2008.

Jonathan M. R. Byrd, Stephen A. Jarvis, and Abhir H. Bhalerao. On the parallelisation of MCMC by speculative chain execution. In *IEEE International Symposium on Parallel and Distributed Processing - Workshop Proceedings*, pages 1–8, 2010.

Trevor Campbell and Jonathan P. How. Approximate decentralized Bayesian inference. In *30th Conference on Uncertainty in Artificial Intelligence*, pages 102–111, 2014.

Trevor Campbell, Julian Straub, John W. Fisher III, and Jonathan P. How. Streaming, distributed variational inference for Bayesian nonparametrics. In *Advances in Neural Information Processing Systems 28*, pages 280–288, 2015.

Tianqi Chen, Emily B. Fox, and Carlos Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning*, June 2014.

John M. Danskin. *The Theory of Max-Min and its Application to Weapons Allocation Problems*. Springer-Verlag, New York, 1967.

Chris De Sa, Kunle Olukotun, and Christopher Ré. Ensuring rapid mixing and low bias for asynchronous Gibbs sampling. In *Proceedings of the 33rd International Conference on Machine Learning*, June 2016.

J. E. Dennis, Jr. and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall Series in Computational Mathematics, 1983.

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 27*, pages 3203–3211, 2014.

Finale Doshi-Velez, David A. Knowles, Shakir Mohamed, and Zoubin Ghahramani. Large scale nonparametric Bayesian inference: Data parallelisation in the Indian buffet process. In *Advances in Neural Information Processing Systems 22*, pages 1294–1302, 2009.

Arnaud Doucet, Michael Pitt, Robert Kohn, and George Deligiannidis. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika*, 102(2):295–313, 2015.

David Duvenaud and Ryan P. Adams. Black-box stochastic variational inference in five lines of python. *NIPS Workshop on Black-box Learning and Inference*, 2015.

Paul Fearnhead, Omiros Papaspiliopoulos, Gareth O. Roberts, and Andrew Stuart. Random-weight particle filtering of continuous time processes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72 (4):497–512, 2010.

Anthony V. Fiacco. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. Academic Press, Inc., 1984.

Andrew Gelman and Xiao-Li Meng. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical science*, pages 163–185, 1998.

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 3rd edition, 2014a.

Andrew Gelman, Aki Vehtari, Pasi Jylänki, Christian Robert, Nicolas Chopin, and John P Cunningham. Expectation propagation as a way of life. *arXiv preprint arXiv:1412.4869*, 2014b.

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 721–741, 1984.

Charles J. Geyer. Practical Markov chain Monte Carlo. *Statistical Science*, pages 473–483, 1992.

Ryan J. Giordano, Tamara Broderick, and Michael I. Jordan. Linear response methods for accurate covariance estimates from mean field variational Bayes. In *Advances in Neural Information Processing Systems 28*, pages 1441–1449, 2015.

Mark Girolami and Ben Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) (With Discussion)*, 73:123 – 214, 03 2011.

Joseph Gonzalez, Yucheng Low, Arthur Gretton, and Carlos Guestrin. Parallel Gibbs sampling: From colored fields to thin junction trees. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 324–332, 2011.

Jackson Gorham and Lester Mackey. Measuring sample quality with Stein's method. In *Advances in Neural Information Processing Systems 28*, pages 226–234, 2015.

Thore Graepel, Joaquin Quñonero Candela, Thomas Borchert, and Ralf Herbrich. Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. In *Proceedings of the 27th International Conference on Machine Learning*, pages 13–20, 2010.

Roger B. Grosse, Zoubin Ghahramani, and Ryan P. Adams. Sandwiching the marginal likelihood using bidirectional Monte Carlo. *arXiv preprint arXiv:1511.02543*, 2015.

Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242, 04 2001.

Leonard Hasenclever, Stefan Webb, Thibaut Lienart, Sebastian Vollmer, Balaji Lakshminarayanan, Charles Blundell, and Yee Whye Teh. Distributed Bayesian learning with stochastic natural-gradient expectation propagation and the posterior server. *arXiv preprint arXiv:1512.09327*, 2016.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

Tom Heskes and Onno Zoeter. Expectation propagation for approximate inference in dynamic Bayesian networks. In *18th Conference on Uncertainty in Artificial Intelligence*, pages 216–223, 2002.

Christian Hipp. Sufficient statistics and exponential families. *The Annals of Statistics*, 2(6):1283–1292, 1974.

Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B. Gibbons, Gregory R. Ganger, Garth Gibson, and Eric P. Xing. More effective distributed ML via a stale synchronous parallel parameter server. In *Advances in Neural Information Processing Systems 26*, pages 1223–1231, 2013.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1): 1303–1347, May 2013.

Zaiying Huang and Andrew Gelman. Sampling for Bayesian computation with large datasets. Technical report, Columbia University, 2005.

Michael C. Hughes and Erik B. Sudderth. Memoized online variational inference for Dirichlet process mixture models. In *Advances in Neural Information Processing Systems 26*, pages 1133–1141, 2013.

Michael C. Hughes, Dae Il Kim, and Erik B. Sudderth. Reliable and scalable variational inference for the hierarchical Dirichlet process. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, pages 370–378, 2015.

Alexander Ihler and David Newman. Understanding errors in approximate distributed latent Dirichlet allocation. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):952–960, 2012.

Pierre E. Jacob and Alexandre H. Thiery. On nonnegative unbiased estimators. *Annals of Statistics*, 43(2):769–784, 04 2015.

Matthew J. Johnson, James Saunderson, and Alan S. Willsky. Analyzing Hogwild parallel Gaussian Gibbs sampling. In *Advances in Neural Information Processing Systems 26*, pages 2715–2723, 2013.

Matthew James Johnson. *Bayesian Time Series Models and Scalable Inference*. PhD thesis, Massachusetts Institute of Technology, 2014.

Robert W. Keener. *Theoretical Statistics: Topics for a Core Course*. Springer Texts in Statistics. Springer New York, 2010.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. *International Conference on Learning Representations*, 2014.

Jack P. C. Kleijnen and Reuven Y. Rubinstein. Optimization and sensitivity analysis of computer simulation models by the score function method. *European Journal of Operational Research*, 88(3):413–427, 1996.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Augustine Kong, Jun S. Liu, and Wing Hung Wong. Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, 1994.

Anoop Korattikara, Yutian Chen, and Max Welling. Austerity in MCMC land: Cutting the Metropolis-Hastings budget. In *Proceedings of the 31th International Conference on Machine Learning*, volume 32, pages 181–189, 2014.

Alp Kucukelbir, Rajesh Ranganath, Andrew Gelman, and David Blei. Automatic variational inference in stan. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 568–576. Curran Associates, Inc., 2015.

Neil D. Lawrence. Modelling in the context of massively missing data, 2015. URL `http://staffwww.dcs.shef.ac.uk/people/N.Lawrence/talks/missingdata_tuebingen15.pdf`.

Benedict Leimkuhler and Xiaocheng Shang. Adaptive thermostats for noisy gradient systems. *SIAM Journal on Scientific Computing*, 38(2):A712–A736, 2016.

Yingzhen Li, José Miguel Hernández-Lobato, and Richard E. Turner. Stochastic expectation propagation. In *Advances in Neural Information Processing Systems 28*, pages 2323–2331, 2015.

L. Lin, K. F. Liu, and J. Sloan. A noisy Monte Carlo algorithm. *Physical Review D*, 61:074505, March 2000.

Zhiyuan Liu, Yuzhou Zhang, Edward Y. Chang, and Maosong Sun. PLDA+: Parallel latent Dirichlet allocation with data placement and pipeline processing. *ACM Transactions on Intelligent Systems and Technology*, 2(3): 26:1–26:18, May 2011.

Anne-Marie Lyne, Mark Girolami, Yves Atchaé, Heiko Strathmann, and Daniel Simpson. On Russian roulette estimates for Bayesian inference with doubly-intractable likelihoods. *Statistical Science*, 30(4):443–467, 11 2015.

Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems 28*, pages 2917–2925, 2015.

David J. C. MacKay. *Information Theory, Inference & Learning Algorithms.* Cambridge University Press, New York, NY, USA, 2003.

Dougal Maclaurin and Ryan P. Adams. Firefly Monte Carlo: Exact MCMC with subsets of data. In *30th Conference on Uncertainty in Artificial Intelligence*, pages 543–552, 2014.

Stephan Mandt, Matthew D. Hoffman, and David M. Blei. A variational analysis of stochastic gradient algorithms. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2015.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 3. Academic Press, 1982.

James McInerney, Rajesh Ranganath, and David M. Blei. The population posterior and bayesian inference on streams. In *Advances in Neural Information Processing Systems 28*, pages 1153–1161, 2015.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

Sean Meyn and Richard L. Tweedie. *Markov Chains and Stochastic Stability.* Cambridge University Press, New York, NY, USA, 2nd edition, 2009.

Thomas P. Minka. Expectation propagation for approximate Bayesian inference. In *17th Conference on Uncertainty in Artificial Intelligence*, pages 362–369, 2001.

Stanislav Minsker, Sanvesh Srivastava, Lizhen Lin, and David B. Dunson. Scalable and robust Bayesian inference via the median posterior. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1656–1664, 2014.

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1791–1799, 2014.

Andriy Mnih and Danilo Jimenez Rezende. Variational inference for monte carlo objectives. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Volodymyr Mnih, Csaba Szepesvári, and Jean-Yves Audibert. Empirical Bernstein stopping. In *Proceedings of the 25th International Conference on Machine Learning*, pages 672–679, 2008.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *15th Conference on Uncertainty in Artificial Intelligence*, pages 467–475, 1999.

Radford M. Neal. An improved acceptance procedure for the hybrid Monte Carlo algorithm. *J. Comput. Phys.*, 111(1):194–203, March 1994.

Radford M. Neal. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, Chapman & Hall/CRC Handbooks of Modern Statistical Methods, pages 113–162. CRC Press, 2010.

Willie Neiswanger, Chong Wang, and Eric Xing. Asymptotically exact, embarrassingly parallel MCMC. In *30th Conference on Uncertainty in Artificial Intelligence*, pages 623–632, 2014.

David Newman, Padhraic Smyth, Max Welling, and Arthur U. Asuncion. Distributed inference for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems 20*, pages 1081–1088, 2007.

David Newman, Arthur U. Asuncion, Padhraic Smyth, and Max Welling. Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10:1801–1828, 2009.

Robert Nishihara, Iain Murray, and Ryan P. Adams. Parallel MCMC with generalized elliptical slice sampling. *Journal of Machine Learning Research*, 15:2087–2112, 2014.

Manfred Opper and Ole Winther. A Bayesian approach to on-line learning. In David Saad, editor, *On-line Learning in Neural Networks*, pages 363–378. Cambridge University Press, New York, NY, USA, 1998.

John Paisley, David M. Blei, and Michael I. Jordan. Variational Bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Omiros Papaspiliopoulos. A methodological framework for Monte Carlo probabilistic inference for diffusion processes. Technical report, Centre for Research in Statistical Methodology, University of Warwick, June 2009.

Omiros Papaspiliopoulos, Gareth O. Roberts, and Martin Sköld. A general framework for the parametrization of hierarchical models. *Statistical Science*, pages 59–73, 2007.

Sam Patterson and Yee Whye Teh. Stochastic gradient Riemannian Langevin dynamics on the probability simplex. In *Advances in Neural Information Processing Systems 26*, pages 3102–3110, 2013.

M. F. Pradier, P. G. Moreno, F. J. R. Ruiz, I. Valera, H. Mollina-Bulla, and F. Perez-Cruz. Map/reduce uncollapsed Gibbs sampling for Bayesian non parametric models. *Workshop in Software Engineering for Machine Learning at NIPS*, 2014.

Maxim Rabinovich, Elaine Angelino, and Michael I. Jordan. Variational Consensus Monte Carlo. In *Advances in Neural Information Processing Systems 28*, pages 1207–1215, 2015.

Rajesh Ranganath, Chong Wang, David M. Blei, and Eric P. Xing. An adaptive learning rate for stochastic variational inference. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 298–306, 2013.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black box variational inference. In *17th International Conference on Artificial Intelligence and Statistics*, pages 814–822, 2014.

Benjamin Recht, Christopher Ré, Stephen J. Wright, and Feng Niu. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems 24*, pages 693–701, 2011.

Jeffrey Regier, Andrew Miller, Jon McAuliffe, Ryan P. Adams, Matt Hoffman, Dustin Lang, David Schlegel, and Prabhat. Celeste: Variational inference for a generative model of astronomical images. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2095–2103, 2015.

Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1278–1286, 2014.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics).* Springer-Verlag New York, Inc., 2004.

Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4): 341–363, 12 1996.

Gareth O. Roberts, Andrew Gelman, and Walter R. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms. *Annals of Applied Probability*, 7:110–120, 1997.

Tim Salimans, Diederik P. Kingma, and Max Welling. Markov chain Monte Carlo and variational inference: Bridging the gap. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1218–1226, 2015.

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

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.

R. J. Serfling. Probability inequalities for the sum in sampling without replacement. *The Annals of Statistics*, 2(1):39–48, 1974.

Xiaocheng Shang, Zhanxing Zhu, Benedict Leimkuhler, and Amos J. Storkey. Covariance-controlled adaptive Langevin thermostat for large-scale Bayesian sampling. In *Advances in Neural Information Processing Systems 28*, pages 37–45, 2015.

Sameer Singh, Michael L. Wick, and Andrew McCallum. Monte Carlo MCMC: Efficient inference by approximate sampling. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1104–1113, 2012.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2951–2959, 2012.

David H. Stern, Ralf Herbrich, and Thore Graepel. Matchbox: Large scale online Bayesian recommendations. In *Proceedings of the 18th International Conference on World Wide Web*, pages 111–120, 2009.

Ingvar Strid. Efficient parallelisation of Metropolis-Hastings algorithms using a prefetching approach. *Computational Statistics & Data Analysis*, 54(11): 2814–2835, November 2010.

Alex Tank, Nicholas Foti, and Emily Fox. Streaming variational inference for Bayesian nonparametric mixture models. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, pages 968–976, 2015.

Yee Whye Teh, Alexandre H. Thiery, and Sebastian J. Vollmer. Consistency and fluctuations for stochastic gradient Langevin dynamics. *The Journal of Machine Learning Research*, 17(1):193–225, 2016.

Wolfgang Wagner. Unbiased Monte Carlo evaluation of certain functional integrals. *Journal of Computational Physics*, 71(1):21–33, 1987.

Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, November 2008.

Ling Wang and Liang Zhang. Stochastic optimization using simulated annealing with hypothesis test. *Applied Mathematics and Computation*, 174 (2):1329–1342, 2006.

Xiangyu Wang and David B. Dunson. Parallel MCMC via Weierstrass sampler. *arXiv preprint arXiv:1312.4605*, 2013.

Karl Weierstrass. Über die analytische darstellbarkeit sogenannter willkrlicher functionen einer reellen vernderlichen. *Sitzungsberichte der Königlich Preuischen Akademie der Wissenschaften zu Berlin*, 1885. (II). Erste Mitteilung (part 1) pp. 633–639, Zweite Mitteilung (part 2) pp. 789–805.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

E. E. Witte, R. D. Chamberlain, and M. A. Franklin. Parallel simulated annealing using speculative computation. *IEEE Transactions on Parallel and Distributed Systems*, 2(4):483–494, 1991.

Minjie Xu, Balaji Lakshminarayanan, Yee Whye Teh, Jun Zhu, and Bo Zhang. Distributed Bayesian posterior sampling via moment sharing. In *Advances in Neural Information Processing Systems 27*, pages 3356–3364, 2014.