

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800

Open access books available

122,000

International authors and editors

135M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.

For more information visit www.intechopen.com



A Brief Tour of Bayesian Sampling Methods

Michelle Y. Wang and Trevor Park

Abstract

Unlike in the past, the modern Bayesian analyst has many options for approximating intractable posterior distributions. This chapter briefly summarizes the class of posterior sampling methods known as Markov chain Monte Carlo, a type of dependent sampling strategy. Varieties of algorithms exist for constructing chains, and we review some of them here. Such methods are quite flexible and are now used routinely, even for relatively complicated statistical models. In addition, extensions of the algorithms have been developed for various goals. General-purpose software is currently also available to automate the construction of samplers, freeing the analyst to focus on model formulation and inference.

Keywords: Markov chain Monte Carlo, Gibbs sampler, slice sampler, Metropolis-Hastings, Hamiltonian Monte Carlo, cluster sampling, JAGS, Stan

1. Introduction

Modern Bayesian data analysis is enabled by specialized computational tools. Except in relatively simple models, explicit solutions for quantities relevant to Bayesian inference are not available. This limitation has sparked the development of many different approximation methods.

Some approximation methods, such as Laplace approximation [1] and variational Bayes [2], are based on replacing the Bayesian posterior density with a computationally convenient approximation. Such methods may have the advantage of relatively quick computation and scalability, but they leave open the question of how much the resulting approximate Bayesian inference can be trusted to reflect the actual Bayesian inference. There is an inherent bias in the approximation that generally cannot be reduced by applying more intensive computation.

When accuracy is important, simulation-based (stochastic) methods offer an attractive alternative. The goal of these methods is to produce a simulation sample (though not necessarily an independent one) from the (joint) posterior distribution. A simulation sample can be used to approximate almost any quantity relevant to Bayesian inference, including posterior expectations, variances, quantiles, and marginal densities. Since the approximations become more exact as more samples are used, accuracy tends to be limited only by the computational resources available.

Random variates from a general probability distribution that has a known density may be simulated using many classical methods, such as accept/reject and importance sampling. However, such methods tend to be efficient only in special cases and often require analytical insight to improve efficiency. The past three

decades have seen interest dramatically increase in the category of Markov chain Monte Carlo (MCMC) methods. Unlike most classical methods, MCMC can often be efficiently automated, even for moderately complicated models. A variety of MCMC methods are available, giving the analyst flexibility in implementation. Moreover, general software is now available that automates most computational details, allowing the analyst to focus on model formulation and inference.

The purpose of this chapter is to offer an introduction to Bayesian simulation methods, with emphasis on MCMC. The motivation and popularity of posterior sampling are illustrated in Section 2. Section 3 describes MCMC and the associated issues including convergence monitoring, mixing, and thinning. Varieties of specific sampling methods are provided in Sections 4 and 5, with the general-purpose software implementing them described in Section 6.

2. Posterior sampling

Bayesian inference requires access to the posterior distribution. Let y denote all of the data to be modeled, and suppose its sampling distribution is in a parametric family with density $\pi(y|\theta)$, where θ represents the parameter (usually a vector), including any hyperparameters. If the prior on θ has density $\pi(\theta)$ then, according to Bayes' rule, the posterior distribution has density

$$\pi(\theta|y) = \frac{\pi(y|\theta)\pi(\theta)}{\pi(y)} \propto_{\theta} \pi(y|\theta)\pi(\theta) \quad (1)$$

where the proportionality is in θ (not y). (An improper prior can be used, provided the posterior is proper.) The normalizing constant $\pi(y)$ is notoriously difficult to compute, so methods that avoid using it are preferred.

Since $\pi(y|\theta)$ and $\pi(\theta)$ are typically specified by the analyst, the (unnormalized) posterior density is readily available and typically easy to evaluate. Nonetheless, most quantities used in Bayesian inference (posterior expected values, quantiles, marginal densities, etc.) are defined by integrals involving the posterior density, which are usually intractable and are difficult to deterministically approximate when θ has more than a few components.

This explains the popularity of posterior sampling. Given a sample from the posterior of sufficient effective size, posterior expected values can be approximated by sample means, posterior quantiles by sample quantiles, posterior marginal densities by sample-based density estimates, and so forth. Most posterior inference is readily accomplished if an efficient method of sampling from the posterior is available.

Independent sampling from the posterior is seemingly ideal, since relatively few samples are required to obtain a good approximation in most cases, and the approximation error is relatively easy to characterize. Unfortunately, methods for independent sampling have proven difficult to implement in a general way that efficiently scales with the dimension of θ . For example, rejection sampling (accept/reject) is efficient only if the posterior is tightly bounded by a known function proportional to a density that is easy to sample. Finding such a function is generally difficult, and even adaptive variants struggle in high-dimensional situations.

Currently, the most efficient generally adaptable methods use *dependent* sampling. Dependent sampling usually incurs a computational cost of acquiring a larger number of samples to attain a given accuracy, but the flexibility of these methods and their scalability to higher dimensions offset this disadvantage.

3. Sampling with Markov chains

MCMC is a type of dependent sampling in which the samples are obtained from successive states of a discrete-time Markov chain [3]. The Markov chain is designed to be easy to simulate and intended to (eventually) produce samples that have a distribution arbitrarily close to the posterior distribution.

Specifically, the Markov chain is designed to have a particular *stationary distribution*: a distribution on the state space of the chain that is preserved by the transition kernel. If the chain is started in the stationary distribution, all successive states will have the stationary distribution. In the most basic case, the state space will be the range of θ , and the stationary distribution will be the posterior distribution. A collection of successive states can then be regarded as a (dependent) sample from the posterior.

Since starting the chain in the stationary distribution is difficult, MCMC relies on the stationary distribution also being the (unique) *limiting distribution*: the distribution to which the states converge (in law) as the time index increases. Conditions under which the chain converges are technical (e.g., [4]) and can be difficult to verify analytically in complicated models. Thus, though convergence properties may benefit from following some general guidelines in specifying the MCMC method, convergence is usually checked empirically.

General convergence monitoring tools and techniques are available to determine by what time point convergence has been practically achieved, so that accurate samples can be collected thereafter. See [5] for an overview. Some tools rely on simulating the chain several times, independently, from different starting points.

Running the chain(s) until declaring convergence is called *burn-in*, or sometimes *warm up*. All values generated during burn-in are discarded, except for the final state, which becomes the starting point for sampling.

The degree of dependence within a Markov chain determines the number of samples needed for a given level of approximation. Most MCMC methods produce chains with positive dependence, requiring a larger number of samples to be taken than if independent sampling were used. Chains that are highly dependent exhibit slow *mixing*: the decay rate of dependence between the states of the chain at two time points as the time lag increases. In extreme cases, slow mixing makes MCMC computationally prohibitive, since an enormous number of samples may be needed to achieve a reasonable approximation. Methods with fast mixing are typically preferred.

When sampling is highly dependent, using only a regularly spaced subsample of the generated values may be almost as accurate as using all of the values. Retaining only the regularly spaced subsample is called *thinning*. Although it does not reduce the amount of computation required, it can dramatically reduce the time and space required for storage of the values.

Characterizing Monte Carlo error in approximations from an MCMC sample is more difficult than from an independent sample. However, effective methods are available for most cases. See [6].

4. Constructing Markov chains for sampling

This section briefly summarizes the most practical and frequently used methods for forming a Markov Chain appropriate for sampling from a posterior distribution. All of them need only a function proportional to the posterior density of θ , as in Eq. (1). For brevity, we denote it as

$$f(\theta) \propto_{\theta} \pi(y|\theta) \pi(\theta) \quad (2)$$

where the proportionality is in θ only, and the dependence on y has been suppressed in the notation.

4.1 Gibbs sampling

Consider a partition of θ into p pieces (which may themselves be vectors):

$$\theta = (\theta_1, \dots, \theta_K) \quad (3)$$

The *full conditional* (or *conditional posterior*) distribution of θ_k is its posterior distribution conditional on all the other pieces θ_{-k} , i.e., the distribution with density

$$\pi(\theta_k | \theta_{-k}, y) \quad (4)$$

Gibbs sampling, in its purest form, is sequential sampling from the full conditional distributions of θ_k , $k = 1, \dots, K$, each time conditioning upon the most recently sampled value for each component of θ_{-k} . Each complete cycle of this process produces a single sampled value of θ , and these successive values form a Markov chain whose stationary distribution (if unique) is the posterior distribution (since each step in the cycle preserves the posterior distribution of θ).

Essentially, Gibbs sampling reduces the problem of sampling θ to the problem of conditionally sampling each of its pieces. It relies on each full conditional being easy to sample. Because the pieces are of lower dimension (perhaps even one-dimensional), they may be easier to sample by conventional methods. Moreover, it is often possible to choose a prior distribution such that many of the full conditionals are easy to sample. For example, when conditional priors are chosen from easily sampled families that are *partially conjugate* to the sampling model (see, e.g., [7]), the Gibbs sampler is easy to construct. Even if a full conditional cannot be directly sampled, its density is proportional to $f(\theta)$, since

$$\pi(\theta_k | \theta_{-k}, y) = \frac{\pi(\theta | y)}{\pi(\theta_{-k} | y)} \propto_{\theta_k} f(\theta) \quad (5)$$

where the proportionality is in θ_k only (for fixed θ_{-k}). The density of the full conditional is therefore known (up to a constant scaling), so techniques described in the following subsections may be used.

Performance of Gibbs sampling can sometimes be improved by modifying the algorithm. For example, the order in which the pieces are sampled can affect the mixing rate (e.g., [8]). Also, replacing some of the full conditional distributions with (partial) posterior marginals results in a *partially collapsed* Gibbs sampler, which may have better sampling properties [9], though must be implemented carefully to preserve the stationary distribution (e.g., [10]).

Even when a Gibbs sampler is easy to implement, its mixing can be arbitrarily slow. This happens especially when there is a high degree of posterior dependence among the pieces of θ , such as when some pieces are highly correlated, or when the posterior density exhibits multiple modes offset “diagonally” from each other. Mixing may be improved by alternating Gibbs sampler cycles with steps of some other kind of MCMC, or by special modifications described in the following subsections.

4.2 Auxiliary variables

Gibbs sampling can be facilitated by techniques that involve sampling more than just the parameter θ . *Data augmentation* involves adding latent variables, usually as

intermediaries in a hierarchical structure, that make full conditionals easier to sample. *Parameter expansion* involves creating extra dimensions in the parameter space that do not affect the Bayesian model, but allow a faster-mixing Markov chain to be constructed.

Data augmentation is natural in models that are defined using random effects. The random effects simply become latent variables to be sampled with the parameters. But it can also be used to add purely artificial latent variables designed to make full conditionals easy to sample. For example, data modeled with a location-scale t -distribution lacks any direct partial conjugacy properties. Nonetheless, a t -distribution can be represented as a scale mixture of normal distributions, with an inverse gamma distribution for the scale (variance). The scale variables then become the latent variables. Both the normal and inverse gamma distributions enjoy partial conjugacy properties that make full conditionals easy to sample. See [7], Section 12.1, for details.

Parameter expansion involves defining a redundant parameter ρ unrelated to the model itself and supplying it with an arbitrary prior density. The expanded parameterization (θ, ρ) is then reparameterized in a way specially chosen to improve Gibbs sampler performance. A basic example can be found in Section 12.1 of [7]. It is sometimes possible to use an improper prior on ρ . This leads to a Gibbs sampler that lacks a stationary distribution, but may still be able to produce valid posterior samples (see [11]).

Parameter expansion is typically used in conjunction with data augmentation, whence it is known as parameter expansion-data augmentation (PX-DA) [12].

4.3 Slice sampling

One general-purpose method to sample from an arbitrary univariate continuous density is to first sample uniformly from the bivariate (unit area) region beneath its graph and then retain only the horizontal coordinate. The uniform sampling could be performed by a simple two-step Gibbs sampler, alternating between vertical and horizontal sampling. This general approach is called *slice sampling* [13]. It can be interpreted as a special auxiliary variables method, with the vertical coordinate representing the auxiliary variable.

For a multivariate θ , slice sampling can be performed on one univariate piece at a time, as in a Gibbs sampler. Specifically, if θ_k is continuous and univariate, then the slice sampler first samples v uniformly from interval $(0, f(\theta))$, then samples θ_k uniformly from $\{\theta_k : f(\theta) > v\}$. Sampling is simplest when the latter set is an interval with easily computed endpoints, but adaptive methods are available for when this is not the case [13].

Though multivariate versions of slice sampling exist (e.g., [14]), practical implementations are often univariate and implemented as a single step within a Gibbs sampler framework, for continuous pieces that would otherwise be difficult to sample.

4.4 Metropolis-Hastings

A general approach to posterior sampling is to perform a carefully controlled random walk over the parameter space. The steps are chosen such that the resulting Markov chain has the posterior as its stationary distribution. This is accomplished by the *Metropolis-Hastings algorithm*.

In one popular version, the properties of the algorithm are determined by the choice of a random walk. The choice is arbitrary, but it is often such that each step is

easy to simulate and can transition from any point in the parameter space to any other point. Let $T(\theta'|\theta)$ be its *transition kernel* for a step from θ to θ' . For example, if θ is chosen according to some continuous distribution with density $\tilde{\pi}(\theta)$, then taking one step of the random walk from θ to θ' will result in θ' having density

$$\int T(\theta'|\theta) \tilde{\pi}(\theta) d\theta \quad (6)$$

(We assume T is time-invariant, although this is not necessary, provided the time dependence does not depend on the history of the Markov chain.) The density $T(\cdot|\theta)$ defines the *proposal distribution* when the current state is θ . Values of this density (up to a constant factor that does not depend on θ) must be computable.

The transitions of the Markov chain are determined by the following algorithm: let θ^{old} be the current state of the chain. Then

1. Sample *proposal* θ' from the proposal distribution at θ^{old} .
2. Compute

$$\alpha = \frac{f(\theta')/T(\theta'|\theta^{\text{old}})}{f(\theta^{\text{old}})/T(\theta^{\text{old}}|\theta')} \quad (7)$$

3. Set the next state of the chain to be

$$\theta^{\text{new}} = \begin{cases} \theta' & \text{with probability } \min(\alpha, 1) \\ \theta^{\text{old}} & \text{otherwise} \end{cases} \quad (8)$$

Note the possibility that the next state of the chain will be identical to the previous state, even if θ is continuous under the posterior. If θ' actually becomes the next state of the chain, we say that the proposal is *accepted*. The long-run fraction of times the proposal is accepted is the *acceptance rate*.

General proof that this algorithm produces a Markov chain with the posterior as its stationary distribution can be found in, for example, [15]. Convergence properties have been extensively studied [4].

One important special case is the *Metropolis algorithm*, in which the transition kernel is symmetric: $T(\theta'|\theta) = T(\theta|\theta')$. In this case, T cancels from Eq. (7), so there is no need to compute its values. If parameter θ is continuous on an open subset of a space of real vectors, a typical example is a multivariate normal proposal distribution centered at the current value (θ^{old}). The covariance matrix is arbitrary and can be chosen to make the sampling more efficient.

Proposal distributions often admit a choice of scaling that can be tuned to improve sampling efficiency. Setting the scale too large leads to a low acceptance rate, hence slow mixing due to many repeated values. Setting the scale too small leads to a high acceptance rate, but each proposal will be close to the current value, and hence the mixing will also be slow. In some cases, theoretical results are available to guide the choice of scale. For example, for the Metropolis algorithm, research suggests that the optimal acceptance rate is about 0.44 for a one-dimensional θ and quickly falls to about 0.23 as the dimension of θ increases [16].

In addition, the shape of the proposal distribution can often be tuned. Perhaps the best shapes are ones that approximate the shape of the posterior distribution, since then proposals will tend to be in directions in which the posterior is wider. While the exact shape of the posterior may not be obvious, it may still be possible to choose a proposal that has a similar covariance structure.

The scale and shape of the proposal can be tuned in an automated manner, by making a preliminary run of the algorithm during which features of the proposal are modified adaptively to improve efficiency. This stage of *adaptation* occurs prior to burn-in: The algorithm is not a Markov chain when the proposal distribution is changed based on the sampling history, so it may not be converging to the posterior. Once adaptation is declared complete, the proposal distribution is kept fixed for burn-in and for sampling.

Although it may not be obvious, exact Gibbs sampling can actually be viewed as a special case of Metropolis-Hastings (e.g., [3]). The α turns out to always equal 1 for this situation, so no tuning is needed. Also, in a Gibbs sampler context, when a piece of θ cannot be easily simulated using conventional methods, its Gibbs step may be replaced with an easier step of Metropolis for the full conditional of that piece.

Since the posterior density is analytically available (up to a constant factor), its local properties may suggest an efficient choice of proposal distribution. For a continuous posterior, *Langevin* methods use the gradient of the log posterior density at the current point to adaptively choose the proposal distribution (e.g., [16]). This provides higher optimal acceptance rates and better scaling properties than pure Metropolis, though at the expense of more computation for each step. While this is an important improvement, modern practice has evolved even further to use more global properties of the posterior density, as detailed in the next subsection.

4.5 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC), also called *hybrid* Monte Carlo, can be regarded as a special case of Metropolis-Hastings that uses a proposal involving a special set of auxiliary variables and the path of a carefully devised differential equation [17]. Computing each proposal is complicated, and perhaps expensive, but this is often compensated by achieving a high acceptance rate even when the step size is large. This results in a sequence of samples that are less dependent, and hence fewer are needed to achieve high approximation accuracy.

HMC can be applied directly to θ if the posterior is continuous and its density is continuously differentiable. Let p represent a vector of auxiliary variables having the same size as θ , but independent of θ . Specify for p an easily-sampled continuous distribution (often multivariate normal) with a continuously differentiable density proportional to $g(p)$. Define

$$H(\theta, p) = -\ln f(\theta) - \ln g(p) \quad (9)$$

Then apply the Metropolis algorithm to sample (θ, p) jointly, with proposals generated as follows:

1. Directly generate p (independently of θ).
2. Starting from (θ^{old}, p) , follow the path $(\theta(t), p(t))$ of the differential equation system defined by

$$\frac{d\theta_k}{dt} = \frac{\partial H}{\partial p_k} \quad \frac{dp_k}{dt} = -\frac{\partial H}{\partial \theta_k} \quad (10)$$

for each element θ_k of θ and corresponding element p_k of p , up to a predetermined point t_L .

3. Let $(\theta(t_L), p(t_L))$ be the new proposed value.

In the Metropolis acceptance step, we use

$$\alpha = \exp \{H(\theta^{\text{old}}, p) - H(\theta(t_L), p(t_L))\} \quad (11)$$

Actually, if the path is followed exactly, the acceptance probability will always be 1, since value of H is constant along any differential equation path [17]. The Metropolis step is needed only because, in practice, a numerical approximation is used to solve the differential equation.

To follow the differential equation numerically, we use the *leapfrog* method [17]. This method has a number of advantages over competing methods, including stability (better preservation of H) and volume preservation, which makes Metropolis valid (i.e., makes the joint transition kernel defined by this process symmetric).

If θ is not entirely continuous, HMC may still be applicable to the continuous pieces of θ , for example, when used as part of a Gibbs sampler. Also, if the posterior density is nonzero only over a certain region, HMC can be adapted for that situation. For example, it is possible to place lower and upper bounds on the elements of θ [17].

The differential equation path of an HMC proposal has a tendency to loop back on itself, making the efficiency sensitive to the length of the path (i.e., the choice of t_L). The no-U-turn sampler (NUTS) [18] is a modification of HMC designed to avoid this behavior. Essentially, it allows for adaptive choice of the leapfrog algorithm's step size and number of steps.

In theory, the computational cost of HMC scales better with the dimension of θ than does the computational cost of ordinary (random-walk) Metropolis methods. An extensive theoretical comparison can be found in [17].

5. Cluster sampling and variation

The first non-local or cluster sampling for Monte Carlo simulation for large systems is the *Swendsen-Wang (SW) algorithm* [19]. It was designed for the Ising and Potts models and was later generalized to other systems. The main component was the random cluster model, represented via percolation models of connecting bonds. Let us start with a spin configuration $\{\sigma\}$ and generate a percolation configuration based on the spin configuration. Next, the old spin configuration is forgotten and a new spin configuration $\{\sigma'\}$ is generated according to percolation. The rule for the process is defined in order for the detailed balance condition to be satisfied. In this way, the transition leaves the equilibrium probability invariant.

Consider a Potts model with probability distribution

$$g(\sigma) = \frac{1}{Z} \exp \left(K \sum_{\langle i,j \rangle} (\delta_{\sigma_i, \sigma_j} - 1) \right) \quad (12)$$

where K is the coupling strength; the spins take on the values $1, 2, \dots, q$, e.g. $\sigma_i = 1, 2, \dots, q$; $\delta_{\sigma_i, \sigma_j}$ is the Kronecker delta, which equals one whenever $\sigma_i = \sigma_j$ and zero otherwise; the summation goes through nearest neighbor pairs; and Z is the partition function.

A SW Monte Carlo move is based on the following two steps: the first step transforms a Potts configuration to a bond configuration, and the second transforms back from bond to a new Potts configuration.

1. If $\sigma_i = \sigma_j$, a bond $n_{ij} = 1$ is created stochastically between neighbor sites i and j with a probability of $1 - (e)^{-K}$. Otherwise, no bond will be present and the bond variable is set to $n_{ij} = 0$.
2. Clusters are identified as sets of sites connected by bonds (otherwise isolated sites). If there is a connected path of bonds joining two sites, they are said to be in the same cluster. A new Potts value is assigned to each cluster, chosen with equal probability among 1 to q . The new Potts variable σ' is determined as the value of the cluster it belongs to.

With this approach, every state can be reached from any other state in one move with a non-zero probability. The two steps leave the probability distribution invariant and the method generates an equilibrium distribution Eq. (12).

One variation of the SW method is generalizing it to arbitrary sampling probabilities defined on graph partitions, which is achieved through considering it as a Metropolis-Hastings algorithm and computing the acceptance probability of the proposed Monte Carlo move [20]. The new inference algorithm begins by calculating graph edge weights using local image features and then is followed by two iterative steps: *Cluster Graph*: cutting the edges probability using their weights, to form connected components; *Relabel Graph*: selecting one connected component, and simultaneously flipping the partition of all its vertices in a probabilistic way. Accordingly, instead of flipping a single vertex as in Gibbs sampler, the split, merge, and re-grouping of a chunk of the graph are realized with this strategy.

The generalized cluster sampling implements ergodic and reversible Markov chain jumps on graph partitions. It is applicable to arbitrary posterior probabilities or energy functions in the space of graphs. Examples in image analysis (e.g., image segmentation) demonstrate that the cluster Monte Carlo is more efficient than the classical Gibbs sampler and performs better than the graph cuts and belief propagation.

6. Software implementation

In the statistics community, the first development of practical general-purpose software for MCMC was the BUGS (Bayesian inference using Gibbs sampling) project, starting in 1989. The original implementation, designed for the Windows operating system, was WinBUGS, which included a graphical interface. When development of WinBUGS ended, the OpenBUGS project was created as a successor. This software uses a special model specification language, the “BUGS language,” that is remarkably flexible. Usually, the analyst only needs to specify the model in the BUGS language and then leave the construction of appropriate samplers to the software. The basic structure is a Gibbs sampler, but the pieces may be sampled using specialized methods.

Inspired by BUGS, a parallel effort called JAGS (Just another Gibbs sampler) was developed. Like BUGS, it is based on Gibbs sampling and, in principle, requires the analyst to specify only a model (written in a variant of the BUGS language), leaving the construction of samplers to an automated engine. It tends to be faster than OpenBUGS, is more actively developed, and features better integration with the R language. It also incorporates efficient slice samplers in some of its steps. JAGS is entirely open-source and has versions for many operating systems.

PyMC’s development was an effort to generalize the process of building Metropolis-Hastings samplers, making MCMC more accessible to non-statisticians. It is now a Python package helping users define stochastic models and construct

Bayesian posterior samples. A large number of problems are suitable for PyMC due to its flexibility and extensibility. Key features include ability to fit Bayesian statistical models via MCMC and other algorithms; a large set of well-documented statistical distributions; a module for modeling Gaussian processes; sampling loops can be manipulated manually, etc.

A more recently introduced tool (since 2012) is the language Stan, which remains under active development (as of this writing). Stan allows model specification, but in an inherently more flexible way than BUGS or its variants. Software for compiling Stan includes the option for MCMC using HMC and NUTS. It therefore tends to produce more nearly independent samples than software based on Gibbs sampling. (There are also options for inference not based on sampling, such as variational methods.) The Stan software integrates with R, Python, MATLAB, Julia, and Stata.

7. Conclusion

This chapter has merely touched upon the important concepts and methods of modern MCMC. Routine-use software automating the construction of samplers is also introduced. There are many good references that provide more detailed theoretical or practical treatment and further extensions, based on which future research can be developed.

Author details

Michelle Y. Wang^{1,2,3*} and Trevor Park¹


1 Department of Statistics, University of Illinois at Urbana-Champaign, USA

2 Department of Psychology, University of Illinois at Urbana-Champaign, USA

3 Department of Bioengineering, University of Illinois at Urbana-Champaign, USA

*Address all correspondence to: ymw@illinois.edu

IntechOpen

© 2020 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Tierney L, Kadane JB. Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*. 1986; **81**(393):82-86
- [2] Jordan MI, Ghahramani Z, Jaakkola TS, Saul LK. An introduction to variational methods for graphical models. *Machine Learning*. 1999; **37**: 183-233
- [3] Geyer CJ. Introduction to Markov chain Monte Carlo. In: Brooks S, Gelman A, Jones GL, Meng X-L, editors. *Handbook of Markov Chain Monte Carlo*. Boca Raton: CRC Press; 2011. pp. 3-48
- [4] Tierney L. Markov chains for exploring posterior distributions. *Ann. Statist.* 1994; **22**(4):1701-1728
- [5] Gelman A, Shirley K. Inference from simulations and monitoring convergence. In: Brooks S, Gelman A, Jones GL, Meng X-L, editors. *Handbook of Markov Chain Monte Carlo*. Boca Raton: CRC Press; 2011. pp. 163-174
- [6] Flegal JM, Jones GL. Implementing MCMC: Estimating with confidence. In: Brooks S, Gelman A, Jones GL, Meng X-L, editors. *Handbook of Markov Chain Monte Carlo*. Boca Raton: CRC Press; 2011. pp. 175-197
- [7] Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB. *Bayesian Data Analysis*. 3rd ed. Boca Raton: CRC Press; 2013. p. 667
- [8] Roberts GO, Sahu SK. Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler. *Journal of the Royal Statistical Society: Series B: Methodological*. 1997; **59**(2):291-317
- [9] Van Dyk DA, Park T. Partially collapsed Gibbs samplers: Theory and methods. *Journal of the American Statistical Association*. 2008; **103**(482): 790-796
- [10] Van Dyk DA, Jiao X. Metropolis-Hastings within partially collapsed Gibbs samplers. *Journal of Computational and Graphical Statistics*. 2015; **24**(2):301-327
- [11] Hobert JP. Stability relationships among the Gibbs sampler and its subchains. *Journal of Computational and Graphical Statistics*. 2001; **10**(2): 185-205
- [12] Liu JS, Wu YN. Parameter expansion for data augmentation. *Journal of the American Statistical Association*. 1999; **94**:1264-1274
- [13] Neal RM. Slice sampling. *Ann. Statist.* 2003; **31**(3):705-741
- [14] Damien P, Wakefield J, Walker S. Gibbs sampling for Bayesian non-conjugate and hierarchical models by using auxiliary variables. *Journal of the Royal Statistical Society, Series B: Statistical Methodology*. 1999; **61**(2): 331-344
- [15] Tierney L. A note on Metropolis-Hastings kernels for general state spaces. *Ann. Appl. Probab.* 1998; **8**(1):1-9
- [16] Roberts GO, Rosenthal JS. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*. 2001; **16**(4):351-367
- [17] Neal RM. MCMC using Hamiltonian dynamics. In: Brooks S, Gelman A, Jones GL, Meng X-L, editors. *Handbook of Markov Chain Monte Carlo*. Boca Raton: Chapman & Hall; 2011. pp. 113-162
- [18] Hoffman MD, Gelman A. The No-U-turn sampler: Adaptively setting path

lengths in Hamiltonian Monte Carlo.
Journal of Machine Learning Research.
2014;**15**:1593-1623

[19] Swendsen RH, Wang JS.
Nonuniversal critical dynamics in
Monte Carlo simulations. Physical
Review Letters. 1987;**58**(2):86-88

[20] Barbu A, Zhu SC. Generalizing
Swendsen-Wang to sampling arbitrary
posterior probabilities. IEEE
Transactions on Pattern Analysis and
Machine Intelligence. 2005;**27**(8):
1239-1253

IntechOpen