1228 Probabilistic Graphical Models
H. Zhu and J. Boedecker
May 1, 2026

Inference with Monte Carlo Methods

Contents

1 Monte Carlo methods

1.1 Monte Carlo integration

We often want to compute the expected value of some function of a random variable, $\mathbf{E}\, f(X)$. This requires computing the following integral:

$$\mathbf{E}\, f(X) = \int f(x)p(x)\, dx, \tag{1.1}$$

where $x \in \mathbf{R}^n$, the function $f\colon \mathbf{R}^n \to \mathbf{R}^m$, and $p(x)$ is the target distribution of random variable X. Note that in many cases, the target distribution may be some posterior $p(x \mid y)$, which can be hard to compute. In such problems, instead, we often work with the unnormalized distribution, $\tilde{p}(x) = p(x, y)$, and then normalize the results using

$$Z = \int p(x, y)\, dx = p(y).$$

In low dimensions (up to, say, 3), we can compute the above integral efficiently using numerical integration, which (adaptively) computes a grid, and then evaluates the function at each point on the grid. But this does not scale to higher dimensions.

An alternative approach is to draw multiple (say, n) random samples, $x \sim p(x)$, and then to compute

$$\mathbf{E}\, f(X) \approx \frac{1}{n} \sum_{i=1}^{n} f(x_i).$$

This is called *Monte Carlo* (MC) *integration*. It has the advantage over numerical integration that the function is only evaluated in places where there is non-negligible probability, so it does not need to uniformly cover the entire space.

If we denote the exact mean by $\mu = \mathbf{E}\, f(X)$, and the MC approximation by $\hat{\mu}$, according to the central limit theorem, it can be shown that with independent samples,

$$(\hat{\mu} - \mu) \to \mathcal{N}\left(0, \frac{\hat{\sigma}^2}{n}\right),$$

where

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - \hat{\mu})^2.$$

Thus for large enough n, we have

$$\mathbf{P}\left(\hat{\mu} - 1.96\sqrt{\frac{\hat{\sigma}^2}{n}} \leq \mu \leq \hat{\mu} + 1.96\sqrt{\frac{\hat{\sigma}^2}{n}}\right) \approx 0.95.$$

The term $\sqrt{\hat{\sigma}^2/n}$ is called the (numerical or empirical) *standard error*, and is an estimate of our uncertainty about our estimate of μ. The remarkable thing to note about the above results is that the standard error in the estimate, is theoretically independent of the dimensionality of the integral.

Example. *Estimating π by Monte Carlo integration.* A *(Euclidean) ball* (or just ball) in $\mathbf{R}^n$ has the form

$$B(x_c, r) = \{x \mid \|x - x_c\|_2 \leq r\} = \{x \mid (x - x_c)^T (x - x_c) \leq r^2\},$$

where $r > 0$, and $\|\cdot\|_2$ denotes the Euclidean norm, *i.e.*, $\|u\|_2 = (u^T u)^{1/2}$. The vector $x_c \in \mathbf{R}^n$ is the *center* of the ball and the scalar r is its *radius*; $B(x_c, r)$ consists of all points within a distance r of the center x_c.

Specifically, let $B(r) = \{(x, y) \mid x^2 + y^2 \leq r^2\}$ denotes a ball in $\mathbf{R}^2$ centered in the origin with radius r, we know that its area is πr^2, but it is also equal to the following definite integral

$$S = \int_{-r}^{r} \int_{-r}^{r} I_B(x, y)\; dxdy,$$

where $I_B(x, y)$ is an indicator function of set $B(r)$ which is 1 for points inside the ball, and 0 outside. Hence, the constant $\pi = S/r^2$. We can approximate this by Monte Carlo integration. Let $p(x)$ and $p(y)$ be uniform distribution on $[-r, r]$, so

$p(x) = p(y) = 1/(2r)$ for all $x, y \in [-r, r]$, and 0 otherwise. Then

$$\begin{aligned}\pi = \frac{1}{r^2} S &= \frac{1}{r^2}(2r)(2r) \iint I_B(x,y) p(x) p(y)\ dxdy \\ &= \frac{1}{r^2} 4r^2 \iint I_B(x,y) p(x) p(y)\ dxdy \\ &\approx 4 \times \frac{1}{n} \sum_{i=1}^{n} I_B(x_i, y_i).\end{aligned}$$

1.2 Sampling from simple distributions

The main computational challenge of MC integration is to efficiently generate samples from the probability distribution $p(x)$. In this section, we discuss a sampling method that is suitable for parametric univariate distributions. These can be used as building blocks for sampling from more complex multivariate distributions.

The simplest method for sampling from a univariate distribution is based on the inverse probability transform. Let F be a *cumulative density function* (CDF) of some distribution we want to sample from, and let F^{-1} be its inverse. If $U \sim \mathcal{U}(0,1)$ is a uniform random variable, then $F^{-1}(U) \sim F$. This can be easily shown as follows:

$$\begin{aligned}\mathbf{P}\left(F^{-1}(U) \leq x\right) &= \mathbf{P}(U \leq F(x)) && \text{(applying } F \text{ to both sides)} \\ &= F(x), && \text{(because } \mathbf{P}(\mathcal{U} \leq y) = y)\end{aligned}$$

where the first line follows since F is a monotonic function, and the second line follows since U is uniform on the unit interval.

Hence we can sample from any univariate distribution, for which we can evaluate its inverse CDF, as follows: generate a random number $u \sim \mathcal{U}(0,1)$ using a *pseudorandom number generator* Let u represent the height up the y axis. Then 'slide along' the x axis until you intersect the F curve, and then 'drop down' and return the corresponding x value. This corresponds to computing $x = F^{-1}(u)$. This process is illustrated in figure 1.

Example. *Sampling from an exponential distribution.* Consider the exponential distribution $\text{Exp}(\lambda)$ with density function

$$p_\lambda(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0. \end{cases}$$

The CDF is

$$F_\lambda(x) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0 \\ 0 & x < 0, \end{cases}$$

whose inverse is the quantile function

$$F_\lambda^{-1}(u) = -\frac{\log(1-u)}{\lambda}$$

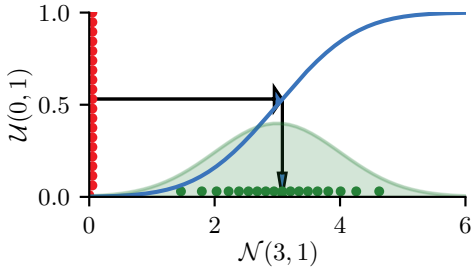

Figure 1 Sampling from $\mathcal{N}(3,1)$ using an inverse CDF. The blue and green curves show the CDF and PDF of the target distribution, and the red and greed dots denote the samples from $\mathcal{U}(0,1)$ and $\mathcal{N}(3,1)$, respectively.

with domain $\mathbf{dom}\, F_\lambda^{-1} = [0,1)$. If $U \sim \mathcal{U}(0,1)$, we know that $F_\lambda^{-1}(U) \sim \text{Exp}(\lambda)$. So we can sample from the exponential distribution by first sampling from the uniform and then transforming the results using $-\log(1-u)/\lambda$.

1.3 Rejection sampling

Suppose we want to sample from the target distribution

$$p(x) = \tilde{p}(x)/Z_p,$$

where $\tilde{p}(x)$ is the unnormalized version, and

$$Z_p = \int \tilde{p}(x)\, dx$$

is the (possibly unknown) normalization constant. One of the simplest approaches to this problem is *rejection sampling*.

In rejection sampling, we require access to a *proposal distribution* $q(x)$ which satisfies $Cq(x) \geq \tilde{p}(x)$, for some constant C. The function $Cq(x)$ provides an upper envelop for $\tilde{p}$. We can use the proposal distribution to generate samples from the target distribution as follows. For each sample, we first sample $x_i \sim q(x)$, which corresponds to picking a random x axis location, and then we sample $u_i \sim \mathcal{U}(0, Cq(x_i))$, which corresponds to picking a random height (y axis location) under the envelope. If $u_i > \tilde{p}(x_i)$, we reject the sample, otherwise we accept it. This process is illustrated in a 1-dimensional example in figure 2.

We now show this procedure is correct. First note that the probability of any given sample x_i being accepted equals the probability of a sample $u_i \sim \mathcal{U}(0, Cq(x_i))$ being less than or equal to $\tilde{p}(x_i)$, *i.e.*,

$$q(\text{accept} \mid x_i) = \int_0^{\tilde{p}(x_i)} \frac{1}{Cq(x_i)}\, du = \frac{\tilde{p}(x_i)}{Cq(x_i)}.$$

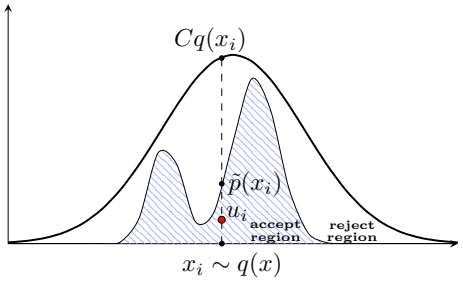

Figure 2 Schematic illustration of rejection sampling. The acceptance region is shown shaded, and the rejection region is the white region between the shaded zone and the upper envelope.

Therefore we have,

$$q(\text{propose and accept } x_i) = q(x_i)q(\text{accept} \mid x_i) = q(x_i)\frac{\tilde{p}(x_i)}{Cq(x_i)} = \frac{\tilde{p}(x_i)}{C}.$$

Integrating both sides give:

$$\int q(x_i)q(\text{accept} \mid x_i)\ dx_i = q(\text{accept}) = \frac{\int \tilde{p}(x_i)\ dx_i}{C} = \frac{Z_p}{C}. \tag{1.2}$$

Hence we see that the distribution of accepted points is given by the target distribution:

$$q(x_i \mid \text{accept}) = \frac{q(x_i, \text{accept})}{q(\text{accept})} = \frac{\tilde{p}(x_i)}{C}\frac{C}{Z_p} = \frac{\tilde{p}(x_i)}{Z_p} = p(x_i).$$

From (1.2) we also know that if $\tilde{p}$ is a normalized target distribution, the acceptance probability is $1/C$. Thus we might want to choose C as small as possible while still satisfying $Cq(x) \geq \tilde{p}(x)$. This means that we want to make our proposal $q(x)$ as close as possible to the target distribution $p(x)$, while still being an upper bound. But this is quite hard to achieve, especially in high dimensions. To see this, consider sampling from $p(x) = \mathcal{N}(0, \sigma_p^2 I)$ using the proposal $q(x) = \mathcal{N}(0, \sigma_q^2 I)$. Obviously we must have $\sigma_q^2 \geq \sigma_p^2$ in order to be an upper bound. In n dimensions, the optimum value is given by $C = (\sigma_q/\sigma_p)^n$. The acceptance rate is $1/C$ (since both p and q are normalized), which decreases exponentially fast with dimension. For example, if σ_q exceeds σ_p by just 1%, then in 1000 dimensions the acceptance ratio will be about 1/20000. This is a fundamental weakness of rejection sampling.

1.4 Importance sampling

We now introduce a Monte Carlo method known as *importance sampling* for approximating integrals of the form (1.1):

$$\mathbf{E}\, f(X) = \int f(x)p(x)\ dx,$$

where the function f is the target function, and $p(x)$ is the target distribution, which is often a conditional distribution of the form $p(x) = p(x \mid y)$. Since in general it is difficult to draw from the target distribution, we will instead draw from some proposal distribution $q(x)$ (which will usually depend on y). We then adjust for the inaccuracies of this by associating weights with each sample, so we end up with a weighted MC approximation:

$$\mathbf{E}\, f(X) \approx \sum_{i=1}^{n} W_i f(x_i).$$

We discuss two cases, first when the target is normalized, and then when it is unnormalized. This will affect the ways the weights are computed, as well as statistical properties of the estimator.

1.4.1 Direct importance sampling

In this section, we assume that we can evaluate the normalized target distribution $p(x)$, but we cannot sample from it. So instead we will sample from the proposal $q(x)$. We can then write

$$\int f(x)p(x)\ dx = \int f(x)\frac{p(x)}{q(x)}q(x)\ dx.$$

We require that the proposal be non-zero whenever the target is non-zero, *i.e.*, the support of $q(x)$ needs to be greater than or equal to the support of $p(x)$. If we draw n samples $x \sim q(x)$, we can write

$$\mathbf{E}\, f(X) \approx \frac{1}{n}\sum_{i=1}^{n}\frac{p(x_i)}{q(x_i)}f(x_i) = \frac{1}{n}\sum_{i=1}^{n} w_i f(x_i),$$

where we define the *importance weight* w_i for each sample as follows:

$$w_i = \frac{p(x_i)}{q(x_i)}, \quad i = 1, \ldots, n.$$

The result is an unbiased estimate of the true mean $\mathbf{E}\, f(X)$.

1.4.2 Self-normalized importance sampling

The disadvantage of direct importance sampling is that we need a way to evaluate the normalized target distribution p in order to compute the weights. It is often much easier to evaluate the unnormalized target distribution $\tilde{p}(x) = Z_p p(x)$, where $Z_p = \int \tilde{p}(x)\ dx$ is the normalization constant. The key idea is to also approximate the normalization constant Z_p with importance sampling. This method is called *self-normalized importance sampling* (SNIS). The resulting estimate is a ratio of two estimates, and hence is biased. However as the number of samples $n \to \infty$, the bias goes to zero (under some weak assumptions, see the references listed in page 18).

In more detail, SNIS is based on this approximation:

$$\mathbf{E}\, f(X) = \int f(x)p(x)\ dx = \frac{\int f(x)\tilde{p}(x)\ dx}{\int \tilde{p}(x)\ dx} = \frac{\int \left(\frac{\tilde{p}(x)}{q(x)} f(x)\right) q(x)\ dx}{\int \left(\frac{\tilde{p}(x)}{q(x)}\right) q(x)\ dx}$$
$$\approx \frac{\frac{1}{n}\sum_{i=1}^n \tilde{w}_i f(x_i)}{\frac{1}{n}\sum_{i=1}^n \tilde{w}_i}, \tag{1.3}$$

where $x_i \sim q(x)$ for all $i = 1, \ldots, n$, and $\tilde{w}_i$ is the unnormalized weight for each sample, defined as

$$\tilde{w}_i = \frac{\tilde{p}(x_i)}{q(x_i)}, \quad i = 1, \ldots, n.$$

We can write (1.3) more compactly as

$$\mathbf{E}\, f(X) \approx \sum_{i=1}^n W_i f(x_i),$$

where W_i is the normalized weight for each sample, defined as

$$W_i = \frac{\tilde{w}_i}{\sum_{i'=1}^n \tilde{w}_{i'}}, \quad i = 1, \ldots, n.$$

This is equivalent to approximating the target distribution using a weighted sum of delta functions:

$$p(x) \approx \hat{p}(x) = \sum_{i=1}^n W_i \delta(x - x_i).$$

As a byproduct of this algorithm we get the following approximation to the normalization constant:

$$Z_p \approx \hat{Z}_p = \frac{1}{n}\sum_{i=1}^n \tilde{w}_i.$$

2 Markov chain Monte Carlo

In §1, we considered non-iterative Monte Carlo methods, including rejection sampling and importance sampling, which generate independent samples from some target distribution. The trouble with these methods is that they often do not work well in high dimensional spaces. In this section, we discuss a popular method for sampling from high-dimensional distributions known as *Markov chain Monte Carlo* (MCMC).

The basic idea behind MCMC is to construct a Markov chain on the state space X whose stationary distribution is the target density $p^*(x)$ of interest. (In a Bayesian context, this is usually a posterior, $p^*(x) \propto p(x \mid y)$, but MCMC can be applied to generate samples from any kind of distribution.) That is, we perform a random walk on the state space, in such a way that the fraction of time we spend

in each state x is proportional to $p^*(x)$. By drawing (correlated) samples $x_0, x_1, \ldots$ from the chain, we can perform Monte Carlo integration with respect to p^*.

Note that the initial samples from the chain do not come from the stationary distribution, and should be discarded. The amount of time it takes to reach stationarity is called the *mixing time* or *burn-in time*. Reducing the burn-in time is one of the most important factors in making the algorithm fast.

2.1 Metropolis-Hastings algorithm

In this section, we describe the simplest kinds of MCMC algorithm known as the *Metropolis-Hastings* (MH) algorithm. The basic idea in MH is that at each step, we propose to move from the current state x to a new state x' with probability $q(x' \mid x)$, where q is called the proposal distribution (or *kernel*). The user is free to use any kind of proposal they want, subject to some conditions which we explain below. This makes MH quite a flexible method. Having proposed a move to x', we then decide whether to accept this proposal, or to reject it, according to some formula, which ensures that the long-term fraction of time spent in each state is proportional to $p^*(x)$. If the proposal is accepted, the new state is x', otherwise the new state is the same as the current state, x (*i.e.*, we repeat the sample).

If the proposal is symmetric, so $q(x' \mid x) = q(x \mid x')$, the acceptance probability is given as follows:

$$A = \min\left\{1, \ \frac{p^*(x')}{p^*(x)}\right\}.$$

We see that if x' is more probable than x, we definitely move there (since $\frac{p^*(x')}{p^*(x)} > 1$), but if x' is less probable than x, we may still move there anyway, depending on the relative probabilities. So instead of greedily moving to only more probable states, we occasionally allow 'downhill' moves to less probable states.

If the proposal is asymmetric, so $q(x' \mid x) \neq q(x \mid x')$, we need the *Hastings correction*, given by the following:

$$A = \min\{1, \alpha\},$$

where

$$\alpha = \frac{p^*(x')q(x \mid x')}{p^*(x)q(x' \mid x)} = \frac{p^*(x')/q(x' \mid x)}{p^*(x)/q(x \mid x')}.$$

This correction is needed to compensate for the fact that the proposal distribution itself (rather than just the target distribution) might favor certain states.

An important reason why MH is a useful algorithm is that, when evaluating α, we only need to know the target density up to a normalization constant. In particular, suppose $p^*(x) = \frac{1}{Z_p}\tilde{p}(x)$, where $\tilde{p}(x)$ is an unnormalized distribution and Z_p is the normalization constant. Then we have

$$\alpha = \frac{(\tilde{p}(x')/Z_p)q(x \mid x')}{(\tilde{p}(x)/Z_p)q(x' \mid x)},$$

where the Z_p's cancel. Hence, we can sample from p^* even if Z_p is unknown.

A proposal distribution q is valid or admissible if it 'covers' the support of the target. Formally, we can write this as

$$\mathbf{supp}\, p^* \subseteq \cup_x \mathbf{supp}\, q(\cdot \mid x).$$

With this, we can state the overall algorithm as follows.

Algorithm 1 METROPOLIS-HASTINGS ALGORITHM.

given proposal distribution q.

initialize x.

repeat

Sample $x' \sim q(x' \mid x)$.

Compute $\alpha \coloneqq \frac{p^*(x')q(x|x')}{p^*(x)q(x'|x)}$.

Compute acceptance probability $A \coloneqq \min\{1, \alpha\}$.

Sample $u \sim \mathcal{U}(0, 1)$.

Set new sample to

$$x \coloneqq \begin{cases} x', & u \leq A \text{ (accept)} \\ x, & u > A \text{ (reject).} \end{cases}$$

until number of iterations reached.

2.1.1 Convergence analysis

We show that the MH procedure generates samples from p^*. The MH algorithm defines a Markov chain with the following transition matrix:

$$p(x' \mid x) = \begin{cases} q(x' \mid x)A(x' \mid x) & x' \neq x \\ q(x \mid x) + \sum_{x' \neq x} q(x' \mid x)(1 - A(x' \mid x)) & \text{otherwise.} \end{cases} \tag{2.1}$$

This follows from a case analysis: if you move to x' from x, you must have proposed it (with probability $q(x' \mid x)$) and it must have been accepted (with probability $A(x' \mid x)$); otherwise you stay in state x, either because that is what you proposed (with probability $q(x \mid x)$), or because you proposed something else (with probability $q(x' \mid x)$) but it was rejected (with probability $1 - A(x' \mid x)$).

Let us analyze this Markov chain. Recall that a chain satisfies *detailed balance* if

$$p(x' \mid x)p^*(x) = p(x \mid x')p^*(x'). \tag{2.2}$$

This means in the in-flow to state x' from x is equal to the out-flow from state x' back to x, and vice versa. If a chain satisfies detailed balance, then p^* is its stationary distribution. Our goal is to prove that the MH algorithm defines a transition function p that satisfies detailed balance and hence that p^* is its stationary

distribution. (If (2.2) holds, we say that p^* is an invariant distribution with respect to the Markov transition kernel p.) To show this, we assume that the Markov chain with transition matrix given by (2.1) is ergodic and irreducible. Consider two states x and x'. Either

$$p^*(x)q(x' \mid x) < p^*(x')q(x \mid x')$$

or

$$p^*(x)q(x' \mid x) \geq p^*(x')q(x \mid x').$$

Without loss of generality, assume that $p^*(x)q(x' \mid x) > p^*(x')q(x \mid x')$. Then,

$$\alpha(x' \mid x) = \frac{p^*(x')q(x \mid x')}{p^*(x)q(x' \mid x)} < 1.$$

Hence, we have $A(x' \mid x) = \alpha(x' \mid x)$ and $A(x \mid x') = 1$. Now to move from x to x' we must first propose x' and then accept it, *i.e.*,

$$p(x' \mid x) = q(x' \mid x)A(x' \mid x) = q(x' \mid x)\frac{p^*(x')q(x \mid x')}{p^*(x)q(x' \mid x)} = \frac{p^*(x')}{p^*(x)}q(x \mid x'),$$

which indicates that

$$p^*(x)p(x' \mid x) = p^*(x')q(x \mid x'). \tag{2.3}$$

Since $A(x \mid x') = 1$, the backward probability can be written as

$$p(x \mid x') = q(x \mid x')A(x \mid x') = q(x \mid x').$$

Inserting this into (2.3), we get

$$p^*(x)p(x' \mid x) = p^*(x')p(x \mid x'), \tag{2.2}$$

so detailed balance holds with respect to p^*. This shows that given the MH transition kernel p, the target distribution p^* is the unique stationary distribution of the Markov chain, since we have assumed that the chain is ergodic and irreducible.

2.1.2 Proposal distributions

We now discuss some common proposal distributions q. Note, however, that good proposal design is often intimately dependent on the form of the target distribution (most often the posterior).

Independence sampler. If we use a proposal of the form $q(x' \mid x) = q(x')$, where the new state is independent of the old state, we get a method known as the *independence sampler*, which is similar to importance sampling (§1.4). The function $q(x')$ can be any suitable distribution, such as a Gaussian. Since Gaussian distribution has non-zero probability density on the entire state space, so it is a valid proposal for any unconstrained continuous state space.

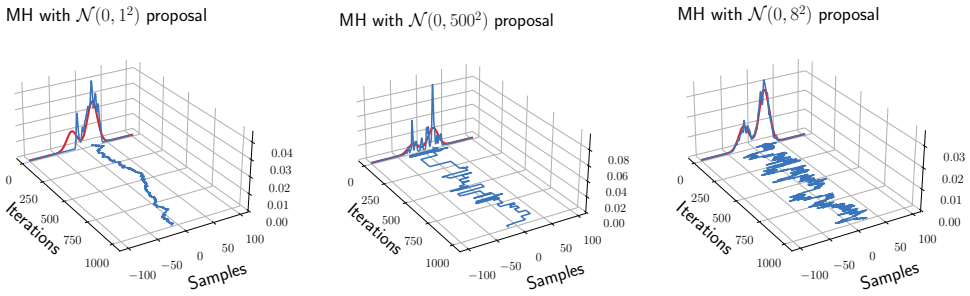

Figure 3 An example of the random walk Metropolis algorithm for sampling from a mixture of two 1-dimensional Gaussians, $\mathcal{N}(-20, 100)$ and $\mathcal{N}(20, 100)$, with mixing weights 0.3 and 0.7. The variance σ^2 of the Gaussian random walk proposal are 1^2 (*left*), 500^2 (*middle*), and 8^2 (*right*).

Random walk Metropolis. The *random walk Metropolis* (RWM) algorithm corresponds to MH with the following proposal

$$x' \sim \mathcal{N}(x, \sigma^2 I),$$

where the mean of this Gaussian distribution is the previous sample x, and the variance σ^2 is a scale factor chose to facilitate rapid mixing. This is equivalent to saying that the random vector $x' - x$ is Gaussian with mean 0 and variance $\sigma^2 I$, *i.e.*, $(x' - x) \sim \mathcal{N}(0, \sigma^2 I)$.

Example. *Sampling from a mixture of Gaussians with random walk Metropolis.* Figure 3 shows an example where we use RWM to sample from a mixture of two 1-dimensional Gaussians. This is a somewhat tricky target distribution, since it consists of two somewhat separated modes. It is very important to set the variance of the proposal σ^2 correctly: if the variance is too low, the chain will only explore one of the modes, as shown in the left panel, but if the variance is too large, most of the moves will be rejected, and the chain will be very sticky, *i.e.*, it will stay in the same state for a long time. This is evident from the long stretches of repeated values in the middle panel. If we set the proposal's variance just right, we get the trace in the right panel, where the samples clearly explore the support of the target distribution.

Composing proposals. If there are several proposals that might be useful, one can combine them using a mixture proposal, which is a convex combination of some base proposals:

$$q(x' \mid x) = \sum_{i=1}^{m} w_i q_i(x' \mid x),$$

where w_i are the mixing weights that sum to one. As long as each q_i is an individually valid proposal, and each $w_i > 0$, then the overall mixture proposal will also be valid. In particular, if each proposal is reversible, so it satisfies detailed balance, then so does the mixture.

2.2 Gibbs sampling

The major problems with MH are the need to choose the proposal distribution, and the fact that the acceptance rate may be low. In this section, we describe an MH method that exploits conditional independence properties of a graphical model to automatically create a good proposal, with acceptance probability 1. This method is known as *Gibbs sampling*. In physics, this method is also known as *Glauber dynamics* or the *heat bath* method. This is the MCMC analog of coordinate descent.

The idea behind Gibbs sampling is to sample each variable in turn, conditioned on the values of all the other variables in the distribution. For example, if we have variable $X \in \mathbf{R}^3$, we use

$$\begin{aligned} x_1' &\sim p(x_1' \mid x_2, x_3) \\ x_2' &\sim p(x_2' \mid x_1', x_3) \\ x_3' &\sim p(x_3' \mid x_1', x_2'). \end{aligned}$$

This readily generalizes to n-dimensional variables. (Note that if X_i is a known variable, we do not sample it, but it may be used as input to the another conditional distributions.) The expression $p(x_i' \mid x_{-i})$ is called the *full conditional* for variable X_i. In general, X_i may only depend on some of the other variables. If we represent $p(x)$ as a graphical model, we can infer the dependencies by looking at the Markov blanket of X_i, which are its neighbors in the graph, so we can write

$$x_i' \sim p(x_i' \mid x_{-i}) = p(x_i' \mid \mathbf{mb}(x_i)).$$

2.2.1 Connections to MH

It turns out that Gibbs sampling is a special case of MH where we use a sequence of proposals of the form

$$q_i(x' \mid x) = p(x' \mid x_{-i}) I_{x_{-i}}(x'_{-i}), \quad i = 1, \ldots, n,$$

for some variable $X \in \mathbf{R}^n$, where $I_{x_{-i}}$ is the indicator function. That is, we move to a new state where x_i is sampled from its full conditional, but x_{-i} is left unchanged. We now show that the acceptance rate of each such proposal is 100%, so the overall algorithm also has an acceptance rate of 100%. For each proposal q_i, we have

$$\begin{aligned} \alpha &= \frac{p(x')q_i(x \mid x')}{p(x)q_i(x' \mid x)} = \frac{p(x_i' \mid x'_{-i})p(x'_{-i})p(x_i \mid x'_{-i})}{p(x_i \mid x_{-i})p(x_{-i})p(x_i' \mid x_{-i})} \\ &= \frac{p(x_i' \mid x_{-i})p(x_{-i})p(x_i \mid x_{-i})}{p(x_i \mid x_{-i})p(x_{-i})p(x_i' \mid x_{-i})} = 1, \end{aligned}$$

where we exploited the fact that $x'_{-i} = x_{-i}$.

Example. *Gibbs sampling for Ising models.* Consider the 2-dimensional lattice $G = (X, E)$ in figure 4. We can represent the joint distribution of X as follows:

$$p(x) = \frac{1}{Z_p} \prod_{(X_i, X_j) \in E} \psi_{ij}(x_i, x_j),$$

where $\psi_{ij}(x_i, x_j)$ is named as the *potential function* of clique $C = \{X_i, X_j\}$. This is called a *lattice model*. An *Ising model* is a special case of lattice models, where the variables X_i are binary for all $i = 1, \ldots, n$. Such models are often used to represent magnetic materials. In particular, each node represents an atom, which can have a magnetic dipole, or *spin*, which is in one of two states, $+1$ and -1. In some magnetic systems, neighboring spins like to be similar; in other systems, they like to be dissimilar. We can capture this interaction by defining the clique potentials as follows:

$$\psi_{ij}(x_i, x_j) = \begin{cases} e^{J_{ij}} & x_i = x_j \\ e^{-J_{ij}} & x_i \neq x_j, \end{cases}$$

where J_{ij} is the coupling strength between nodes X_i and X_j. If two nodes are not connected in the graph, we set $J_{ij} = 0$. We assume that the weight matrix is symmetric, so $J_{ij} = J_{ji}$. Often we also assume all edges have the same strength, so $J_{ij} = J$ for all edges. Thus we have

$$\psi_{ij}(x_i, x_j) = \begin{cases} e^{J} & x_i = x_j \\ e^{-J} & x_i \neq x_j. \end{cases}$$

To perform the sampling for an n-dimensional random vector X following an Ising model, we need to compute the full conditional:

$$p(x_i \mid x_{-i}) \propto \prod_{X_j \in \mathbf{adj}(X_i)} \psi_{ij}(x_i, x_j),$$

for all $i = 1, \ldots, n$. In the case of an Ising model with edge potentials $\psi_{ij}(x_i, x_j) = \exp(J x_i x_j)$, where $x_i, x_j \in \{-1, +1\}$, the full conditional becomes

$$\begin{aligned} p(x_i = +1 \mid x_{-i}) &= \frac{\prod_{X_j \in \mathbf{adj}(X_i)} \psi_{ij}(x_i = +1, x_j)}{\prod_{X_j \in \mathbf{adj}(X_i)} \psi_{ij}(x_i = +1, x_j) + \prod_{X_j \in \mathbf{adj}(X_i)} \psi_{ij}(x_i = -1, x_j)} \\ &= \frac{\exp(J \sum_{X_j \in \mathbf{adj}(X_i)} x_j)}{\exp(J \sum_{X_j \in \mathbf{adj}(X_i)} x_j) + \exp(-J \sum_{X_j \in \mathbf{adj}(X_i)} x_j)} \\ &= \frac{\exp(J\eta_i)}{\exp(J\eta_i) + \exp(-J\eta_i)}, \end{aligned}$$

where J is the coupling strength, and $\eta_i = \sum_{X_j \in \mathbf{adj}(X_i)} x_j$. It is easy to see that $\eta_i = x_i(a_i - d_i)$, where a_i is the number of neighbors that agree with (have the same sign as) node X_i, and d_i is the number of neighbors who disagree. If this number is equal, the 'forces' on X_i cancel out, so the full conditional is uniform.

2.2.2 Metropolis within Gibbs

When implementing Gibbs sampling, we have to sample from the full conditionals. If the distributions are conjugate, we can compute the full conditional in closed form, but in the general case, we will need to devise special algorithms to sample from the full conditionals. One approach is to use the MH algorithm, which is called *Metropolis within Gibbs*. In particular, to sample $x_i' \sim p(x_i' \mid x_{1:i-1}', x_{i+1:n})$, we proceed in 4 steps:

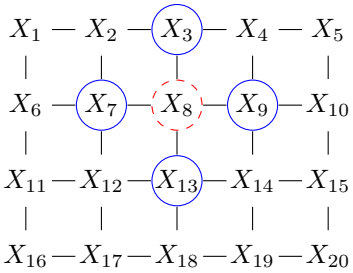

Figure 4 A 2-dimensional lattice for random vector $X \in \mathbf{R}^{20}$ represented as a undirected graph. The red node X_8 is independent of the other nodes given its neighbors (blue nodes).

1. Propose $x_i'' \sim q(x_i'' \mid x_i)$.
2. Compute the acceptance probability $A_i = \min\{1, \alpha_i\}$, where
$$\alpha_i = \frac{p(x_i'' \mid x'_{1:i-1}, x_{i+1:n}) q(x_i \mid x_i'')}{p(x_i \mid x'_{1:i-1}, x_{i+1:n}) q(x_i'' \mid x_i)}.$$
3. Sample $u \sim \mathcal{U}(0, 1)$.
4. Set $x_i' = x_i''$ if $u < A_i$, and $x_i' = x_i$ otherwise.

2.3 Hamiltonian Monte Carlo

Many MCMC algorithms perform poorly in high dimensional spaces, because they rely on a form of random search based on local perturbations. In this section, we discuss a method known as *Hamiltonian Monte Carlo* (HMC), that leverages gradient information to guide the local moves.

2.3.1 Hamiltonian mechanics

Consider a particle rolling around an energy landscape. We can characterize the motion of the particle in terms of its position $\theta \in \mathbf{R}^n$, and its momentum $v \in \mathbf{R}^n$. The set of possible values for (θ, v) is called the *phase space*. We define the Hamiltonian function for each point in phase space as follows:

$$\mathcal{H}(\theta, v) = \mathcal{E}(\theta) + \mathcal{K}(v),$$

where $\mathcal{E}(\theta)$ is the *potential energy*, $\mathcal{K}(v)$ is the *kinetic energy*, and the Hamiltonian is the total energy. In a physical setting, the potential energy is due to the pull of gravity, and the momentum is due to the motion of the particle. In a statistical setting, we often take the potential energy to be

$$\mathcal{E}(\theta) = -\log \tilde{p}(\theta),$$

where $\tilde{p}(\theta)$ is a possibly unnormalized distribution, such as $p(\theta \mid \mathcal{D})$, and the kinetic energy to be

$$\mathcal{K}(v) = \frac{1}{2} v^T \Sigma^{-1} v, \tag{2.4}$$

where $\Sigma \in \mathbf{S}_{++}^n$ is a symmetric, positive definite matrix, known as the *mass matrix*.

Stable orbits are defined by trajectories in phase space that have a constant energy. The trajectory of a particle within an energy level set can be obtained by solving the following continuous time differential equations, known as *Hamilton's equations*:

$$\begin{cases} \dfrac{d\theta}{dt} = \dfrac{\partial \mathcal{H}}{\partial v} = \dfrac{\partial \mathcal{K}}{\partial v} \\ \dfrac{dv}{dt} = -\dfrac{\partial \mathcal{H}}{\partial \theta} = -\dfrac{\partial \mathcal{E}}{\partial \theta}. \end{cases}$$

To see why energy is conserved, note that

$$\frac{d\mathcal{H}}{dt} = \sum_{i=1}^{n} \left(\frac{\partial \mathcal{H}}{\partial \theta_i} \frac{d\theta_i}{dt} + \frac{\partial \mathcal{H}}{\partial v_i} \frac{dv_i}{dt} \right) = \sum_{i=1}^{n} \left(\frac{\partial \mathcal{H}}{\partial \theta_i} \frac{\partial \mathcal{H}}{\partial v_i} - \frac{\partial \mathcal{H}}{\partial \theta_i} \frac{\partial \mathcal{H}}{\partial v_i} \right) = 0.$$

Intuitively, we can understand this result as follows: a satellite in orbit around a planet will 'want' to continue in a straight line due to its momentum, but will get pulled in towards the planet due to gravity, and if these forces cancel, the orbit is stable. If the satellite starts spiraling towards the planet, its kinetic energy will increase but its potential energy will decrease. Note that the mapping from (θ_t, v_t) to $(\theta_{t+\Delta t}, v_{t+\Delta t})$ for some time increment Δt is invertible for small enough time steps. Furthermore, this mapping is volume preserving, so has a Jacobian determinant of 1. These facts will be important later when we turn this system into an MCMC algorithm.

2.3.2 Integrating Hamilton's equations

In this section, we discuss how to simulate Hamilton's equations in discrete time.

Euler's method. The simplest way to model the time evolution is to update the position and momentum simultaneously by a small amount, known as the step size η:

$$\begin{cases} v_{t+1} = v_t + \eta \dfrac{dv}{dt}\Big|_{\theta=\theta_t, v=v_t} = v_t - \eta \dfrac{\partial \mathcal{E}}{\partial \theta}\Big|_{\theta=\theta_t} \\ \theta_{t+1} = \theta_t + \eta \dfrac{d\theta}{dt}\Big|_{\theta=\theta_t, v=v_t} = \theta_t + \eta \dfrac{\partial \mathcal{K}}{\partial v}\Big|_{v=v_t}. \end{cases}$$

If the kinetic energy has the form in (2.4), then the second expression simplifies to

$$\theta_{t+1} = \theta_t + \eta \Sigma^{-1} v_t.$$

This is known as the *Euler's method*.

Modified Euler's method. The *modified Euler's method* is slightly more accurate, and works as follows. It first update the momentum, and then update the position using the new momentum:

$$\begin{cases} v_{t+1} = v_t + \eta \dfrac{dv}{dt}\Big|_{\theta=\theta_t, v=v_t} = v_t - \eta \dfrac{\partial \mathcal{E}}{\partial \theta}\Big|_{\theta=\theta_t} \\ \theta_{t+1} = \theta_t + \eta \dfrac{d\theta}{dt}\Big|_{\theta=\theta_t, v=v_{t+1}} = \theta_t + \eta \dfrac{\partial \mathcal{K}}{\partial v}\Big|_{v=v_{t+1}} \end{cases}.$$

Unfortunately, the asymmetry of this method can cause some theoretical problems.

Leapfrog integrator. The *leapfrog integrator* is a symmetrized version of the modified Euler's method. We first perform a 'half' update of the momentum, then a full update of the position, and then finally another 'half' update of the momentum:

$$\begin{cases} v_{t+1/2} = v_t - \dfrac{\eta}{2} \dfrac{\partial \mathcal{E}}{\partial \theta}\Big|_{\theta=\theta_t} \\ \theta_{t+1} = \theta_t + \eta \dfrac{\partial \mathcal{K}}{\partial v}\Big|_{v=v_{t+1/2}} \\ v_{t+1} = v_{t+1/2} - \dfrac{\eta}{2} \dfrac{\partial \mathcal{E}}{\partial \theta}\Big|_{\theta=\theta_{t+1}} \end{cases}.$$

If we perform multiple leapfrog steps, it is equivalent to performing a half step update of v at the beginning and end of the trajectory, and alternating between full step updates of θ and v in between.

2.3.3 The HMC algorithm

We now describe how to use Hamiltonian dynamics to define an MCMC sampler in the expanded state space (θ, v). The target distribution has the form

$$p(\theta, v) = \frac{1}{Z} \exp(-\mathcal{H}(\theta, v)) = \frac{1}{Z} \exp\left(-\mathcal{E}(\theta) - \frac{1}{2} v^T \Sigma^{-1} v\right).$$

Then we can just 'throw away' the v's so that the result will be the samples θ from the desired marginal:

$$p(\theta) = \int p(\theta, v)\, dv = \frac{1}{Z_\theta} e^{-\mathcal{E}(\theta)} \int \frac{1}{Z_v} e^{-\frac{1}{2} v^T \Sigma^{-1} v}\, dv = \frac{1}{Z_\theta} e^{-\mathcal{E}(\theta)}.$$

Suppose the previous state of the Markov chain is (θ_{t-1}, v_{t-1}), to sample the next state, we proceed as follows. We set the initial position to $\theta_0' = \theta_{t-1}$, and sample a new random momentum, $v_0' \sim \mathcal{N}(0, \Sigma)$[1]. We then initialize a random

[1]Note that the Σ here denotes the covariance matrix of some Gaussian distribution, instead of the mass matrix in Hamiltonian mechanics.

trajectory in the phase space, starting at (θ'_0, v'_0), and followed for L leapfrog steps, until we get to the final proposed state $(\theta^*, v^*) = (\theta'_L, v'_L)$. If we have simulated Hamiltonian mechanics correctly, the energy should be the same at the start and the end of this process. If not, we say the HMC has *diverged*, and we reject the sample. If the energy is constant, we compute the MH acceptance probability as

$$A = \min\left\{1, \frac{p(\theta^*, v^*)}{p(\theta_{t-1}, v_{t-1})}\right\} = \min\left\{1, \exp\left(-\mathcal{H}(\theta^*, v^*) + \mathcal{H}(\theta_{t-1}, v_{t-1})\right)\right\},$$

where the transition probabilities cancel since the proposal is reversible. Finally, we accept the proposal by setting $(\theta_t, v_t) = (\theta^*, v^*)$ with probability A, otherwise we set $(\theta_t, v_t) = (\theta_{t-1}, v_{t-1})$. (In practice we don't need to keep the momentum term v, it is only used inside of the leapfrog algorithm.) The pseudocode for this procedure is shown in algorithm 2.

Note that we need to sample a new momentum at each iteration to satisfy ergodicity. To see why, recall that $\mathcal{H}(\theta, v)$ stays approximately constant as we move through phase space. If $\mathcal{H}(\theta, v) = \mathcal{E}(\theta) + \frac{1}{2} v^T \Sigma^{-1} v$, then clearly $\mathcal{E}(\theta) \leq \mathcal{H}(\theta, v) = h$ for all locations θ along the trajectory. Thus the sampler cannot reach states where $\mathcal{E}(\theta) > h$. To ensure the sampler explores the full space, we must pick a random momentum at the start of each iteration.

Algorithm 2 HAMILTONIAN MONTE CARLO.

given the number of leapfrog steps L, the step size η, and the covariance matrix Σ.

repeat

Generate random momentum $v_{t-1} \sim \mathcal{N}(0, \Sigma)$.
Set $(\theta'_0, v'_0) \coloneqq (\theta_{t-1}, v_{t-1})$.
Half step for momentum: $v'_{1/2} \coloneqq v'_0 - \frac{\eta}{2} \nabla \mathcal{E}(\theta'_0)$.
for $l = 1, \ldots, L-1$:
$\theta'_l \coloneqq \theta'_{l-1} + \eta \Sigma^{-1} v'_{l-1/2}$.
$v'_{l+1/2} \coloneqq v'_{l-1/2} - \eta \nabla \mathcal{E}(\theta'_l)$.
Full step for location: $\theta'_L \coloneqq \theta'_{L-1} + \eta \Sigma^{-1} v'_{L-1/2}$.
Half step for momentum: $v'_L \coloneqq v'_{L-1/2} - \frac{\eta}{2} \nabla \mathcal{E}(\theta'_L)$.
Obtain proposal $(\theta^*, v^*) \coloneqq (\theta'_L, v'_L)$.
Compute acceptance probability $A \coloneqq \min\{1, \exp(-\mathcal{H}(\theta^*, v^*) + \mathcal{H}(\theta_{t-1}, v_{t-1}))\}$.
Set $\theta_t \coloneqq \theta^*$ with probability A, other wise $\theta_t \coloneqq \theta_{t-1}$.

until number of iterations reached.

Bibliography

This chapter is mostly adapted from [Mur23, §11 and §12]. For more details on Monte Carlo methods see also [Liu04, RC04, KTB13, BZ20].

Except for using inverse probability transform introduced in this chapter, the *Box-Muller method* provides another approach to sample from a Gaussian distribution, which was originally developed by Box and Muller [BM58].

The statistical properties of the SNIS estimator as we mentioned briefly in §1.4.2 are discussed in detail in [RC04].

The MCMC algorithm has an interesting history. It was discovered by physicists working on the atomic bomb at Los Alamos during World War II, and was first published in the open literature [MRR+53] in a chemistry journal. An extension was published in the statistics literature [Has70], but was largely unnoticed. A special case (Gibbs sampling, §2.2) was independently invented by Geman and Geman [GG84] in the context of Ising models. But it was not until [GS90] that the algorithm became well-known to the wider statistical community. Since then it has become wildly popular in Bayesian statistics, and is becoming increasingly popular in machine learning. For more details on the MCMC theory, see *e.g.*, [GRS95] and [BZ20]. For more details on the implementation side, see *e.g.*, the article by Lao *et al.* [LSL+20].

The paper [RR01] provided some analysis regarding the mixing rate of random walk Metropolis methods. They showed that if the posterior is Gaussian, the asymptotically optimal value of σ^2 is $2.38^2/n$, where n is the dimensionality of x. This results in an acceptance rate of 0.234, which in this case, is the optimal tradeoff between exploring widely enough to cover the distribution without being rejected too often. For a more recent account of optimal acceptance rates for RWM, see also [Bed08].

The papers [TZ02] and [MKSK12] proposed that in the case where the target distribution is a posterior given some data, $p^*(x) = p(x \mid \mathcal{D})$, it is helpful to condition the proposal of MH not just on the previous hidden state, but also the visible data, *i.e.*, to use $q(x' \mid x, \mathcal{D})$. This is called *data-driven MCMC*, and a detailed introduction of this method can be found in their publications.

One can change the parameters of the MH proposal as the algorithm is running to increase efficiency. This is called *adaptive MCMC*. This allows one to start with a broad covariance (say), allowing large moves through the space until a mode is found, followed by a narrowing of the covariance to ensure careful exploration of the region around the mode. However, one must be careful not to violate the Markov property; thus the parameters of the proposal should not depend on the entire history of the chain. It turns out that a sufficient condition to ensure this is that the adaption is 'faded out' gradually over time. See *e.g.*, [AT08] for details.

It is necessary to start MCMC in an initial state that has non-zero probability. A natural approach is to first use an optimizer to find a local mode. However, at such points the gradients of the log joint are zero, which can cause problems for some gradient-based MCMC methods. Several alternatives for selecting the MCMC initial state was discussed in a tutorial by Andrieu and Thoms [AT08].

In Gibbs sampling, we can sample some of the nodes in parallel, without affecting correctness. In particular, suppose we can create a *coloring* of the (moralized) undirected graph, such that no two neighboring nodes have the same color. Then we can sample all

the nodes of the same color in parallel, and cycle through the colors sequentially. Details about this approach can be found in [GLGG11]. In general, computing an optimal coloring is NP-complete, but we can use efficient heuristics such as those in [Kub04].

The fact that the acceptance rate is 100% does not necessarily mean that Gibbs will converge rapidly, since it only updates one coordinate at a time. In some cases we can efficiently sample groups of variables at a time. This is called *blocked Gibbs sampling*, and can make much bigger moves through the state space. The articles [JKK95] and [WY02] can be referred to for more detailed information about this approach. Besides, we can sometimes gain even greater speedups by analytically integrating out some of the unknown quantities. This is called a *collapsed Gibbs sampler*, and it tends to be more efficient, since it is sampling in a lower dimensional space. Some examples about how this method works are included in [Mur23, §12.3.8].

Lattice models and Ising models are all special cases of undirected graphical models, which are also called *Markov random fields* (MRFs). For more information about probabilistic calculation on MRFs and their properties, one can refer to [Mur23, §4.3].

Hamiltonian Monte Carlo were originally derived from physics [DKPR87, Nea93, Mac03, Nea11, Bet17]. The method was originally called *hybrid Monte Carlo* [DKPR87]. It was introduced to the statistics community by Neal [Nea93], and was renamed to Hamiltonian Monte Carlo in [Mac03]. For a more detailed theory and mathematical analysis about Hamiltonian Monte Carlo methods, one can refer to [BZ20]. In practice, there are many widely used libraries for applying HMC in stochastic inference, such as PyMC [APAC+23] and TensorFlow Probability [DLT+17].

In practice, to diagnose the MCMC convergence after sampling is also very important, although we did not go into much detail about this aspect. A thorough introduction about different metrics and techniques can be found in [Mur23, §12.6].

References

[APAC+23] O. Abril-Pla, V. Andreani, C. Carroll, L. Dong, J. J. Fonnesbeck, M. Kochurov, R. Kumar, J. Lao, C. C. Luhmann, O. A. Martin, M. Osthege, R. Vieira, T. Wiecki, and R. Zinkov. PyMC: A modern, and comprehensive probabilistic programming framework in Python. *PeerJ Computer Science*, 9:e1516, 2023.

[AT08] C. Andrieu and J. Thoms. A tutorial on adaptive MCMC. *Statistics and Computing*, 18:343–373, 2008.

[Bed08] M. Bedard. Optimal acceptance rates for Metropolis algorithms: Moving beyond 0.234. *Stochastic Processes and their Applications*, 118(12):2198–2222, 2008.

[Bet17] M. Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv*, 1701.02434, 2017.

[BM58] G. E. P. Box and M. E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, 1958.

[BZ20] A. Barbu and S.-C. Zhu. *Monte Carlo Methods*. Springer, 2020.

[DKPR87] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

[DLT+17] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous. Tensorflow distributions. *arXiv*, 1711.10604, 2017.

[GG84] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

[GLGG11] J. Gonzalez, Y. Low, A. Gretton, and C. 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. JMLR Workshop and Conference Proceedings, 2011.

[GRS95] W. R. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. CRC Press, 1995.

[GS90] A. E. Gelfand and A. F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409, 1990.

[Has70] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[JKK95] C. S. Jensen, U. Kjærulff, and A. Kong. Blocking Gibbs sampling in very large probabilistic expert systems. *International Journal of Human-Computer Studies*, 42(6):647–666, 1995.

[KTB13] D. P. Kroese, T. Taimre, and Z. I. Botev. *Handbook of Monte Carlo Methods*. John Wiley & Sons, 2013.

[Kub04] M. Kubale, editor. *Graph Colorings*, volume 352 of *Contemporary Mathematics*. American Mathematical Society, 2004.

[Liu04] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2004.

[LSL+20] J. Lao, C. Suter, I. Langmore, C. Chimisov, A. Saxena, P. Sountsov, D. Moore, R. A. Saurous, M. D. Hoffman, and J. V. Dillon. tfp.mcmc: Modern Markov chain Monte Carlo tools built for modern hardware. *arXiv*, 2002.01184, 2020.

[Mac03] D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[MKSK12] J. Min, J. Kim, S. Shin, and I. S. Kweon. Efficient data-driven MCMC sampling for vision-based 6D SLAM. In *2012 IEEE International Conference on Robotics and Automation*, pages 3025–3032. IEEE, 2012.

[MRR+53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[Mur23] K. P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

[Nea93] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto Toronto, ON, Canada, 1993.

[Nea11] R. M. Neal. MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, editors, *Handbook of Markov Chain Monte Carlo*. CRC Press, 1st edition, 2011.

[RC04] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2nd edition, 2004.

[RR01] G. O. Roberts and J. S. Rosenthal. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, 16(4):351–367, 2001.

[TZ02] Z. Tu and S.-C. Zhu. Image segmentation by data-driven Markov chain Monte Carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):657–673, 2002.

[WY02] D. J. Wilkinson and S. K. H. Yeung. Conditional simulation from highly structured Gaussian systems, with application to blocking-MCMC for the Bayesian analysis of very large linear models. *Statistics and Computing*, 12(3):287–300, 2002.