$s) \propto (r_s - 1)^{-\alpha}$ for some $\alpha > 0$. But it might be possible to use a more complex function of all the r_j.

4. Study the "scanning future" method (Meirovitch 1982, Meirovitch 1985). Investigate its potential for the nonlinear filtering problem and the molecular structural optimization problem.

5. Investigate further the difference between the method of "split-track filter" and the mixture Kalman filter.

5

Metropolis Algorithm and Beyond

We have discussed in the previous chapters the important role of Monte Carlo methods in evaluating integrals and simulating stochastic systems. The most critical step in developing an efficient Monte Carlo algorithm is the simulation (sampling) from an appropriate probability distribution $\pi(\mathbf{x})$. When directly generating independent samples from $\pi(\mathbf{x})$ is not possible, we have to either opt for an *importance sampling* strategy, in which random samples are generated from a trial distribution *different* from (but close to) the target one and then weighted according to the importance ratio; or produce statistically *dependent* samples based on the idea of *Markov chain Monte Carlo* sampling. The importance sampling approach and its extensions have been discussed in Chapters 2–4. In this chapter, we introduce the cornerstone of all Markov chain-based Monte Carlo methods: the algorithm proposed in a very short paper (four pages) by Nicholas Metropolis, Arianna Rosenbluth, Marshall Rosenbluth, Augusta Teller, and Edward Teller in 1953.

Let $\pi(\mathbf{x}) = Z^{-1} \exp\{-h(\mathbf{x})\}$ be the target distribution under investigation (presumably all probability distribution functions can be written in this form), where the normalizing constant, or the *partition function*, Z, is often unknown to us. In principle, $Z = \int \exp\{-h(\mathbf{x})\} d\mathbf{x}$ is "knowable," but evaluating Z is no easier (and often harder) than the original problem of simulating from π. Motivated by computational problems in statistical physics, Metropolis et al. (1953) introduced the fundamental idea of evolving a Markov process to achieve the sampling of π. This idea, later known as the *Metropolis algorithm*, is of great simplicity and power — its variations and extensions have now been widely adopted by researchers in many

different scientific fields, including biology, chemistry, computer sciences, economics, engineering, material sciences, physics, statistics, and others.

The Metropolis algorithm can be used to generate random samples from virtually any target distribution $\pi(\mathbf{x})$ known up to a normalizing constant, regardless of its analytical complexity and its dimensionality. Although this claim is true in theory, a potential problem with these Markov-chain-based Monte Carlo methods is that the resulting samples are often highly correlated. Therefore, the estimates resulting from these samples tend to have greater (often much greater) variances than those resulting from independent samples. Various attempts have been made in many different fields (e.g., physics, chemistry, structural biology, and statistics) to overcome these limitations. Interesting research topics include the design of Markov-chain-based Monte Carlo algorithms that can generate less correlated samples, the finding of more efficient ways to use generated Monte Carlo samples, the assessment of statistical errors of the estimates, etc. Detailed discussions regarding these topics will be given in the later chapters.

5.1 The Metropolis Algorithm

The basic idea of the Metropolis algorithm is to simulate a Markov chain in the state space of $\mathbf{x}$ so that the limiting/stationary/equilibrium[1] distribution of this chain is the target distribution π. Note that in traditional Markov chain analysis, one is often given a *transition rule*[2] and is interested in knowing what the stationary distribution is (see Section 12.1 for an introduction to Markov chains), whereas in Markov chain Monte Carlo simulations, one *knows* the equilibrium distribution and is interested in prescribing an efficient transition rule so as to reach this equilibrium.

Starting with any configuration $\mathbf{x}^{(0)}$, the Metropolis algorithm proceeds by iterating the following two steps.

M1: Propose a random "unbiased perturbation" of the current state $\mathbf{x}^{(t)}$ so as to generate a new configuration $\mathbf{x}'$. Mathematically, $\mathbf{x}'$ can be seen as being generated from a *symmetric* probability transition function[3] (often called the *proposal function* or *trial proposal*) $T(\mathbf{x}^{(t)}, \mathbf{x}')$ [i.e., $T(\mathbf{x}, \mathbf{x}') = T(\mathbf{x}', \mathbf{x})$]; calculate the change $\Delta h = h(\mathbf{x}') - h(\mathbf{x}^{(t)})$.

[1] There are subtle differences among these three concepts: limiting, stationary, or equilibrium distributions. But for most practical examples, they are the same thing. See the Appendix and Karlin and Taylor (1998) for more details.

[2] A *transition rule* is a probabilistic law, or more precisely, a conditional distribution, that dictates the chances of moving from one point in the state space to another.

[3] A function $T(\mathbf{x}, \mathbf{y})$ is called a probability transition function if it is non-negative and satisfies $\sum_{\text{all } \mathbf{y}} T(\mathbf{x}, \mathbf{y}) = 1$, for all $\mathbf{x}$.

M2: Generate a random number $U \sim \text{Uniform}[0,1]$. Let $\mathbf{x}^{(t+1)} = \mathbf{x}'$ if

$$U \le \pi(\mathbf{x}')/\pi(\mathbf{x}^{(t)}) \equiv \exp(-\Delta h),$$

and let $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)}$ otherwise.

A more casual but perhaps better known description of the Metropolis algorithm is as follows: At each iteration, (a) a small but random perturbation of the current configuration is made, (b) the "gain" in an objective function [i.e., $-h(\mathbf{x})$] resulting from this perturbation is computed, (c) a random number U is generated independently, and (d) the new configuration is accepted if $\log(U)$ is smaller than or equal to the "gain" and is rejected otherwise. Heuristically, the Metropolis algorithm is constructed based on a "trial-and-error" strategy.

Metropolis et al. (1953) restricted their choices of the "perturbation rule" to the symmetric ones. According to this perturbation rule, the chance of obtaining $\mathbf{x}'$ from perturbing $\mathbf{x}$ is always equal to that of obtaining $\mathbf{x}$ from perturbing $\mathbf{x}'$. Intuitively, this means that there is no "trend bias" at the proposal stage. Mathematically, this symmetry requirement can be expressed as

$$T(\mathbf{x}, \mathbf{x}') = T(\mathbf{x}', \mathbf{x}).$$

The Metropolis scheme has been extensively used in statistical physics over the past five decades and is the cornerstone of all Markov chain Monte Carlo (MCMC) techniques recently adopted and further developed in the statistics community.

As an illustration, we consider the simulation of a simple hard-shell ball model for gas. In this model, the positions of K nonoverlapping hard-shell balls, with equal diameters, are required to be uniformly distributed in the box $[0, A] \times [0, B]$. Let $(X, Y) = \{(x_i, y_i),\ i = 1, \ldots, K\}$ denote the positions of these balls. The target distribution of interest, $\pi(X, Y)$, is then *uniform* for all allowable configurations (i.e., nonoverlapping and within the box). The Metropolis algorithm for this simulation can be implemented as follows: (a) Pick a ball at random, say, the ball at position (x_i, y_i); (b) propose to move this ball to a new position $(x_i', y_i') = (x_i + \delta_1, y_i + \delta_2)$, where $\delta_j \sim N(0, \sigma_0^2)$; and (c) accept the proposed position (x_i', y_i') if it does not violate the constraints, otherwise stay put. With $K = 6$, $d = 0.8$, $A = B = 3.5$, and starting positions of the balls at regular grids, we adjusted σ_0^2 to 0.5, which gave us an acceptance rate of about 30%. Figure 5.1 shows two snapshots of this simulation: The first one was taken after 1000 iterations, and the second one taken after 2000 iterations.

Another example is the simulation of the Ising model. As described in Section 1.3, the Ising model takes a probabilistic form

$$\pi(\mathbf{x}) \propto \exp\{-U(\mathbf{x})/\beta T\},$$

where $\mathbf{x} = (x_s,\ s \in \mathcal{L})$ and $x_s = \pm 1$, $\mathcal{L}$ is a lattice space; and $U(\mathbf{x}) = -J \sum_{s \sim s'} \delta_{x_s = x_{s'}}$. To simulate from this model, one needs to prescribe a

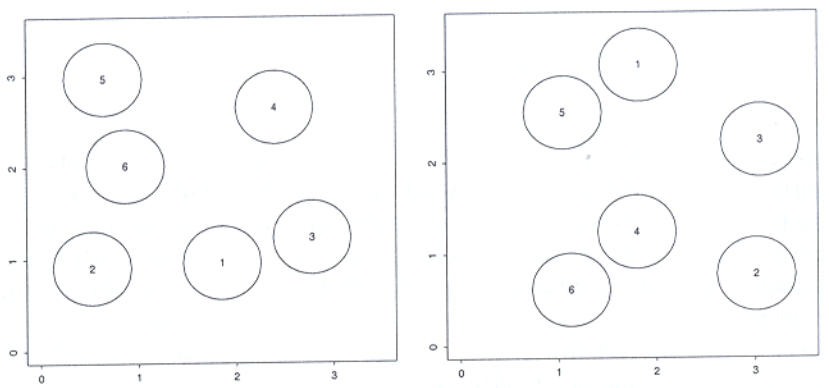

FIGURE 5.1. The simulation of a hard-shell ball model by the Metropolis algorithm. Left: after 1000 iterations; right: after 2000 iterations.

way to "perturb" the current configuration. A convenient proposal transition function is as follows: Pick a site, say, σ, at random, and negate its current value x_σ to $-x_\sigma$. Thus, the proposed new configuration $\mathbf{x}'$ differs from the initial one $\mathbf{x}$ only by a single site.

To be more concrete, consider the simulation of a one-dimensional Ising model in which $\mathbf{x} = (x_1, \ldots, x_d)$ and $U(\mathbf{x}) = -J\sum_{s=1}^{d-1} x_s x_{s+1}$. By letting $J = \mu\beta T$, we write the target distribution as

$$\pi(\mathbf{x}) = \frac{1}{Z}\exp\left\{\mu\sum_{s=1}^{d-1} x_s x_{s+1}\right\}. \tag{5.1}$$

Suppose the current configuration is $\mathbf{x}^{(t)} = (x_1^{(t)}, \ldots, x_d^{(t)})$; then the next state $\mathbf{x}^{(t+1)}$ is produced by the Metropolis rule as follows:

- Choose a site, say, the jth site at random and set its current spin $x_j^{(t)}$ to the opposite. Thus, the newly proposed configuration is $\mathbf{x}' = (x_1^{(t)}, \ldots, -x_j^{(t)}, \ldots, x_d^{(t)})$.
- Compute the Metropolis ratio. In this case, it is easy to check that the random flipping process is completely symmetric; hence, $T(\mathbf{x}^{(t)}, \mathbf{x}') = T(\mathbf{x}', \mathbf{x}^{(t)})$, and, when $j \neq 1$ or d,

$$r = \pi(\mathbf{x}')/\pi(\mathbf{x}^{(t)}) = \exp\left\{-2\mu x_j^{(t)}\left(x_{j-1}^{(t)} + x_{j+1}^{(t)}\right)\right\}.$$

- Simulate an independent uniform random variable U. Let $\mathbf{x}^{(t+1)} = \mathbf{x}'$ if $U \leq r$ and let $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)}$ otherwise.

Since this distribution has many components, it is difficult to have an overall sense of how the chain moves in the space. In Figure 5.2(a), we plot the

traces of the total magnetization, defined as $M^{(t)} = \sum_{i=1}^{d} x_i^{(t)}$, for the first 2000 steps of a simulation. In this example, we took $\mu = 1$, $d = 50$, and the "all-up" starting configuration [i.e., $\mathbf{x}^{(0)} = (1, \ldots, 1)$].

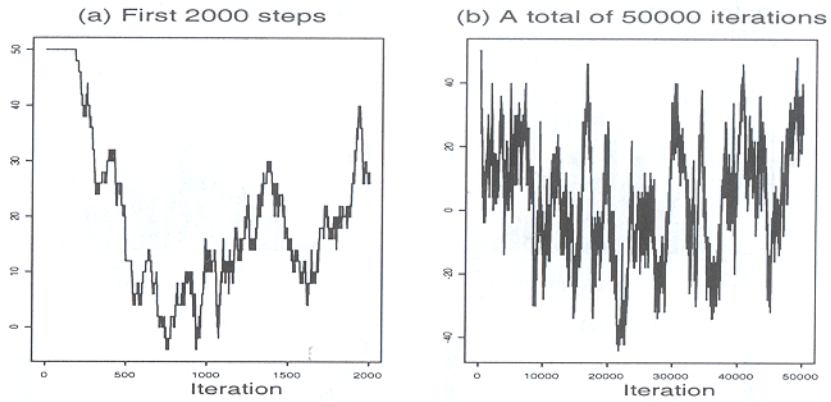

FIGURE 5.2. Simulation of the 1-D Ising model. The trace plots of (a) the first 2000 steps and (b) the total of 50,000 steps.

For this example, we can, in fact, compute the normalizing constant Z analytically and conduct an *exact* simulation (Section 2.4). Note that

$$\begin{aligned}\sum_{x_1} g(\mathbf{x}) &= \left(e^{\mu x_2} + e^{-\mu x_2}\right)\exp\left\{\mu\sum_{i=2}^{d-1} x_i x_{i+1}\right\} \\ &= \left(e^{-\mu} + e^{\mu}\right)\exp\left\{\mu\sum_{i=2}^{d-1} x_i x_{i+1}\right\}.\end{aligned}$$

We can recursively sum out x_2, x_3, etc., and obtain that

$$\sum_{x_1,\ldots,x_d} g(\mathbf{x}) = \sum_{x_d}\left[\cdots\sum_{x_2}\left\{\sum_{x_1} g(\mathbf{x})\right\}\cdots\right] = 2\left(e^{-\mu} + e^{\mu}\right)^{d-1},$$

for $d \geq 2$. Thus, the marginal distribution of x_d is $x_d = 1$ or -1 with equal probability (which is actually obvious without doing any computation). Conditional on x_d, the distribution of x_{d-1} is

$$\Pr(x_{d-1} = x_d) = e^{\mu}/(e^{\mu} + e^{-\mu}),$$

and $\Pr(x_{d-1} = -x_d) = e^{-\mu}/(e^{\mu} + e^{-\mu})$. Thus, we can recursively simulate $\mathbf{x}$ backward (Section 2.4).

We implemented both the exact simulation method and the Metropolis algorithm for the 1-D Ising model (5.1) with $\mu = 1$ and $\mu = 2$, respectively.

Figures 5.3(a) and 5.3(c) show the histograms of the total magnetization variable M from 20,000 exact samples, for model (5.1) with $\mu = 1$ and $\mu = 2$, respectively. Figures 5.3 (b) and 5.3(d) show the corresponding histograms from 20,000 Monte Carlo samples generated from 1,000,000 Metropolis sampling steps. The chosen samples were 1 in every 50 lags [i.e., $\mathbf{x}^{(50)}, \mathbf{x}^{(100)}, \ldots, \mathbf{x}^{(50k)}, \ldots$].

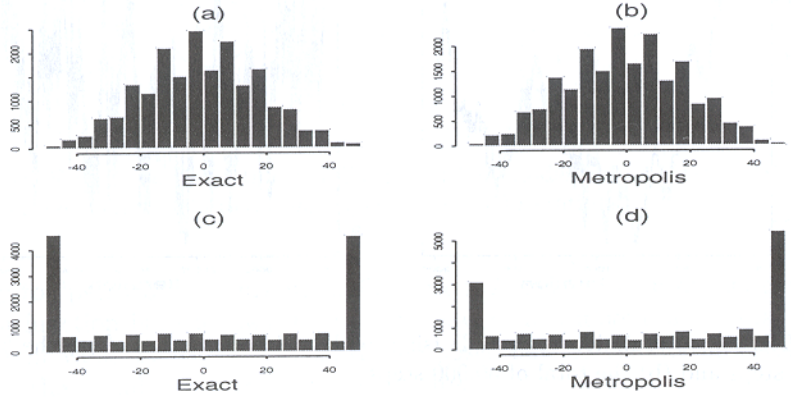

FIGURE 5.3. Histograms of the total magnetization M. (a) and (c): using 20,000 exact samples from the 1-D Ising models with $\mu = 1$ and $\mu = 2$; (b) and (d): using 20,000 samples chosen from 1 million Metropolis steps for each model.

From these simple graphs, we can make several interesting observations: (i) The computational effort for each Metropolis step is roughly 1/50th of that of exact simulation, so 1 million Metropolis steps took roughly the same CPU time as the production of 20,000 exact samples; (ii) when $\mu = 1$, the samples produced by the Metropolis algorithm were almost as good as those produced by independent sampling (at least to our eyes); (iii) when $\mu = 2$, the independent sampling showed an obvious advantage. These observations, in fact, reflect some important features of and deeper issues about the Metropolis algorithm: Each step of the Metropolis algorithm is usually very simple, but the Monte Carlo samples produced by a Metropolis sampler may become very "sticky"(e.g., getting stuck in local modes) in distributions with "low temperature" (i.e., high-energy barrier). In fact, the "stickiness" is already observable when $\mu = 1$ from Figure 5.2 (a): If one starts from an "all-up" configuration, it took about 1000 steps to get to the "ball park" of the interesting region and it took about 2000 steps to complete an "up-down" cycle. Quantitative measurement of this "stickiness" is often expressed as *autocorrelations*. A lag-k autocorrelation for a time series $M^{(1)}, M^{(2)}, \ldots$, is defined as

$$\rho_k = \text{corr}(M^{(1)}, M^{(k+1)}),$$

under their stationary distribution. Higher autocorrelations imply that the produced samples are stickier. Figure 5.4 shows the autocorrelation plots (i.e., a plot for ρ_k versus k) for those Monte Carlo samples in Figures 5.3(b) and 5.3(d), respectively.

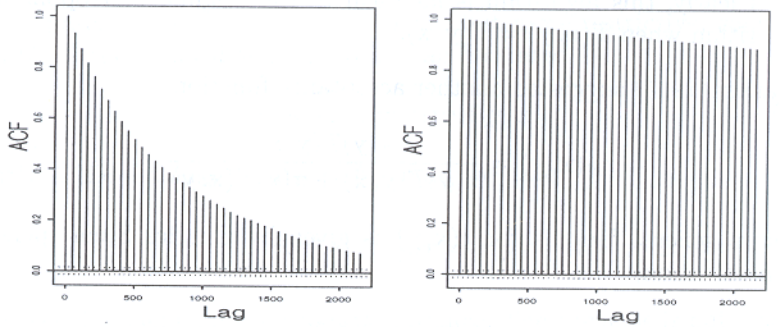

FIGURE 5.4. Autocorrelation plots of the time series $M^{(n)}$ produced by the Metropolis sampler for the 1-D Ising model with (a) $\mu = 1$ and (b) $\mu = 2$.

One observes that the 2000-lag autocorrelation of the $M^{(n)}$ series produced by the Metropolis sampler for $\mu = 2$ is still as high as about 0.9, implying that roughly one independent sample is as good as 20,000 Metropolis steps. More discussions on efficiency analysis of Markov chain Monte Carlo methods are discussed in Section 5.8 and Chapter 12.

5.2 Mathematical Formulation and Hastings's Generalization

The Metropolis algorithm prescribes a *transition rule* for a Markov chain. It uses a symmetric proposal function $T(\mathbf{x}, \mathbf{y})$ to suggest a possible move and then employs an acceptance-rejection rule to "thin it down." Hastings (1970) later extended the algorithm to the case when T is not necessarily symmetric. In Hastings' generalization, the only serious restriction on the proposal function is that $T(\mathbf{x}, \mathbf{y}) > 0$ if and only if $T(\mathbf{y}, \mathbf{x}) > 0$. With this transition function, one can implement the following iteration:

Metropolis-Hastings Algorithm. Given current state $\mathbf{x}^{(t)}$:

- Draw $\mathbf{y}$ from the proposal distribution $T(\mathbf{x}^{(t)}, \mathbf{y})$.
- Draw $U \sim$ Uniform[0,1] and update

$$\mathbf{x}^{(t+1)} = \begin{cases} \mathbf{y}, & \text{if } U \le r(\mathbf{x}^{(t)}, \mathbf{y}) \\ \mathbf{x}^{(t)} & \text{otherwise.} \end{cases} \tag{5.2}$$

where Metropolis et al. (1953) and Hastings (1970) suggested using

$$r(\mathbf{x},\mathbf{y}) = \min\left\{1, \frac{\pi(\mathbf{y})T(\mathbf{y},\mathbf{x})}{\pi(\mathbf{x})T(\mathbf{x},\mathbf{y})}\right\}.$$

Clearly, this algorithm is identical to the original Metropolis algorithm when $T(\mathbf{x},\mathbf{y}) = T(\mathbf{y},\mathbf{x})$.

Barker (1965) suggested another acceptance function:

$$r_B(\mathbf{x},\mathbf{y}) = \frac{\pi(\mathbf{y})T(\mathbf{y},\mathbf{x})}{\pi(\mathbf{y})T(\mathbf{y},\mathbf{x}) + \pi(\mathbf{x})T(\mathbf{x},\mathbf{y})}.$$

A more general formula for $r(\mathbf{x},\mathbf{y})$ is given by Charles Stein (personal communication):

$$r(\mathbf{x},\mathbf{y}) = \frac{\delta(\mathbf{x},\mathbf{y})}{\pi(\mathbf{x})T(\mathbf{x},\mathbf{y})}, \tag{5.3}$$

where $\delta(\mathbf{x},\mathbf{y})$ is any *symmetric function* in $\mathbf{x}$ and $\mathbf{y}$ that makes $r(\mathbf{x},\mathbf{y}) \leq 1$ for all $\mathbf{x},\mathbf{y}$. The intuition behind the ratio $T(\mathbf{y},\mathbf{x})/T(\mathbf{x},\mathbf{y})$ is that it compensates the "flow bias" of the proposal function.

If a rejection function of the form (5.3) is used, then for any $\mathbf{y} \neq \mathbf{x}$, the actual transition probability from $\mathbf{x}$ to $\mathbf{y}$ is

$$A(\mathbf{x},\mathbf{y}) = T(\mathbf{x},\mathbf{y})r(\mathbf{x},\mathbf{y}) = T(\mathbf{x},\mathbf{y})\frac{\delta(\mathbf{x},\mathbf{y})}{\pi(\mathbf{x})T(\mathbf{x},\mathbf{y})} = \pi(\mathbf{x})^{-1}\delta(\mathbf{x},\mathbf{y}). \tag{5.4}$$

Because $\delta(\mathbf{x},\mathbf{y}) = \delta(\mathbf{y},\mathbf{x})$, we have that $\pi(\mathbf{x})A(\mathbf{x},\mathbf{y}) = \pi(\mathbf{y})A(\mathbf{y},\mathbf{x})$. This implies that the Markov chain induced by the Metropolis-Hastings rule is *reversible* and has π as its invariant distribution (the next section).

For discrete state spaces, Peskun (1973) showed that the optimal choice of $r(\mathbf{x},\mathbf{y})$ in terms of statistical efficiency is the one in the original Metropolis algorithm (see Section 13.3.1 for more details). But the issue is less clear in terms of convergence rate of the induced Markov chain (Frigessi, Distefano, Hwang and Sheu 1993, Liu 1996c) As we will show in the next section, a main criterion used in the design of a Markov transition rule such as the Metropolis-Hastings algorithm is to ensure that the target distribution $\pi(\mathbf{x})$ is the invariant distribution of this chain.

5.3 Why Does the Metropolis Algorithm Work?

We first verify that the Metropolis-Hastings algorithm prescribes a transition rule with respect to which the target distribution $\pi(\mathbf{x})$ is invariant. Let $A(\mathbf{x},\mathbf{y})$ be the *actual* transition function of the algorithm. It differs

from the proposal function $T(\mathbf{x},\mathbf{y})$ because an acceptance-rejection step is involved. We are required to show that

$$\int \pi(\mathbf{x})A(\mathbf{x},\mathbf{y})d\mathbf{x} = \pi(\mathbf{y}). \tag{5.5}$$

Fortunately, there is an easier-to-check, but more restrictive, condition than (5.5), the *detailed balance*, which can be stated as

$$\pi(\mathbf{x})A(\mathbf{x},\mathbf{y}) = \pi(\mathbf{y})A(\mathbf{y},\mathbf{x}). \tag{5.6}$$

Clearly, if the detailed balance (5.6) holds, we have

$$\int \pi(\mathbf{x})A(\mathbf{x},\mathbf{y})d\mathbf{x} = \int \pi(\mathbf{y})A(\mathbf{y},\mathbf{x})d\mathbf{x} = \pi(\mathbf{y})\int A(\mathbf{y},\mathbf{x})d\mathbf{x} = \pi(\mathbf{y}).$$

Thus, the detailed balance *ensures* invariance. The converse is not true. In Markov chain literature, chains that satisfy the detailed balance condition are called *reversible*.

We can write down $A(\mathbf{x},\mathbf{y})$ explicitly for the Metropolis algorithm: For any $\mathbf{x} \neq \mathbf{y}$, the probability that we actually make the move from $\mathbf{x}$ to $\mathbf{y}$ is equal to the proposal probability, $T(\mathbf{x},\mathbf{y})$, multiplied by the acceptance probability; that is,

$$A(\mathbf{x},\mathbf{y}) = T(\mathbf{x},\mathbf{y})\min\left\{1, \frac{\pi(\mathbf{y})T(\mathbf{y},\mathbf{x})}{\pi(\mathbf{x})T(\mathbf{x},\mathbf{y})}\right\}, \tag{5.7}$$

for $\mathbf{x} \neq \mathbf{y}$. Hence,

$$\begin{aligned}\pi(\mathbf{x})A(\mathbf{x},\mathbf{y}) &= \pi(\mathbf{x})T(\mathbf{x},\mathbf{y})\min\left\{1, \frac{\pi(\mathbf{y})T(\mathbf{y},\mathbf{x})}{\pi(\mathbf{x})T(\mathbf{x},\mathbf{y})}\right\} \\ &= \min\{\pi(\mathbf{x})T(\mathbf{x},\mathbf{y}), \pi(\mathbf{y})T(\mathbf{y},\mathbf{x})\},\end{aligned}$$

which is a *symmetric function* in $\mathbf{x}$ and $\mathbf{y}$. Thus, the detailed balance condition is satisfied.

More generally, as long as the transition function $A(\mathbf{x},\mathbf{y})$ is of the form

$$A(\mathbf{x},\mathbf{y}) = \pi(\mathbf{y})\delta(\mathbf{x},\mathbf{y}),$$

where $\delta(\mathbf{x},\mathbf{y})$ is a symmetric function in $\mathbf{x},\mathbf{y}$, one can easily verify that the detailed balance condition has to be satisfied. The difficulty in the construction of A, however, is that the symmetric function $\delta(\mathbf{x},\mathbf{y})$ has to be chosen properly so that the integral $\int A(\mathbf{x},\mathbf{y})d\mathbf{y} = 1$. One can easily check that the Metropolis transition can be written as (for $\mathbf{x} \neq \mathbf{y}$)

$$A(\mathbf{x},\mathbf{y}) = \pi(\mathbf{y})\min\left\{\frac{T(\mathbf{x},\mathbf{y})}{\pi(\mathbf{y})}, \frac{T(\mathbf{y},\mathbf{x})}{\pi(\mathbf{x})}\right\}.$$

The acceptance rule proposed by Barker (1965) corresponds to a transition (for $\mathbf{x} \neq \mathbf{y}$)

$$A(\mathbf{x},\mathbf{y}) = \pi(\mathbf{y})\frac{T(\mathbf{x},\mathbf{y})T(\mathbf{y},\mathbf{x})}{\pi(\mathbf{y})T(\mathbf{y},\mathbf{x}) + \pi(\mathbf{x})T(\mathbf{x},\mathbf{y})}.$$

It was not clear, then, which acceptance rule is better. Peskun (1973) later showed that the Metropolis rule generally works better in terms of statistical efficiency.

By the standard Markov chain theory, if the chain is irreducible[4], aperiodic[5] [this is almost surely true for the Metropolis algorithm (Tierney 1994)], and possesses an invariant distribution, then the chain will become stationary at its invariant distribution, π. Therefore, if we run this chain long enough (say, after a burn-in period of n_0 steps), the samples $\mathbf{x}_{n_0+1}, \mathbf{x}_{n_0+2}, \ldots$ produced by the chain can be regarded as approximately following the target distribution π. One then realizes the task of drawing random (but correlated) samples from a given distribution.

5.4 Some Special Algorithms

To illustrate how the Metropolis-Hastings rule is practiced, we describe a few special algorithms that have appeared frequently in the literature.

5.4.1 *Random-walk Metropolis*

Suppose the target distribution $\pi(\mathbf{x})$ is defined on the d-dimensional Euclidean space $\mathbb{R}^d$. A natural "perturbation" of the current configuration $\mathbf{x}^{(t)}$ is the addition of a random "error;" that is, the next candidate position is proposed as $\mathbf{x}' = \mathbf{x}^{(t)} + \epsilon_t$, where $\epsilon_t \sim g_\sigma(\cdot)$ is independent and identically distributed for different t. Here, σ represents the "range" of the proposal exploration and is controlled by the user. In problems where we do not have much information on the shape of the target distribution, we often end up letting $g_\sigma(\cdot)$ be a spherically symmetric distribution. Typical choices include the spherical Gaussian distribution $N(0, \sigma^2 I)$ or the uniform distribution in a ball of radius σ. Clearly, if one does not exercise the Metropolis rejection rule to this proposal, the resulting walk will drift away to infinity and never come back (when $d \geq 3$). It is thus worthwhile to point out that in a Metropolis algorithm, the proposal chain is not required to have any good "global properties" other than being irreducible

[4] A Markov chain is said to be *irreducible* if the chain has nonzero probability (density) to move from one position in the state space to any other position in a finite number of steps.

[5] A Markov chain is said *aperiodic* if the maximum common divider of the number of steps it takes for the chain to come back to the starting point (any) is equal to one.

(see footnote 4 on page 114). But we do require some local properties for $T(\mathbf{x},\mathbf{y})$ [e.g., $T(\mathbf{x},\mathbf{y}) > 0$ whenever $T(\mathbf{y},\mathbf{x}) > 0$].

Given the current state $\mathbf{x}^{(t)}$, the random-walk Metropolis algorithm iterates the following steps:

- Draw $\epsilon \sim g_\sigma$ and set $\mathbf{x}' = \mathbf{x}^{(t)} + \epsilon$, where g_σ is a spherically symmetric distribution and σ can be controlled by the user.
- Simulate $u \sim$ Uniform[0,1] and update

$$\mathbf{x}^{(t+1)} = \begin{cases} \mathbf{y} & \text{if } u \leq \dfrac{\pi(\mathbf{x}')}{\pi(\mathbf{x}^{(t)})} \\ \mathbf{x}^{(t)} & \text{otherwise.} \end{cases}$$

In the example of simulating six hard-shell balls in a box, we used the random-walk method for each individual ball. In an interesting study, Gelman, Roberts and Gilks (1995) suggested that a rule of thumb in choosing σ in a random-walk Metropolis is to maintain a 25% to 35% acceptance rate. This rule is supported by a theoretical analysis for a Gaussian target density (more details in Section 5.8).

5.4.2 *Metropolized independence sampler*

A very special choice of the proposal transition function $T(\mathbf{x},\mathbf{y})$ is an *independent* trial density $g(\mathbf{y})$; that is, the proposed move $\mathbf{y}$ is generated from $g(\cdot)$ independent of the previous state $\mathbf{x}^{(t)}$. This method, as first suggested in Hastings (1970), appears to be an alternative to the rejection sampling and importance sampling. Its convergence properties was studied in Liu (1996a), where all the eigenvalues and eigenfunctions of the actual transition function are derived (see Section 13.4).

The MIS Scheme: given the current state $\mathbf{x}^{(t)}$,

- Draw $\mathbf{y} \sim g(\mathbf{y})$.
- Simulate $u \sim$ Uniform[0,1] and let

$$\mathbf{x}^{(t+1)} = \begin{cases} \mathbf{y} & \text{if } u \leq \min\left\{1, \dfrac{w(\mathbf{y})}{w(\mathbf{x}^{(t)})}\right\} \\ x^{(t)} & \text{otherwise,} \end{cases}$$

where $w(\mathbf{x}) = \pi(\mathbf{x})/g(\mathbf{x})$ is the usual *importance sampling weight.*

As with the rejection method, the efficiency of MIS depends on how close the trial density $g(\mathbf{y})$ is to the target $\pi(\mathbf{y})$. To ensure robust performance, it is advisable to let $g(\cdot)$ be a relatively long-tailed distribution. Gelman and Rubin (1992) and Tierney (1994) suggested that one can insert a couple of

MIS steps into Gibbs iteration when correctly sampling from a conditional distribution is difficult. The idea is useful in many Bayesian computations in which each conditional density can be approximated reasonably well by a Gaussian distribution. To accommodate irregular tail behaviors, it is essential to use a long-tailed t-distribution as $g(\mathbf{x})$.

5.4.3 *Configurational bias Monte Carlo*

The configurational bias Monte Carlo (CBMC) algorithm can be viewed as an SIS-based Metropolized independence sampler. Suppose the argument of the target distribution, $\mathbf{x}$, can be decomposed as $\mathbf{x} = (x_1, \ldots, x_d)$. As in a sequential importance sampler (Section 2.6.3 and Chapters 3 and 4), we assume that there is a sequence of *auxiliary distributions*

$$\pi_1(x_1), \pi_2(x_1, x_2), \ldots, \pi_{d-1}(\mathbf{x}_{d-1}), \pi(\mathbf{x})$$

that can help us construct the trial sampling distribution.

Let the trial sampling distribution of $\mathbf{x}$ be

$$g(\mathbf{x}) = g_1(x_1)g_2(x_2 \mid x_1) \cdots g_d(x_d \mid \mathbf{x}_{d-1}).$$

To implement a CBMC algorithm (Siepmann and Frenkel 1992), we first draw $\mathbf{x}^{(0)}$ from $g(\mathbf{x})$ via a SIS strategy and compute its importance weight $w^{(0)}$ (up to a normalizing constant). Suppose that currently we have $\mathbf{x}^{(t)}$ with weight $w(\mathbf{x}^{(t)})$; then, at the next iteration, we do the following:

- Independently generate a trial configuration $\mathbf{y}$ from $g(\ )$ (using the sequential approach); compute its importance weight

  $$w(\mathbf{y}) = \pi(\mathbf{y})/g(\mathbf{y}),$$

  which can often be derived recursively as

  $$w(\mathbf{y}) = \frac{\pi_1(y_1)}{g_1(y_1)} \frac{\pi_2(y_1, y_2)}{g_2(y_2 \mid y_1)\pi_1(y_1)} \cdots \frac{\pi_d(y_1, \ldots, y_d)}{g_d(y_d \mid \mathbf{y}_{d-1})\pi_{d-1}(\mathbf{y}_{d-1})}.$$

  (This recursion is key to the SIS approach.)

- Accept $\mathbf{y}$; that is, let $\mathbf{x}^{(t+1)} = \mathbf{y}$, with probability $\min\left\{1, \frac{w(\mathbf{y})}{w(\mathbf{x}^{(t)})}\right\}$; and let $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)}$ otherwise.

This procedure is exactly the same as the *Metropolized independence sampler* described in the previous subsection, except that the trial density is built up sequentially and the importance weight computed recursively.

A useful modification of the foregoing CBMC procedure is to incorporate a stage-wise rejection decision. Suppose all the previous $k-1$ steps in the sequential simulation of the trial configuration $\mathbf{x}'$ have been accepted. Then, at the kth stage of SIS, we accept $\mathbf{x}'_k = (x'_1, \ldots, x'_k)$ with probability

$$p_k = \min\left\{1, \frac{\pi_k(\mathbf{x}'_k)\pi_{k-1}(\mathbf{x}_{k-1})g_k(x_k \mid \mathbf{x}_{k-1})}{\pi_k(\mathbf{x}_k)\pi_{k-1}(\mathbf{x}'_{k-1})g_k(x'_k \mid \mathbf{x}'_{k-1})}\right\} = \min\left\{1, \frac{u_k(\mathbf{x}'_k)}{u_k(\mathbf{x}_k)}\right\},$$

where u_k is the same as that in (3.10). In other words, the acceptance probability is equal to the ratio of the incremental importance weights between the trial and the current configurations. When rejected, we go back to the first stage to rebuild the whole configuration. It should be noted that one does not need to perform the acceptance-rejection decision at every stage and she/he has a complete control on when to conduct conduct the acceptance-rejection step.

This multistage method has been shown effective in simulating superfluid Helium 4 and other quantum mechanical systems (Ceperley 1995). Compared to the CBMC, this multistage sampler can force an early stop so as to save computing power. However, the chance of the final acceptance of a complete configuration in the multistage approach should be smaller than that in the CBMC because

$$\min(1, a_1) \times \cdots \times \min(1, a_d) \le \min(1,\ a_1 \times \cdots \times a_d).$$

To show that the multistage modification of CBMC is proper, we can write down its actual transition function and prove that it satisfies the detailed balance. Here, we provide only a proof for the case when $d = 2$ [i.e. $\mathbf{x} = (x_1, x_2)$]. The general proof is left to the reader. Suppose the auxiliary distributions when $d = 2$ are $\pi_1(x_1)$ and $\pi_2(\mathbf{x}) \equiv \pi(\mathbf{x})$. Suppose the current state is $\mathbf{x}$. Then, the probability of accepting a new configuration $\mathbf{x}' = (x'_1 x'_2) \neq \mathbf{x}$ is

$$\begin{aligned} P(\mathbf{x} \to \mathbf{x}') &= g_1(x'_1)g_2(x'_2|x'_1) \min\left\{1, \frac{\pi_1(x'_1)g_1(x_1)}{\pi_1(x_1)g_1(x'_1)}\right\} \\ &\quad \times \min\left\{1, \frac{\pi(x'_1, x'_2)\pi_1(x_1)g_2(x_2 \mid x_1)}{\pi(x_1, x_2)\pi_1(x'_1)g_1(x'_2 \mid x'_1)}\right\} \\ &= \pi(\mathbf{x}') \min\{g_1(x'_1)\pi_1(x_1), \pi_1(x'_1)g_1(x_1)\} \\ &\quad \times \min\left\{\frac{\pi_1(x'_1)g_1(x'_2|x'_1)}{\pi(\mathbf{x}')}, \frac{\pi_1(x_1)g_1(x_2|x_1)}{\pi(\mathbf{x})}\right\}. \end{aligned}$$

Hence, this transition function is indeed of the form $\pi(\mathbf{x}')\delta(\mathbf{x}, \mathbf{x}')$, where δ is a symmetric function. The detailed balance condition is thus satisfied (see the argument in Section 5.3).

5.5 Multipoint Metropolis Methods

In principle, the Metropolis sampling method discussed in Section 5.2 can be applied to almost any target distribution. In practice, however, it is

not infrequent to discover that finding a good proposal transition kernel is rather difficult. Although the important generalization of Hastings (1970) enables one to use asymmetric proposal functions, a simple random-walk-type proposal is still most frequently seen in practice simply because there is no obviously advantageous alternative available. It is then often the case that a small step-size in the proposal transition (for the algorithms similar to those described in Section 5.4.1) will result in exceedingly slow movement of the corresponding Markov chain, whereas a large step-size will result in very low acceptance rate. In both cases, the mixing rate of the algorithm would be very slow.

Here, we describe a generalization of the Metropolis-Hastings's transition rule. This new rule (Frenkel and Smit 1996, Liu, Liang and Wong 2000, Qin and Liu 2000) enables a MCMC sampler to make large step-size jumps without lowering the acceptance rate.

5.5.1 *Multiple independent proposals*

Suppose $T(\mathbf{x}, \mathbf{y})$ is an arbitrary proposal transition function and $\delta(\mathbf{x}, \mathbf{y})$ is an arbitrary symmetric and non-negative function. A modest requirement is that $T(\mathbf{x}, \mathbf{y}) > 0$ if and only if $T(\mathbf{y}, \mathbf{x}) > 0$. Define

$$w(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{x}) T(\mathbf{x}, \mathbf{y}) \lambda(\mathbf{x}, \mathbf{y}), \tag{5.8}$$

where $\lambda(\mathbf{x}, \mathbf{y})$ is a non-negative symmetric function in $\mathbf{x}$ and $\mathbf{y}$ that can be chosen by the user. The only requirement is that $\lambda(\mathbf{x}, \mathbf{y}) > 0$ whenever $T(\mathbf{x}, \mathbf{y}) > 0$. We present a few choices of $\lambda(\mathbf{x}, \mathbf{y})$ in the latter part of this section. Suppose the *current state* is $\mathbf{x}^{(t)} = \mathbf{x}$; then, a MTM transition is defined as follows:

Multiple-Try Metropolis (MTM)

- Draw k independent trial proposals, $\mathbf{y}_1, \ldots, \mathbf{y}_k$, from $T(\mathbf{x}, \cdot)$. Compute $w(\mathbf{y}_j, \mathbf{x})$ as in (5.8) for $j = 1, \ldots, k$.
- Select $\mathbf{y}$ among the trial set $\{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$ with probability proportional to $w(\mathbf{y}_j, \mathbf{x})$, $j = 1, \ldots, k$. Then, produce a "reference set" by drawing $\mathbf{x}_1^*, \ldots, \mathbf{x}_{k-1}^*$ from the distribution $T(\mathbf{y}, \cdot)$. Let $\mathbf{x}_k^* = \mathbf{x}$.
- Accept $\mathbf{y}$ with probability

$$r_g = \min\left\{1, \frac{w(\mathbf{y}_1, \mathbf{x}) + \cdots + w(\mathbf{y}_k, \mathbf{x})}{w(\mathbf{x}_1^*, \mathbf{y}) + \cdots + w(\mathbf{x}_k^*, \mathbf{y})}\right\} \tag{5.9}$$

  and reject it with probability $1 - r_g$. The quantity r_g is called the *generalized M-H ratio.*

When $T(\mathbf{x}, \mathbf{y})$ is symmetric, for example, one can choose $\lambda(\mathbf{x}, \mathbf{y}) = T^{-1}(\mathbf{x}, \mathbf{y})$. Then, $w(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{x})$. In this case, the MTM algorithm is simplified as the following algorithm, known as *orientational bias* Monte Carlo (OBMC) in the field of molecular simulation.

OBMC Algorithm

- Draw k trials $\mathbf{y}_1, \ldots, \mathbf{y}_k$ from a *symmetric* proposal function $T(\mathbf{x}, \mathbf{y})$.
- Select $\boldsymbol{Y} = \mathbf{y}_l$ among the $\mathbf{y}$'s with probability proportional to $\pi(\mathbf{y}_j)$, $j = 1, \ldots, k$; then, draw the reference points $\mathbf{x}_1', \ldots, \mathbf{x}_{k-1}'$ from the distribution $T(\mathbf{y}_l, \mathbf{x}')$. Let $\mathbf{x}_k' = \mathbf{x}$.
- Accept $\mathbf{y}_l$ with probability

$$\min\left\{1, \frac{\pi(\mathbf{y}_1) + \cdots + \pi(\mathbf{y}_k)}{\pi(\mathbf{x}_1') + \cdots + \pi(\mathbf{x}_k')}\right\}$$

  and reject with the remaining probability.

The proof of the correctness of this method is straightforward (Liu, Liang and Wong 2000). Roughly speaking, one can directly check the detailed balance by writing down what the algorithmic instructions mean mathematically. To illustrate the idea, we prove the case for $k = 2$.

Proof: Let $A(\mathbf{x}, \mathbf{y})$ be the actual transition probability for moving from $\mathbf{x}$ to $\mathbf{y}$ in a MTM sampler. Suppose $\mathbf{x} \neq \mathbf{y}$ and let I indicate which of $\mathbf{y}_j$ has been selected. Since $w(\mathbf{y}, \mathbf{x}) = \pi(\mathbf{y}) T(\mathbf{y}, \mathbf{x}) \lambda(\mathbf{y}, \mathbf{x})$ and the $\mathbf{y}_j$ are exchangeable, we have

$$\begin{aligned}
\pi(\mathbf{x}) A(\mathbf{x}, \mathbf{y}) &= 2\, \pi(\mathbf{x}) P[(\boldsymbol{Y}_1 = \mathbf{y}) \cap (I = 1) \mid \mathbf{x}] \qquad \text{(symmetry)} \\
&= 2\, \pi(\mathbf{x}) \int T(\mathbf{x}, \mathbf{y}) T(\mathbf{x}, \mathbf{y}_2) \frac{w(\mathbf{y}, \mathbf{x})}{w(\mathbf{y}, \mathbf{x}) + w(\mathbf{y}_2, \mathbf{x})} \\
&\quad \times \min\left\{1, \frac{w(\mathbf{y}, \mathbf{x}) + w(\mathbf{y}_2, \mathbf{x})}{w(\mathbf{x}, \mathbf{y}) + w(\mathbf{x}_2^*, \mathbf{y})}\right\} T(\mathbf{y}, \mathbf{x}_2^*) d\mathbf{y}_2 d\mathbf{x}_2^* \\
&= 2\, \frac{w(\mathbf{x}, \mathbf{y}) w(\mathbf{y}, \mathbf{x})}{\lambda(\mathbf{y}, \mathbf{x})} \int T(\mathbf{x}, \mathbf{y}_2) T(\mathbf{y}, \mathbf{x}_2^*) \\
&\quad \times \min\left\{\frac{1}{w(\mathbf{y}, \mathbf{x}) + w(\mathbf{y}_2, \mathbf{x})}, \frac{1}{w(\mathbf{x}, \mathbf{y}) + w(\mathbf{x}_2^*, \mathbf{y})}\right\} d\mathbf{y}_2 d\mathbf{x}_2^*.
\end{aligned}$$

The final expression is symmetric in $\mathbf{x}$ and $\mathbf{y}$ because $\lambda(\mathbf{x}, \mathbf{y}) = \lambda(\mathbf{y}, \mathbf{x})$. Thus, we proved that $\pi(\mathbf{x}) A(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{y}) A(\mathbf{y}, \mathbf{x})$, which is the detailed balance condition. $\diamond$

Another interesting application is to combine the MTM approach with the Metropolized independence sampler (Section 5.4.2). Because the trial

samples are generated independently, one does not need to generate another "reference set." More precisely, suppose the current state is $\mathbf{x}^{(t)} = \mathbf{x}$ in our MCMC iteration; then, the next state can be generated by the following *multiple-trial Metropolized independence sampler* (MTMIS):

MTMIS:

- Generate a trial set of i.i.d. samples by drawing $\mathbf{y}_j \sim p(\mathbf{y})$, $j = 1, \ldots, k$, independently, where $p(\ )$ is a trial distribution chosen by the user. Compute $w(\mathbf{y}_j) = \pi(\mathbf{y}_j)/p(\mathbf{y}_j)$ and $W = \sum_{j=1}^{k} w(\mathbf{y}_j)$.
- Draw $\mathbf{y}$ from the trial set $\{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$ with probability proportional to $w(\mathbf{y}_j)$.
- Let $\mathbf{x}^{(t+1)} = \mathbf{y}$ with probability

$$\min\left\{1, \frac{W}{W - w(\mathbf{y}) + w(\mathbf{x})}\right\}$$

  and let $\mathbf{x}^{(t+1)} = \mathbf{x}$ with the remaining probability.

The idea of using MTM to make large-step moves along certain favorable directions is a useful heuristic and can be applied broadly. We will show later (Chapter 11) how the MTM can be applied to improve the algorithm's performance in more complicated settings.

5.5.2 *Correlated multipoint proposals*

Based on the work of OBMC and MTM, Qin and Liu (2000) provide a more general scheme, termed as the *multipoint* method, which allows one to choose from multiple *correlated* proposals at each iteration. Its application in hybrid Monte Carlo has shown promising results. Suppose the current state is $\mathbf{x}^{(t)} = \mathbf{x}$. We generate k trial proposals as follows: Let $\mathbf{y}_1 \sim P_1(\cdot \mid \mathbf{x})$ and let

$$\mathbf{y}_j \sim P_j(\cdot \mid \mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_{j-1}), \quad j = 2, \ldots, k.$$

For brevity, we also let $\mathbf{y}_{[1:j]} = (\mathbf{y}_1, \ldots, \mathbf{y}_j)$, $\mathbf{y}_{[j:1]} = (\mathbf{y}_j, \ldots, \mathbf{y}_1)$ and let

$$P_j(\mathbf{y}_{[1:j]} \mid \mathbf{x}) = P_1(\mathbf{y}_1 \mid \mathbf{x}) \cdots P_j(\mathbf{y}_j \mid \mathbf{x}, \mathbf{y}_{[1:j-1]}).$$

A weight function is defined as

$$w_j(\mathbf{x}, \mathbf{y}_{[1:j]}) = \pi(\mathbf{x}) P_j(\mathbf{y}_{[1:j]} \mid \mathbf{x}) \lambda_j(\mathbf{x}, \mathbf{y}_{[1:j]}), \tag{5.10}$$

where $\lambda_j(\ )$ is a *sequentially symmetric* function; that is,

$$\lambda_j(a, b, \ldots, z) = \lambda_j(z, \ldots, b, a).$$

The general algorithm is as follows:

Multipoint Method:

- Sample $\mathbf{y}$ from the trial set $\{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$ with probability proportional to $w(\mathbf{y}_{[l:1]}, \mathbf{x})$; suppose $\mathbf{y}_j$ is chosen.
- Create a reference set by letting $\mathbf{x}_l^* = \mathbf{y}_{j-l}$ for $l = 1, \ldots, j-1$, $\mathbf{x}_j^* = \mathbf{x}$, and drawing

$$\mathbf{x}_m^* \sim P_m(\cdot \mid \mathbf{y}, \mathbf{x}_{[1:m-1]}^*),$$

  for $m = j+1, \ldots, k$.
- Let $\mathbf{x}^{(t+1)} = \mathbf{y}$ with probability

$$r_{mp} = \min\left\{1, \frac{\sum_{l=1}^{k} w(\mathbf{y}_{[l:1]}, \mathbf{x})}{\sum_{l=1}^{k} w(\mathbf{x}_{[l:1]}^*, \mathbf{y})}\right\},$$

  and let $\mathbf{x}^{(t+1)} = \mathbf{x}$ with the remaining probability.

The simplest choice of $\lambda_j(\ )$ for (5.10) is the constant function. But we may, in some cases, want to give larger weights to larger j's since these points are "farther" away from the initial point $\mathbf{x}$. When P_j is constructed by composing a symmetric transition kernel j times, the resulting function is *sequentially symmetric*. Thus, we can choose λ_j as v_j/P_j, where v_j is a constant, so that $w_j(\mathbf{y}_{[j:1]}, \mathbf{x}) = v_j \pi(\mathbf{y}_j)$. The resulting algorithm is very similar to OBMC.

When the state space is $\mathbb{R}^d$, we can create a random-grid Monte Carlo algorithm similar to the random-ray Monte Carlo method (Liu, Liang and Wong 2000) to be described in Section 6.3.3. At each iteration, we do the following steps.

Random-Grid Method:

- Randomly generate a direction $\boldsymbol{e}$ and a grid size r.
- Construct the candidate set as

$$\mathbf{y}_l = \mathbf{x} + l \cdot r \cdot \vec{\mathbf{e}}, \quad \text{for } l = 1, \ldots, k.$$

- Draw $\mathbf{y} = \mathbf{y}_j$ from $\{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$ with probability proportional to $u_j \pi(\mathbf{y}_j)$, where u_j is a constant chosen by the user (e.g., $u_j = \sqrt{j}$).
- Construct the reference set by letting $\mathbf{x}_l^* = \mathbf{y} - l \cdot r \cdot \boldsymbol{e}$, for $l = 1, \ldots k$. Therefore, $\mathbf{x}_l^* = \mathbf{y}_{j-l}$ for $l < j$ and $\mathbf{x}_l^* = \mathbf{x} - (l-j) \cdot r \cdot \vec{\mathbf{e}}$ for $l \geq j$.
- Accept the candidate $\mathbf{y}$ with probability

$$p = \min\left\{1, \sum_{l=1}^{k} \pi(\mathbf{y}_l) / \sum_{l=1}^{k} \pi(\mathbf{x}_l^*)\right\},$$

  and reject otherwise.

5.6 Reversible Jumping Rule

In applications such as image analysis (Grenander and Miller 1994) and Bayesian model selections (Green 1995), one often needs to design a sampler that jumps between different dimensional spaces. In principle, one still can follow the Metropolis-Hastings's rule to guide for the design of such a sampler. The only technical complication is in ensuring the reversibility of proposals for jumping between two different dimensional spaces.

Suppose $\mathcal{X}$ is the state space of interest and $\mathcal{Y}$ is a subspace of $\mathcal{X}$ with a lower dimensionality. For example, $\mathcal{Y}$ can be a manifold defined as $\mathcal{Y} = \{\mathbf{x} : f(\mathbf{x}) = 0\}$ for some differentiable function f. Furthermore, we suppose that the target distribution $\pi(\mathbf{x})$, known up to a normalizing constant, lives on these two spaces *simultaneously*. This distribution can be represented as

$$\pi(\mathbf{x}) \propto q_0(\mathbf{x})|_{\mathbf{x}\in\mathcal{Y}} + q_1(\mathbf{x}), \tag{5.11}$$

where q_0 and q_1 are two unnormalized probability density functions defined on their respective spaces [i.e., $q_i(\mathbf{x}) = c_i\pi_i(\mathbf{x})$ with c_i unknown]. Therefore, if we draw a random sample from $\mathcal{X}$ according to $\pi_1(\mathbf{x})$, the chance that it lies in $\mathcal{Y}$ is zero! To make things worse, we assume that the ratio of the two constants, $\gamma = c_0/c_1$, is generally unknown to us. If we can design a Monte Carlo algorithm to sample from $\pi(\mathbf{x})$, the ratio γ can be estimated by the ratio of the number of samples lying in $\mathcal{Y}$ over that lying in $\mathcal{X}$.

The above setting is of particular interest in Bayesian hypothesis testing problems. Suppose we have a probability model $f(\mathbf{y} \mid \boldsymbol{\theta})$ where $\boldsymbol{\theta} = (\theta_0, \theta_1)$ and we are interested in testing H_0: $\theta_0 = \theta_1$ versus H_1: $\theta_0 \neq \theta_1$ in light of an observation $\mathbf{y}$. We let model M_1 correspond to the unrestricted parameter space $\boldsymbol{\theta}$ (two dimensional) and let model M_0 correspond to the subspace defined by $\theta_0 - \theta_1 = 0$. It is sometimes natural assume that the two models are equally likely *a priori*; then, the posterior distribution of $\boldsymbol{\theta}$ under the "mixture" of two plausible models is

$$\pi(\boldsymbol{\theta}) \propto f(\mathbf{y} \mid \boldsymbol{\theta}) f_0(\boldsymbol{\theta})|_{\theta_0=\theta_1} + f(\mathbf{y} \mid \boldsymbol{\theta}) f_0(\boldsymbol{\theta})|_{\theta_0\neq\theta_1}.$$

Statisticians are often interested in estimating the ratio of the two normalizing constants,

$$\frac{c_0}{c_1} \equiv \frac{\int_{\theta_0=\theta_1} f(\mathbf{y} \mid \boldsymbol{\theta}) f_0(\boldsymbol{\theta}) d\boldsymbol{\theta}}{\int f(\mathbf{y} \mid \boldsymbol{\theta}) f_1(\boldsymbol{\theta}) d\boldsymbol{\theta}},$$

which reflects the posterior odds ratio of model M_0 versus model M_1. As we mentioned in the previous paragraph, this ratio can be estimated if we can design a Monte Carlo scheme to sample π.

In order to design a Monte Carlo Markov chain that lives on both the general state space and the restricted space, we need to have two *different* proposals, one for $\mathcal{Y} \to \mathcal{X}$ and another for $\mathcal{X} \to \mathcal{Y}$. Since $\mathcal{Y}$ is of lower

dimensional, any transition from $\mathcal{Y}$ to $\mathcal{X}$ must have a degenerate density with respect to the dominant measure on $\mathcal{X}$, implying that no proposal for moves from $\mathcal{X} \to \mathcal{Y}$ can be properly "reversed" by a proposal from $\mathcal{Y}$ to $\mathcal{X}$. To overcome this difficulty, we must have a "matching space" $\mathcal{Z}$, so that $\mathcal{Y} \times \mathcal{Z}$ has the same dimension as $\mathcal{X}$ and a matching proposal $g(\mathbf{z} \mid \mathbf{y})$. With the matched space, one can come up with two non-degenerate proposals and follow the Metropolis-Hastings's rule to design jumps.

Green (1995) presented a formal treatment of this type of moves (involving change-of-variables and Jacobians) and named them *reversible jumps*. Here, we study only a sufficiently instructive special case: $\mathcal{X} = \mathcal{Y} \times \mathcal{Z}$. Therefore, each $\mathbf{x}$ can be written as $\mathbf{x} = (\mathbf{y}, \mathbf{z})$, and "subspace" $\mathcal{Y}$ in fact corresponds to $\mathcal{Y} \times \{\mathbf{z}_0\}$ for some $\mathbf{z}_0 \in \mathcal{Z}$. In order to jump from $\mathcal{Y}$ to $\mathcal{X}$, we may first propose $\mathbf{y} \to \mathbf{y}'$ by a proposal transition $T_1(\mathbf{y}, \mathbf{y}')$, match $\mathbf{y}'$ with a $\mathbf{z}'$ drawn from $g(\cdot \mid \mathbf{y}')$, and then let $\mathbf{x}' = (\mathbf{y}', \mathbf{z}')$. This can be viewed as an *expansion* transition. A *contraction* transition is needed to propose from $\mathbf{x} \in \mathcal{X}$ back into $\mathcal{Y}$. This can be achieved by first dropping the $\mathbf{z}$ component in $\mathbf{x}$, and then proposing $\mathbf{y}'$ from $T_2(\mathbf{y}, \mathbf{y}')$. According the Metropolis-Hastings rule, the expansion proposal $\mathbf{y} \to \mathbf{x}'$ is accepted with probability

$$\alpha = \min\left\{1, \frac{q_1(\mathbf{y}', \mathbf{z}')T_2(\mathbf{y}', \mathbf{y})}{q_0(\mathbf{y})T_1(\mathbf{y}, \mathbf{y}')g(\mathbf{z}' \mid \mathbf{y}')}\right\},$$

where q_0 and q_1 are as defined in (5.11). The contraction proposal $\mathbf{x} \to \mathbf{y}'$ (where $\mathbf{x} = (\mathbf{y}, \mathbf{z})$) is accepted with probability

$$\beta = \min\left\{1, \frac{q_0(\mathbf{y}')T_1(\mathbf{y}', \mathbf{y})g(\mathbf{z} \mid \mathbf{y})}{q_1(\mathbf{y}, \mathbf{z})T_2(\mathbf{y}, \mathbf{y}')}\right\}.$$

The expansion proposal in the foregoing procedure can be seen as first "proposing" and then "lifting" (from a lower dimensional space to the higher one). Similarly, we can conduct "lifting" first and "proposing" afterward. More precisely, in order to accomplish the proposal $\mathbf{y} \to \mathbf{x}'$, we can first draw $\mathbf{z} \sim g(\cdot \mid \mathbf{y})$ and then draw $\mathbf{x}'$ from $S_1[(\mathbf{y}, \mathbf{z}), \cdot]$. The contraction move is achieved by first proposing $\mathbf{x} \to \mathbf{x}' = (\mathbf{y}', \mathbf{z}')$ according to $S_2(\mathbf{x}, \mathbf{x}')$ and then dropping $\mathbf{z}'$. Thus, the acceptance probabilities are respectively

$$\alpha' = \min\left\{1, \frac{q_1(\mathbf{x}')S_2[\mathbf{x}', (\mathbf{y}, \mathbf{z})]}{q_0(\mathbf{y})g(\mathbf{z} \mid \mathbf{y})S_1[(\mathbf{y}, \mathbf{z}), \mathbf{x}']}\right\},$$

$$\beta' = \min\left\{1, \frac{q_0(\mathbf{y}')g(\mathbf{z}' \mid \mathbf{y}')S_1[(\mathbf{y}', \mathbf{z}'), \mathbf{x}]}{q_1(\mathbf{x})S_2[\mathbf{x}, (\mathbf{y}', \mathbf{z}')]}\right\}.$$

Note that both S_1 and S_2 are proposals in the higher-dimensional space $\mathcal{X}$, whereas both T_1 and T_2 are proposals in the lower-dimensional subspace $\mathcal{Y}$. Thus, without having other justifications, we prefer lifting *after*

proposing than the other way around because it is often easier to propose lower-dimensional moves. It is conceivable, however, that lifting *before* proposing can sometimes help the chain escape from a local energy trap of the lower-dimensional space. Similar ideas have been employed in the clustering method and simulated tempering (Chapters 7 and 10).

A useful strategy in improving Monte Carlo sampling efficiency is to introduce a number of related probabilistic systems with different levels of "difficulties" (in terms of Monte Carlo sampling) and then simulate them together. These auxiliary systems are often made by varying a "temperature" parameter in the original target distribution for the sake of easy manipulation (i.e., simulated tempering and parallel tempering; see Chapter 10). However, it may be more efficient to consider a system consisting of spaces with different dimensions (Liu and Sabatti 1998). To sample from this augmented system, one needs to use the reversible jumping rule. It should be noted that the basic principle behind the reversible jumps is similar to that behind the sequential importance sampling and the CBMC (Sections 2.6.3 and 5.4.3) because the move from $\mathbf{y}$ to $\mathbf{x}$ can be seen as a one-step SIS update.

5.7 Dynamic Weighting

Wong and Liang (1997) introduced the use of a dynamic weighting variable for controlling Markov chain simulation. By using this scheme, they were able to obtain better results for many optimization problems, such as the traveling salesman problem and neural network training, and high-dimensional integration problems, such as the Ising model simulation.

To start a dynamic weighting scheme, we first augment the sample space $\mathcal{X}$ to $\mathcal{X} \times \mathbb{R}^+$ so as to include a weight variable. Similar to the Metropolis algorithm, we also need a proposal function $T(\mathbf{x}, \mathbf{y})$ on the space $\mathcal{X}$. Suppose at iteration t, we have $(\mathbf{x}^{(t)}, w^{(t)}) = (\mathbf{x}, w)$. Then an *R-type move* is defined as follows:

- Draw $\mathbf{y}$ from $T(\mathbf{x}, \mathbf{y})$ and compute the Metropolis-Hastings ratio

$$r(\mathbf{x}, \mathbf{y}) = \frac{\pi(\mathbf{y})T(\mathbf{y}, \mathbf{x})}{\pi(\mathbf{x})T(\mathbf{x}, \mathbf{y})}.$$

- Choose $\theta = \theta(w, \mathbf{x}) > 0$, and draw U from Uniform(0,1). Then let

$$(\mathbf{x}^{(t+1)}, w^{(t+1)}) = \begin{cases} (\mathbf{y}, wr(\mathbf{x}, \mathbf{y}) + \theta) & \text{if } U \le \dfrac{wr(\mathbf{x}, \mathbf{y})}{wr(\mathbf{x}, \mathbf{y}) + \theta} \\ \left(\mathbf{x}^{(t)}, \dfrac{w(wr(\mathbf{x}, \mathbf{y}) + \theta)}{\theta}\right) & \text{otherwise.} \end{cases} \tag{5.12}$$

It is easy to check that the R-type move does not have π as its equilibrium distribution. Wong and Liang (1997) propose to use *invariance with respect to importance weighting* (IWIW) for justifying the above scheme; that is, if the joint distribution of $(\mathbf{x}, w)$ is $f(\mathbf{x}, w)$ and $\mathbf{x}$ is said *correctly weighted* by w with respect to π if $\sum_w wf(\mathbf{x}, w) \propto \pi(\mathbf{x})$. A transition rule is said to satisfy IWIW if it *maintains* the *correctly weightedness* for the joint distribution of $(\mathbf{x}, w)$. Clearly, the R-type move satisfies IWIW.

The purpose of introducing importance weights into the dynamic Monte Carlo process is to provide a means for the system to make large transitions not allowable by the standard Metropolis transition rules. The weight variable is updated in a way that allows for an adjustment of the bias induced by such non-Metropolis moves. Although this algorithm has been applied successfully in many difficult optimization and simulation problems [see Section 10.6 and Liang (1997)], theoretical properties of this algorithm are still rather subtle. A first theory is recently given by Liu, Liang and Wong (2001) and some of which, together with another type of dynamic weighting scheme, the *Q-type* move, will be presented in Section 13.6. An important application of the dynamic weighting method is to be combined with a *simulated tempering* algorithm, and this aspect will be discussed in more detail in Section 10.6.

5.8 Output Analysis and Algorithm Efficiency

In analyzing outputs from a Markov chain Monte Carlo algorithm (this applies to all the later chapters), one of the major concerns is its *statistical efficiency* in estimating the expectation of interest. Let us suppose that the Markov chain is irreducible and aperiodic (see footnotes 4 and 5 on page 114) and converges to its unique stationary distribution, $\pi(\mathbf{x})$. At the heart of every MCMC computation is the estimation of $E_\pi h(\mathbf{x})$ for a certain $h(\ )$ of interest. Thus, what we really care about at the end is how accurate we can estimate this quantity. Suppose we have drawn samples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}$ via a MCMC sampler with $\pi(\mathbf{x})$ as its equilibrium distribution. Let us further assume that we have run the process long enough (in the previous day, say) and have thrown away the initial n_0 iterations needed for the equilibration of the chain (i.e., we assume that $\mathbf{x}^{(0)} \sim \pi$). Then,

$$\begin{aligned} m\text{var}\left\{\frac{h(\mathbf{x}^{(1)}) + \cdots + h(\mathbf{x}^{(m)})}{m}\right\} &= \sigma^2\left[1 + 2\sum_{j=1}^{m-1}\left(1 - \frac{j}{m}\right)\rho_j\right] \\ &\approx \sigma^2\left[1 + 2\sum_{j=1}^{\infty}\rho_j\right] \end{aligned} \tag{5.13}$$

where $\sigma^2 = \text{var}[h(\mathbf{x})]$ and $\rho_j = \text{corr}\{h(\mathbf{x}^{(1)}), h(\mathbf{x}^{(j+1)})\}$. In physics literature (Goodman and Sokal 1989), one defines the *integrated autocorrelation time* of $h(\mathbf{x})$ as

$$\tau_{\text{int}}(h) = \frac{1}{2} + \sum_{j=1}^{\infty} \rho_j.$$

Then we have

$$m\text{var}(\hat{h}) = 2\tau_{\text{int}}(h)\sigma^2.$$

This variance is, in effect, equal to that of an estimator with $m/[2\tau_{\text{int}}(h)]$ independent random samples. Thus, $m/[2\tau_{\text{int}}(h)]$ is often called the *effective sample size.*

If is often observed that ρ_j decays exponentially. Therefore, we can model the autocorrelation curve as

$$|\rho_j| \sim \exp\left\{-\frac{j}{\tau_{\text{exp}}(h)}\right\},$$

which gives rise to the expression

$$\tau_{\text{exp}}(h) = \limsup_{j\to\infty} \frac{j}{-\log|\rho_j|},$$

where $\tau_{\text{exp}}(h)$ is called the *exponential autocorrelation time.* When $\tau_{\text{exp}}(h)$ is large, we can see that

$$\tau_{\text{int}}(h) \approx \sum_{j=0}^{\infty} e^{-j/\tau_{\text{exp}}(h)} - \frac{1}{2} = \frac{1}{1 - e{-}1/\tau_{\text{exp}}(h)} - \frac{1}{2} \approx \tau_{\text{exp}}(h).$$

The "relaxation time" of the system is defined as

$$\tau_{\text{exp}} = \sup_{h\in L^2(\pi)} \tau_{\text{exp}}(h).$$

The concepts of the autocorrelation and relaxation time are also closely related to the convergence rate of the algorithm (i.e., the second largest eigenvalue of the Markov chain transition matrix). More precisely, if we let h be an eigenfunction that corresponds to an eigenvalue λ of the transition matrix, then we have $\rho_j(h) = \lambda^j$. Hence,

$$\tau_{\text{int}}(h) = \frac{1+\lambda}{2(1-\lambda)}, \quad \tau_{\text{exp}}(h) = -\frac{1}{\log|\lambda|},$$

and the relaxation time is

$$\tau_{\text{exp}} = -\frac{1}{\log|\lambda_2|},$$

where λ_2 is the second largest eigenvalue in modular of the transition matrix. Thus, τ_{exp} reflects the convergence speed of an MCMC sampler, whereas τ_{int} is most relevant when the statistical efficiency of the algorithm is of interest.

It is commonly agreed that finding an ideal proposal chain is an art. In fact, the Metropolis algorithm aided with Hastings's (1970) generalization is so general that one always tends to feel unsatisfactory in settling down on any specific proposal chain. It is important, therefore, to analyze autocorrelation curves of an algorithm in order to obtain its behavioral characteristics. Peskun (1973) suggests that two Markov chains, with transition functions P_1 and P_2, respectively, and the same equilibrium distribution, should be compared based on a statistical criterion — the asymptotic variance of the corresponding estimator (5.13). When the state space is finite, he found an explicit asymptotic formula for (5.13), from which he concluded that for a given proposal transition, the Metropolis acceptance-rejection rule is "optimal" in the sense of having the smallest asymptotic variance for the resulting estimates. Consequently, Barker's proposal is less desirable in general. An interesting twist of Peskun's result will be described more details in Section 5.4.2.

Another interesting result is given by Gelman, Roberts and Gilks (1995) for a continuous state space. Suppose the target distribution is $N(0,1)$ and a random-walk Metropolis algorithm (Section 5.4.1) is used for its simulation. The proposal transition is of the form $\mathbf{x}' = \mathbf{x}^{(t)} + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$. Gelman, Roberts and Gilks (1995) show that the optimal choice for σ that gives the smallest autocorrelation is $\sigma = 2.38$. A slightly larger σ does not affect the efficiency much, but a smaller one has a significant adverse effect. These assertions can be verified by direct simulation. This optimal σ corresponds to an acceptance rate of 44%, suggesting that this is a useful reference number to be watched when tuning a proposal distribution. Roberts, Gelman and Gilks (1997) recommended calibrating the acceptance rate to about 25% for a high-dimensional model and to about 50% for models of dimensions 1 or 2.

5.9 Problems

1. Implement a Metropolis algorithm to sample from a Poisson (λ) distribution. Test it for λ=3, 5, and 10. A suggestion for the proposal function: the simple random walk on a line.

2. Let $\pi(\mathbf{x})$ be $N(0,1)$. Implement the random-walk Metropolis algorithm to sample from π, using $N(\cdot, \sigma^2)$ as the proposal function. Plot the autocorrelation curve for the corresponding chain. Argue why $\sigma = 2.38$ is a good choice.

3. Prove that the multistage modification of the CBMC method is proper for all d (Section 5.4.3).

4. Show that the detailed balance condition $\pi(\mathbf{x})A(\mathbf{x},\mathbf{y}) = \pi(\mathbf{y})A(\mathbf{y},\mathbf{x})$ guarantees that π is the invariant distribution of $A(\mathbf{x},\mathbf{y})$.

5. Suppose the Metropolized independence sampler (MIS) is applied to sample from $\pi(\mathbf{x})$, where $\mathbf{x}$ is defined on a finite state space and the trial distribution is $g(\mathbf{x})$.

   (a) Write down the actual transition matrix A for the MIS.

   (b) Show that the second largest eigenvalue of A is $\min_{\mathbf{x}}\{\pi(\mathbf{x})/g(\mathbf{x})\}$.

   (c) Find its corresponding eigenvector.

6. Show that the random-grid method is proper [i.e., it leaves the target distribution $\pi(\mathbf{x})$ invariant]. Implement the random-grid method to sample from a multidimensional Gaussian distribution. Study empirically how the choices of k and the grid-size distribution affect algorithmic efficiency.

7. Show that the reversible jump algorithm as described in Section 5.6 leaves π invariant.

8. Show that the R-type dynamic weighting rule satisfies the IWIW property.

9. Show that the Q-type dynamic weighting rule does not satisfy the IWIW property.

10. Implement both the random-ray and the random-grid methods to replace the griddy-Gibbs method in Example 6.1 of Ritter and Tanner (1992).

11. Prove that the MTMIS algorithm gives rise to a reversible Markov chain whose equilibrium distribution is π.

6
The Gibbs Sampler

The proposal transition $T(\mathbf{x},\mathbf{y})$ in a Metropolis sampler is often an arbitrary choice out of convenience. In many applications, the proposal is chosen to be a locally uniform move. In fact, the use of symmetric and locally uniform proposals is so prevailing that these are often referred to as "unbiased proposals" in the literature. If not subjecting to the Metropolis-Hastings rejection rule, this type of move would have led to a form of simple random walk in the configuration space $\mathcal{X}$. Although such a proposal is simple both conceptually and operationally, the performance of the resulting Metropolis algorithm is often inefficient because the proposal is too "noisy." In contrast, the conditional sampling techniques to be discussed in this and the next chapters enable a MCMC sampler to follow the local dynamics of the target distribution. A distinctive feature of these MCMC algorithms is that at each iteration, they use conditional distributions (i.e., those distributions resulting from constraining the target distribution π on certain subspaces) to construct Markov chain moves. As a consequence, no rejection is incurred at any of its sampling steps. The multipoint methods described in Section 5.5 are similar in spirit but are computationally more expensive than conditional sampling.

6.1 Gibbs Sampling Algorithms

The Gibbs sampler (Geman and Geman 1984) is a special MCMC scheme. Its most prominent feature is that the underlying Markov chain is con-