17. J. P. C. Kleijnen. *Statistical Techniques in Simulation, Part 1*. Marcel Dekker, New York, 1974.

18. J. P. C. Kleijnen. Analysis of simulation with common random numbers: A note on Heikes et al. *Simuletter*, 11:7–13, 1976.

19. D. P. Kroese and R. Y. Rubinstein. The transform likelihood ratio method for rare event simulation with heavy tails. *Queueing Systems*, 46:317–351, 2004.

20. A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, New York, 3rd edition, 2000.

21. D. Lieber, A. Nemirovski, and R. Y. Rubinstein. A fast Monte Carlo method for evaluation of reliability indices. *IEEE Transactions on Reliability Systems*, 48(3):256–261, 1999.

22. D. Lieber, R. Y. Rubinstein, and D. Elmakis. Quick estimation of rare events in stochastic networks. *IEEE Transaction on Reliability*, 46:254–265, 1997.

23. J. S. Liu. *Monte Carlo Strategies in Scientifi c Computing*. Springer-Verlag, New York, 2001.

24. D. L. McLeish. *Monte Carlo Simulation and Finance*. John Wiley & Sons, New York, 2005.

25. M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. Dover Publications, New York, 1981.

26. C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, New York, 2nd edition, 2004.

27. S. M. Ross. *Simulation*. Academic Press, New York, 3rd edition, 2002.

28. R. Y. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley & Sons, New York, 1981.

29. R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unifi ed Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*. Springer-Verlag, New York, 2004.

30. R. Y. Rubinstein and R. Marcus. Efficiency of multivariate control variables in Monte Carlo simulation. *Operations Research*, 33:661–667, 1985.

31. R. Y. Rubinstein and B. Melamed. *Modern Simulation and Modeling*. John Wiley & Sons, New York, 1998.

32. R. Y. Rubinstein and A. Shapiro. *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization Via the Score Function Method*. John Wiley & Sons, New York, 1993.

33. R.Y. Rubinstein, M. Samorodnitsky, and M. Shaked. Antithetic variables, multivariate dependence and simulation of complex stochastic systems. *Management Science*, 31:66–77, 1985.

34. I. M. Sobol. *A Primer for the Monte Carlo Method*. CRC Press, Boca Raton, FL, 1994.

35. W. Whitt. Bivariate distributions with given marginals. *Annals of Statistics*, 4(6):1280–1289, 1976.

CHAPTER 6

MARKOV CHAIN MONTE CARLO

6.1 INTRODUCTION

In this chapter we present a powerful generic method, called *Markov chain Monte Carlo* (MCMC), for *approximately* generating samples from an arbitrary distribution. This, as we learned in Section 2.5, is typically not an easy task, in particular when $\mathbf{X}$ is a random vector with dependent components. An added advantage of MCMC is that it only requires specification of the target pdf up to a (normalization) constant.

The MCMC method is due to Metropolis et al. [17]. They were motivated by computational problems in statistical physics, and their approach uses the idea of generating a Markov chain whose limiting distribution is equal to the desired target distribution. There are many modifications and enhancement of the original Metropolis [17] algorithm, most notably the one by Hastings [10]. Nowadays, any approach that produces an ergodic Markov chain whose stationary distribution is the target distribution is referred to as MCMC or *Markov chain sampling* [19]. The most prominent MCMC algorithms are the Metropolis–Hastings and the Gibbs samplers, the latter being particularly useful in Bayesian analysis. Finally, MCMC sampling is the main ingredient in the popular *simulated annealing* technique [1] for discrete and continuous optimization.

The rest of this chapter is organized as follows. In Section 6.2 we present the classic Metropolis–Hastings algorithm, which simulates a Markov chain such that its stationary distribution coincides with the target distribution. An important special case is the *hit-and-run* sampler, discussed in Section 6.3. Section 6.4 deals with the *Gibbs sampler*, where the underlying Markov chain is constructed based on a sequence of conditional distributions.

Simulation and the Monte Carlo Method, Second Edition. By R.Y. Rubinstein and D. P. Kroese

Section 6.5 explains how to sample from distributions arising in the Ising and Potts models, which are extensively used in statistical mechanics, while Section 6.6 deals with applications of MCMC in Bayesian statistics. In Section 6.7 we show that both the Metropolis–Hastings and Gibbs samplers can be viewed as special cases of a general MCMC algorithm and present the slice and reversible jump samplers. Section 6.8 deals with the classic simulated annealing method for finding the global minimum of a multiextremal function, which is based on the MCMC method. Finally, Section 6.9 presents the perfect sampling method, for sampling exactly from a target distribution rather than approximately.

6.2 THE METROPOLIS–HASTINGS ALGORITHM

The main idea behind the Metropolis–Hastings algorithm is to simulate a Markov chain such that the stationary distribution of this chain coincides with the target distribution.

To motivate the MCMC method, assume that we want to generate a random variable X taking values in $\mathscr{X} = \{1, \ldots, m\}$, according to a target distribution $\{\pi_i\}$, with

$$\pi_i = \frac{b_i}{C}, \quad i \in \mathscr{X}, \tag{6.1}$$

where it is assumed that all $\{b_i\}$ are strictly positive, m is large, and the normalization constant $C = \sum_{i=1}^{m} b_i$ is difficult to calculate. Following Metropolis et al. [17], we construct a Markov chain $\{X_t,\ t = 0, 1, \ldots\}$ on $\mathscr{X}$ whose evolution relies on an arbitrary transition matrix $\mathbf{Q} = (q_{ij})$ in the following way:

- When $X_t = i$, generate a random variable Y satisfying $\mathbb{P}(Y = j) = q_{ij},\ j \in \mathscr{X}$. Thus, Y is generated from the m-point distribution given by the i-th row of $\mathbf{Q}$.
- If $Y = j$, let

$$X_{t+1} = \begin{cases} j & \text{with probability } \alpha_{ij} = \min\left\{\frac{\pi_j\, q_{ji}}{\pi_i\, q_{ij}}, 1\right\} = \min\left\{\frac{b_j\, q_{ji}}{b_i\, q_{ij}}, 1\right\}, \\ i & \text{with probability } 1 - \alpha_{ij}. \end{cases}$$

It follows that $\{X_t, t = 0, 1, \ldots\}$ has a one-step transition matrix $\mathbf{P} = (p_{ij})$ given by

$$p_{ij} = \begin{cases} q_{ij}\, \alpha_{ij}, & \text{if } i \neq j \\ 1 - \sum_{k \neq i} q_{ik}\, \alpha_{ik}, & \text{if } i = j\,. \end{cases} \tag{6.2}$$

Now it is easy to check (see Problem 6.1) that, with α_{ij} as above,

$$\pi_i\, p_{ij} = \pi_j\, p_{ji}, \quad i, j \in \mathscr{X}\,. \tag{6.3}$$

In other words, the *detailed balance equations* (1.38) hold, and hence the Markov chain is time reversible and has stationary probabilities $\{\pi_i\}$. Moreover, this stationary distribution is also the *limiting* distribution if the Markov chain is irreducible and aperiodic. Note that there is no need for the normalization constant C in (6.1) to define the Markov chain.

The extension of the above MCMC approach for generating samples from an arbitrary multidimensional pdf $f(\mathbf{x})$ (instead of π_i) is straightforward. In this case, the nonnegative probability transition function $q(\mathbf{x}, \mathbf{y})$ (taking the place of q_{ij} above) is often called the *proposal* or *instrumental* function. Viewing this function as a conditional pdf, one also writes $q(\mathbf{y} \mid \mathbf{x})$ instead of $q(\mathbf{x}, \mathbf{y})$. The probability $\alpha(\mathbf{x}, \mathbf{y})$ is called the *acceptance probability*. The original Metropolis algorithm [17] was suggested for symmetric proposal functions, that is, for $q(\mathbf{x}, \mathbf{y}) = q(\mathbf{y}, \mathbf{x})$. Hastings modified the original MCMC algorithm to allow nonsymmetric proposal functions. Such an algorithm is called a *Metropolis–Hastings algorithm*. We call the corresponding Markov chain the *Metropolis–Hastings Markov chain*.

In summary, the Metropolis–Hastings algorithm, which, like the acceptance-rejection method, is based on a trial-and-error strategy, is comprised of the following iterative steps:

Algorithm 6.2.1 (Metropolis–Hastings Algorithm)

Given the current state $\mathbf{X}_t$:

1. *Generate* $\mathbf{Y} \sim q(\mathbf{X}_t, \mathbf{y})$.
2. *Generate* $U \sim \mathsf{U}(0, 1)$ *and deliver*

$$\mathbf{X}_{t+1} = \begin{cases} \mathbf{Y}, & \text{if } U \leq \alpha(\mathbf{X}_t, \mathbf{Y}) \\ \mathbf{X}_t & \text{otherwise}\,, \end{cases} \tag{6.4}$$

where

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\{\varrho(\mathbf{x}, \mathbf{y}), 1\}\,, \tag{6.5}$$

with

$$\varrho(\mathbf{x}, \mathbf{y}) = \frac{f(\mathbf{y})\, q(\mathbf{y}, \mathbf{x})}{f(\mathbf{x})\, q(\mathbf{x}, \mathbf{y})}\,. \tag{6.6}$$

By repeating Steps 1 and 2, we obtain a sequence $\mathbf{X}_1, \mathbf{X}_2, \ldots$ of *dependent* random variables, with $\mathbf{X}_t$ approximately distributed according to $f(\mathbf{x})$, for large t.

Since Algorithm 6.2.1 is of the acceptance-rejection type, its efficiency depends on the acceptance probability $\alpha(\mathbf{x}, \mathbf{y})$. Ideally, one would like $q(\mathbf{x}, \mathbf{y})$ to reproduce the desired pdf $f(\mathbf{y})$ as faithfully as possible. This clearly implies maximization of $\alpha(\mathbf{x}, \mathbf{y})$. A common approach [19] is to first parameterize $q(\mathbf{x}, \mathbf{y})$ as $q(\mathbf{x}, \mathbf{y}; \theta)$ and then use stochastic optimization methods to maximize this with respect to θ. Below we consider several particular choices of $q(\mathbf{x}, \mathbf{y})$.

■ **EXAMPLE 6.1 Independence Sampler**

The simplest Metropolis-type MCMC algorithm is obtained by choosing the proposal function $q(\mathbf{x}, \mathbf{y})$ to be independent of $\mathbf{x}$, that is, $q(\mathbf{x}, \mathbf{y}) = g(\mathbf{y})$ for some pdf $g(\mathbf{y})$. Thus, starting from a previous state $\mathbf{X}$ a candidate state $\mathbf{Y}$ is generated from $g(\mathbf{y})$ and accepted with probability

$$\alpha(\mathbf{X}, \mathbf{Y}) = \min\left\{\frac{f(\mathbf{Y})\, g(\mathbf{X})}{f(\mathbf{X})\, g(\mathbf{Y})}, 1\right\}.$$

This procedure is very similar to the original acceptance-rejection methods of Chapter 2 and, as in that method, it is important that the proposal distribution g is close to the target f. Note, however, that in contrast to the acceptance-rejection method the independence sampler produces *dependent* samples.

■ **EXAMPLE 6.2 Uniform Sampling**

Being able to sample *uniformly* from some discrete set $\mathscr{Y}$ is very important in many applications; see, for example, the algorithms for counting in Chapter 9. A simple general procedure is as follows. Define a *neighborhood* structure on $\mathscr{Y}$. Any neighborhood structure is allowed, as long as the resulting Metropolis–Hastings Markov chain is irreducible and aperiodic. Let $n_{\mathbf{x}}$ be the number of neighbors of a state $\mathbf{x}$. For the proposal distribution, we simply choose each possible neighbor of the current state $\mathbf{x}$ with equal probability. That is, $q(\mathbf{x}, \mathbf{y}) = 1/n_{\mathbf{x}}$. Since the target pdf $f(\mathbf{x})$ here is constant, the acceptance probability is

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\{n_{\mathbf{x}}/n_{\mathbf{y}}, 1\} .$$

By construction, the limiting distribution of the Metropolis-Hastings Markov chain is the uniform distribution on $\mathscr{Y}$.

■ **EXAMPLE 6.3 Random Walk Sampler**

In the random walk sampler the proposal state $\mathbf{Y}$, for a given current state $\mathbf{x}$, is given by $\mathbf{Y} = \mathbf{x} + \mathbf{Z}$, where $\mathbf{Z}$ is typically generated from some spherically symmetrical distribution (in the continuous case), such as $\mathsf{N}(\mathbf{0}, \Sigma)$. Note that the proposal function is symmetrical in this case; thus,

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\left\{\frac{f(\mathbf{y})}{f(\mathbf{x})}, 1\right\}. \tag{6.7}$$

■ **EXAMPLE 6.4**

Let the random vector $\mathbf{X} = (X_1, X_2)$ have the following two-dimensional pdf:

$$f(\mathbf{x}) = c \exp(-(x_1^2 x_2^2 + x_1^2 + x_2^2 - 8x_1 - 8x_2)/2) , \tag{6.8}$$

where $c \approx 1/20216.335877$ is a normalization constant. The graph of this density is depicted in Figure 6.1.

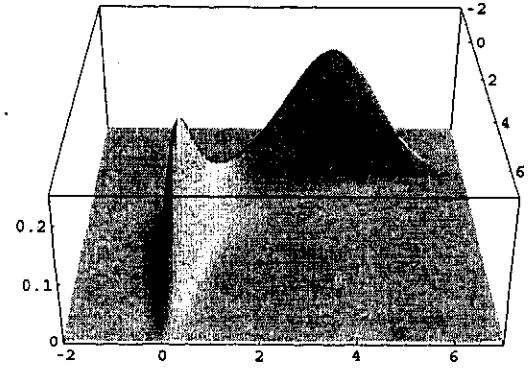

Figure 6.1 The density $f(x_1, x_2)$.

Suppose we wish to estimate $\ell = \mathbb{E}[X_1]$ via the CMC estimator

$$\widehat{\ell} = \frac{1}{N}\sum_{t=1}^{N} X_{t1},$$

using the random walk sampler to generate a dependent sample $\{\mathbf{X}_t\}$ from $f(\mathbf{x})$. A simple choice for the increment $\mathbf{Z}$ is to draw the components of $\mathbf{Z}$ independently, from a $\mathsf{N}(0, a^2)$ distribution for some $a > 0$. Note that if a is chosen too small, say less than 0.5, the components of the samples will be strongly positively correlated, which will lead to a large variance for $\widehat{\ell}$. On the other hand, for a too large, say greater than 10, most of the samples will be rejected, leading again to low efficiency. Below we choose a moderate value of a, say $a = 2$. The random walk sampler is now summarized as follows:

Procedure (Random Walk Sampler)

1. Initialize $\mathbf{X}_1 = (X_{11}, X_{12})$. Set $t = 1$.
2. Draw $Z_1, Z_2 \sim \mathsf{N}(0,1)$ independently. Let $\mathbf{Z} = (Z_1, Z_2)$ and $\mathbf{Y} = \mathbf{X}_t + 2\,\mathbf{Z}$. Calculate $\alpha = \alpha(\mathbf{X}_t, \mathbf{Y})$ as in (6.7).
3. Draw $U \sim \mathsf{U}[0,1]$. If $U < \alpha$, let $\mathbf{X}_{t+1} = \mathbf{Y}$; otherwise, let $\mathbf{X}_{t+1} = \mathbf{X}_t$.
4. Increase t by 1. If $t = N$ (sample size) stop; otherwise, repeat from Step 2.

We ran the above algorithm to produce $N = 10^5$ samples. The last few hundred of these are displayed in the left plot of Figure 6.2. We see that the samples closely follow the contour plot of the pdf, indicating that the correct region has been sampled. This is corroborated by the right plot of Figure 6.2, where we see that the histogram of the x_1 values is close to the true pdf (solid line).

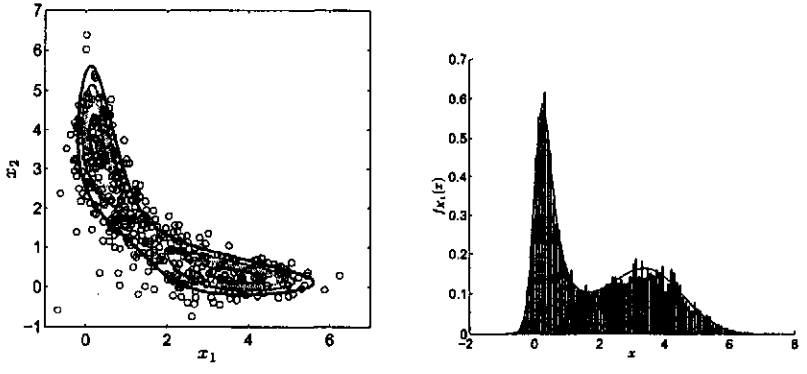

Figure 6.2 The left plot shows some samples of the random walk sampler along with several contour lines of f. The right plot shows the histogram of the x_1 values along with the true density of X_1.

We obtained an estimate $\widehat{\ell} = 1.89$ (the true value is $\mathbb{E}[X_1] \approx 1.85997$). To obtain a CI, we can use (4.18), where $\widetilde{S}$ estimates the asymptotic variance, or employ the batch

above uniform hit-and-run algorithm by accepting the candidate $\mathbf{y}$ with probability

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\{f(\mathbf{y})/f(\mathbf{x}), 1\}\,, \tag{6.9}$$

as in (6.4) (note that $q(\mathbf{y}, \mathbf{x})/q(\mathbf{x}, \mathbf{y})$ equals 1). Thus, the general hit-and-run algorithm with the above Metropolis acceptance criterion is summarized as follows [20].

Algorithm 6.3.1 (Continuous Hit-and-Run Algorithm)

1. *Initialize* $\mathbf{X}_1 \in \mathscr{X}$ *and set* $t = 1$.
2. *Generate a random direction* $\mathbf{d}_t$ *according to a uniform distribution on the unit* n*-dimensional hypersphere.*
3. *Generate a candidate point* $\mathbf{Y} = \mathbf{X}_t + \lambda \mathbf{d}_t$ *uniformly distributed over the line set*

$$\mathscr{M}_t = \{\mathbf{x} : \mathbf{x} \in \mathscr{X} \text{ and } \mathbf{x} = \mathbf{X}_t + \lambda \mathbf{d}_t,\ \lambda \in \mathbb{R}\}\,.$$

   If $\mathscr{M}_t = \emptyset$, *go to Step 2.*

4. *Set*

$$\mathbf{X}_{t+1} = \begin{cases} \mathbf{Y} & \text{with probability } \alpha(\mathbf{X}_t, \mathbf{Y}) \text{ in (6.9)} \\ \mathbf{X}_t & \text{otherwise.} \end{cases}$$

5. *If a stopping criterion is met, stop. Otherwise increment* t *and return to Step 2.*

Hit-and-run has been very successful over continuous domains. Recently an analogous sampler over a discrete domain has been developed in Baumert et al. [4]. Discrete hit-and-run generates two independent random walks on a lattice to form a bidirectional path that is analogous to the random bidirectional line generated in continuous hit-and-run. It then randomly selects a (feasible) discrete point along that path as the candidate point. By adding a Metropolis acceptance-rejection step, discrete hit-and-run converges to an arbitrary discrete target pdf f.

Let $\mathscr{X}$ be a bounded subset of $\mathbb{Z}^n$ — the set of n-dimensional vectors with integer valued coordinates — that is contained in the hyperrectangle $\mathscr{R} = \{\mathbf{x} \in \mathbb{Z}^n : l_i \le x_i \le u_i,\ i = 1, \ldots, n\}$. The discrete hit-and-run algorithm is stated below.

Algorithm 6.3.2 (Discrete Hit-and-Run Algorithm)

1. *Initialize* $\mathbf{X}_1 \in \mathscr{X}$ *and set* $t = 1$.
2. *Generate a bidirectional walk by generating two independent nearest neighbor random walks in* $\mathscr{R}$ *that start at* $\mathbf{X}_t$ *and end when they step out of* $\mathscr{R}$. *One random walk is called the* forward walk *and the other is called the* backward walk. *The bidirectional walk may have loops but has finite length with probability 1. The sequence of points visited by the bidirectional walk is stored in an ordered list, denoted* $\mathscr{L}_t$.
3. *Generate a candidate point* $\mathbf{Y}$ *uniformly distributed on the intersection* $\mathscr{M}_t = \mathscr{X} \cap \mathscr{L}_t$.
4. *Set*

$$\mathbf{X}_{t+1} = \begin{cases} \mathbf{Y} & \text{with probability } \alpha(\mathbf{X}_t, \mathbf{Y}) \text{ in (6.9)} \\ \mathbf{X}_t & \text{otherwise.} \end{cases}$$

5. *If a stopping criterion is met, stop. Otherwise, increment* t *and return to Step 2.*

Figure 6.5 illustrates the discrete hit-and-run algorithm with a bidirectional walk starting at point $\mathbf{x}$ on the discrete points in a square. The feasible set $\mathscr{X}$ is indicated by the three disconnected shaded regions. The candidate point $\mathbf{y}$ is chosen uniformly from the points in the bidirectional walk that are also in $\mathscr{X}$. It is accepted according to the Metropolis acceptance probability,

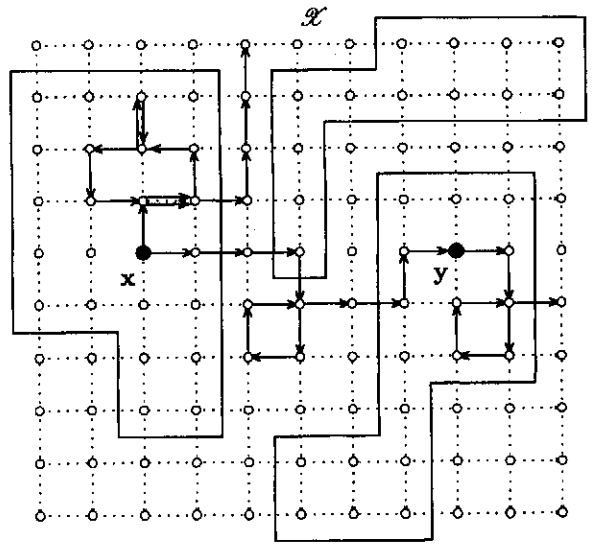

Figure 6.5 Illustration of the bidirectional walk in two dimensions.

It is shown in [4] that in several situations the rate of convergence of the discrete hit-and-run algorithm is polynomial of order n^4. Convergence to the target distribution has been proved in [4] for several variations of the bidirectional walk. One such candidate point generator, called *sphere biwalk*, not only converges to the target discrete pdf but also converges to continuous hit-and-run as the grid of the finite domain becomes finer [22].

The hit-and-run algorithm can be embedded within an optimization framework to yield two global optimization algorithms: *hide-and-seek* [20] and *improving hit-and-run* [26]. The latter has been applied successfully to practical problems including composite material design and shape optimization and has been shown to have polynomial complexity, on average, for a class of quadratic programs. In Section 6.8 we show how to turn an MCMC sampler into an optimization algorithm using simulated annealing.

6.4 THE GIBBS SAMPLER

The *Gibbs sampler* (Geman and Geman [6]) uses a somewhat different methodology from the Metropolis–Hastings algorithm and is particularly useful for generating n-dimensional random vectors. The distinguishing feature of the Gibbs sampler is that the underlying Markov chain is constructed, in a deterministic or random fashion, from a sequence of conditional distributions.

means method of Section 4.3.2.1. Figure 6.3 displays the estimated (auto)covariance function $\widehat{R}(k)$ for $k = 0, 1, \ldots, 400$. We see that up to about 100 the covariances are nonnegligible. Thus, to estimate the variance of $\widehat{\ell}$, we need to include all nonzero terms in (4.17), not only the variance $R(0)$ of X_1. Summing over the first 400 lags, we obtained an estimate of 10.41 for the asymptotic variance. This gives an estimated relative error for $\widehat{\ell}$ of 0.0185 and an 95% CI of (1.82, 1.96). A similar CI was found when using the batch means method with 500 batches of size 200.

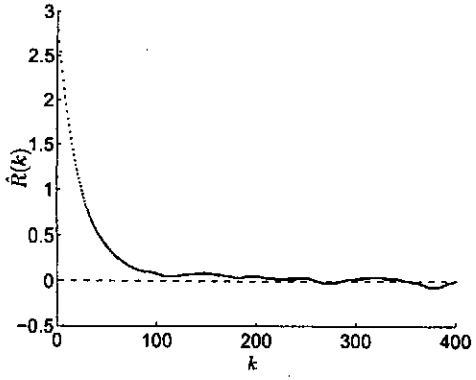

Figure 6.3 The estimated covariance function for the $\{X_{t1}\}$ for lags k up to 400.

While MCMC is a generic method and can be used to generate random samples virtually from any target distribution, regardless of its dimensionality and complexity, potential problems with the MCMC method are:

1. The resulting samples are often highly correlated.

2. Typically, it takes a considerable amount of time until the underlying Markov chain settles down to its steady state.

3. The estimates obtained via MCMC samples often tend to have much greater variances then those obtained from independent sampling of the target distribution. Various attempts have been made to overcome this difficulty. For details see, for example, [13] and [19].

Remark 6.2.1 At this point we must stress that although it is common practice to use MCMC to sample from $f(\mathbf{x})$ in order to estimate any expectation $\ell = \mathrm{E}_f[H(\mathbf{X})]$, the *actual* target for estimating ℓ is $g^*(\mathbf{x}) \propto |H(\mathbf{x})| f(\mathbf{x})$. Namely, sampling from $g^*(\mathbf{x})$ gives a minimum variance estimator (zero variance in the case $H(\mathbf{x}) \geqslant 0$). Thus, it is important to distinguish clearly between using MCMC for generating from some difficult pdf $f(\mathbf{x})$ and using MCMC to estimate a quantity such as ℓ. For the latter problem, much more efficient techniques can be used, such as importance sampling; moreover, a good importance sampling pdf can be obtained adaptively, as with the CE and TLR methods.

6.3 THE HIT-AND-RUN SAMPLER

The *hit-and-run* sampler, pioneered by Robert Smith [24], is among the first MCMC samplers in the category of *line samplers* [2]. As in the previous section, the objective is to sample from a target distribution $f(\mathbf{x})$ on $\mathscr{X} \subset \mathbb{R}^n$. Line samplers afford the opportunity to reach across the entire feasible region $\mathscr{X}$ in one step.

We first describe the original hit-and-run sampler for generating from a *uniform* distribution on a bounded open region $\mathscr{X}$ of $\mathbb{R}^n$. At each iteration, starting from a current point $\mathbf{x}$, a *direction vector* $\mathbf{d}$ is generated uniformly on the surface of an n-dimensional hypersphere. The intersection of the corresponding bidirectional line (through $\mathbf{x}$) and the enclosing box of $\mathscr{X}$ defines a line segment $\mathscr{L}$. The next point $\mathbf{y}$ is then selected uniformly from the intersection of $\mathscr{L}$ and $\mathscr{X}$.

Figure 6.4 illustrates the hit-and-run algorithm for generating uniformly from the set $\mathscr{X}$ (the gray region), which is bounded by a square. Given the point $\mathbf{x}$ in $\mathscr{X}$, a random direction $\mathbf{d}$ is generated, which defines the line segment $\mathscr{L} = uv$. Then a point $\mathbf{y}$ is chosen uniformly on $\mathscr{M} = \mathscr{L} \cap \mathscr{X}$, for example, by the acceptance-rejection method; that is, one generates a point uniformly on $\mathscr{L}$ and then accepts this point only if it lies in $\mathscr{X}$.

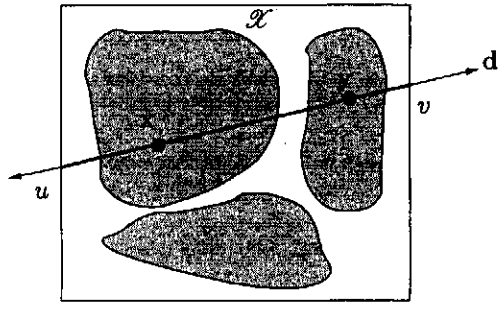

Figure 6.4 Illustration of the hit-and-run algorithm on a square in two dimensions.

Smith [24] showed that hit-and-run asymptotically generates uniformly distributed points over *arbitrary* open regions of $\mathbb{R}^n$. One desirable property of hit-and-run is that it can globally reach any point in the set in one step, that is, there is a strictly positive probability of sampling any neighborhood in the set. This property, coupled with a symmetry property, is important in deriving the limiting distribution. Lovász [14] proved that hit-and-run on a convex body in n dimensions produces an approximately uniformly distributed sample point in polynomial time, $\mathcal{O}(n^3)$, the best-known bound for such a sampling algorithm. He noted that the hit-and-run algorithm appears in practice to offer the most rapid convergence to a uniform distribution [14, 15]. Hit-and-run is unique in that it only takes polynomial time to get out of a corner; in contrast, *ball walk* takes exponential time to get out of a corner [16].

Note that the hit-and-run algorithm described above is a special case of the Metropolis–Hastings Algorithm 6.2.1, where the proposal function $q(\mathbf{x}, \mathbf{y})$ is symmetric and the target $f(\mathbf{x})$ is constant. It follows that each candidate point is accepted with probability 1. To generate from a *general* strictly positive continuous pdf $f(\mathbf{x})$, one can simply modify the

Gibbs sampling is advantageous if it is easier to sample from the conditional distributions than from the joint distribution. The essential idea of the Gibbs sampler — updating one part of the previous element while keeping the other parts fixed — is useful in many instances where the state variable is a random variable taking values in a general space, not just in $\mathbb{R}^n$; see [11].

Suppose that we wish to sample a random vector $\mathbf{X} = (X_1, \ldots, X_n)$ according to a target pdf $f(\mathbf{x})$. Let $f(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ represent the conditional pdf of the i-th component, X_i, given the other components $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. The Gibbs sampler consists of the following iterative steps.

Algorithm 6.4.1 (Gibbs Sampler)

A. For a given $\mathbf{X}_t$, generate $\mathbf{Y} = (Y_1, \ldots, Y_n)$ as follows:

1. *Draw Y_1 from the conditional pdf $f(x_1 \mid X_{t,2}, \ldots, X_{t,n})$.*
2. *Draw Y_i from $f(x_i \mid Y_1, \ldots, Y_{i-1}, X_{t,i+1}, \ldots, X_{t,n})$, $i = 2, \ldots, n-1$.*
3. *Draw Y_n from $f(x_n \mid Y_1, \ldots, Y_{n-1})$.*

B. Let $\mathbf{X}_{t+1} = \mathbf{Y}$.

Note that in the Gibbs sampler *all* samples are accepted, in contrast to the Metropolis–Hastings algorithm. We will see in Section 6.7 that under mild conditions the limiting distribution of the process $\{\mathbf{X}_t, t = 1, 2, \ldots\}$, generated via the Gibbs sampler, is precisely $f(\mathbf{x})$. Moreover, under some other simple conditions, it can be shown (see [13], [19]) that the convergence to the desired pdf is geometrically fast.

■ **EXAMPLE 6.5 Example 6.4 (Continued)**

We shall show how to sample easily from the pdf f in (6.8) via the Gibbs sampler. Namely, writing

$$f(x, y) = c_1(y) \exp\left(-\frac{1+y^2}{2}\left(x - \frac{4}{1+y^2}\right)^2\right),$$

where $c_1(y)$ depends only on y, we see that, conditional on y, X has a normal distribution with expectation $4/(1+y^2)$ and variance $1/(1+y^2)$. The conditional distribution of Y given x follows in the same way. The corresponding Gibbs sampler is thus:

Procedure

1. Initialize X_1 and Y_1. Set $t = 1$.
2. If t is odd, draw $Z \sim \mathsf{N}(0,1)$. Put $a = 1/(1+Y_t^2)$. Set $Y_{t+1} = 4a + Z\sqrt{a}$ and $X_{t+1} = X_t$.
3. If t is even, draw $Z \sim \mathsf{N}(0,1)$. Put $a = 1/(1+X_t^2)$. Set $X_{t+1} = 4a + Z\sqrt{a}$ and $Y_{t+1} = Y_t$.
4. Increase t by 1. If $t = N$ (sample size) stop; otherwise, repeat from Step 2.

Remark 6.4.1 (Systematic and Random Gibbs Samplers) Note that Algorithm 6.4.1 presents a *systematic* coordinatewise Gibbs sampler. That is, the vector $\mathbf{X}$ is updated in a deterministic order: $1, 2, \ldots, n, 1, 2, \ldots$. In the *random* coordinatewise Gibbs sampler the coordinates are chosen randomly, such as by generating them independently from a discrete uniform n-point pdf. In that case the Gibbs sampler can be viewed as an instance of the Metropolis–Hastings sampler, namely, with the transition function

$$q(\mathbf{x}, \mathbf{y}) = \frac{1}{n} f(y_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = \frac{1}{n} \frac{f(\mathbf{y})}{\sum_{y_i} f(\mathbf{y})},$$

where $\mathbf{y} = (x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n)$. Since $\sum_{y_i} f(\mathbf{y})$ can also be written as $\sum_{x_i} f(\mathbf{x})$, we have

$$\varrho(\mathbf{x}, \mathbf{y}) = \frac{f(\mathbf{y})\, q(\mathbf{y}, \mathbf{x})}{f(\mathbf{x})\, q(\mathbf{x}, \mathbf{y})} = \frac{f(\mathbf{y})\, f(\mathbf{x})}{f(\mathbf{x})\, f(\mathbf{y})} = 1,$$

so that the acceptance probability $\alpha(\mathbf{x}, \mathbf{y})$ is 1 in this case.

Here is another example of an application of the Gibbs sampler.

■ **EXAMPLE 6.6 Closed Network of Queues in a Product Form**

Consider m customers moving among n queues in a closed queueing network. Denote by $X_i(t)$ the number of customers in queue i, $i = 1, \ldots, n$, and let $\mathbf{X}(t) = (X_1(t), \ldots, X_n(t))$ and $\mathbf{x} = (x_1, \ldots, x_n)$. It is well known [21] that if the limit

$$\lim_{t \to \infty} \mathbb{P}(\mathbf{X}(t) = \mathbf{x}) = \pi(\mathbf{x})$$

exists, then, for exponentially distributed service times, the joint discrete pdf $\pi(\mathbf{x})$ can be written in *product form* as

$$\pi(\mathbf{x}) = C \prod_{i=1}^{n} f_i(x_i), \quad \text{for } \sum_{i=1}^{n} x_i = m, \tag{6.10}$$

where the $\{f_i(x_i),\ x_i \geq 0\}$ are *known* discrete pdfs, and C is a normalization constant. For a concrete example see Problem 6.11.

The constant C is in general difficult to compute. To proceed, writing $S(\mathbf{x}) = \sum_{i=1}^{n} x_i$ and $\mathscr{X}^* = \{\mathbf{x} : S(\mathbf{x}) = m\}$, we have

$$C^{-1} = \sum_{\mathbf{x} \in \mathscr{X}^*} \prod_{i=1}^{n} f_i(x_i), \tag{6.11}$$

which requires the evaluation of the product of n pdfs for each $\mathbf{x}$ in the set $\mathscr{X}^*$. This set has a total of $|\mathscr{X}^*| = \binom{m+n-1}{n-1}$ elements (see Problem 6.10), which rapidly grows very large.

We now show how to compute C based on Gibbs sampling. To apply the Gibbs sampler, we need to be able to generate samples from the conditional distribution of X_i given the other components. Note that we only have to generate $X_1, \ldots, X_{n-1}$, since $X_n = m - \sum_{k=1}^{n-1} X_k$. For $i = 1, \ldots, n-1$ we have

$$f(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n-1}) \propto f_i(x_i)\, f_n(m - \sum_{k=1}^{n-1} x_i) \tag{6.12}$$

for $x_i \in \{0, 1, \ldots, m - x_1 - \cdots - x_{i-1} - x_{i+1} - \cdots - x_{n-1}\}$. Sampling from these conditional pdfs can often be done efficiently, in particular when the $\{f_i\}$ are members of an exponential family; see also Problem 6.11.

Now that we can sample (approximately) from $\pi(\mathbf{x})$, it is straightforward to estimate the normalization constant C by observing that

$$\mathbf{E}_\pi \left[\frac{1}{\prod_{i=1}^n f_i(X_i)} \right] = \sum_{\mathbf{x} \in \mathscr{X}^*} \frac{1}{\prod_{i=1}^n f_i(x_i)} C \prod_{i=1}^n f_i(x_i) = |\mathscr{X}^*| C \,.$$

This suggests the following estimator for C, obtained from a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from π:

$$\widehat{C} = \binom{m+n-1}{n-1}^{-1} \frac{1}{N} \sum_{k=1}^N \prod_{i=1}^n \frac{1}{f_i(X_{ki})} \,,$$

where X_{ki} is the i-th component of $\mathbf{X}_k$.

6.5 ISING AND POTTS MODELS

6.5.1 Ising Model

The Ising model is one of the most popular and most extensively studied models in statistical mechanics. It describes the interaction of idealized magnets, called *spins*, that are located on a two- or three-dimensional lattice. In the basic two-dimensional case the spins are located on the lattice $\{1, \ldots, n\} \times \{1, \ldots, n\}$, and each of the n^2 sites has four nearest neighbors, possibly including boundary sites, which "wrap around" to the other side of the lattice, creating a so-called *torus*. See Figure 6.6, where the four light gray sites are the neighbors of the dark gray site.

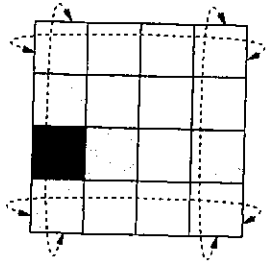

Figure 6.6 The boundary sites wrap around. The neighbors of the dark gray site are the light gray sites.

Let $\{1, \ldots, n^2\}$ be an enumeration of the sites. Each spin can be in one of two states: -1 or 1. Each of the 2^{n^2} *configurations* of spins $\mathbf{s} = (s_1, \ldots, s_{n^2})$ carries an amount of *total energy*

$$E(\mathbf{s}) = -A \sum_{i \leftrightarrow j} s_i s_j - B \sum_i s_i \,,$$

where A and B are constants; in most studies $A = 1$ and $B = 0$, which we will now assume. The quantities $\sum_{i \leftrightarrow j} s_i s_j$ and $\sum_i s_i$ are called the *interaction energy* and *magnetization*,

respectively. The notation $\sum_{i \leftrightarrow j}$ indicates that the summation is taken over neighboring pairs (i, j).

In thermal equilibrium the distribution of the spins, say π, follows the Boltzmann law: $\pi(\mathbf{s}) \propto \exp(-E(\mathbf{s})/T)$, where T is a fixed temperature. In other words, we have

$$\pi(\mathbf{s}) = \frac{e^{\frac{1}{T} \sum_{i \leftrightarrow j} s_i s_j}}{\mathscr{Z}} \,,$$

where $\mathscr{Z}$ is the normalization constant, called the *partition function*. Apart from $\mathscr{Z}$, particular quantities of interest are the *mean energy per spin* $\mathbb{E}_\pi[\sum_{i \leftrightarrow j} S_i S_j / n^2]$ and the *mean magnetization per spin* $\mathbb{E}_\pi[\sum_i S_i / n^2]$. These quantities can be obtained via Monte Carlo simulation, provided that one can sample efficiently from the target distribution π (see below).

In Figure 6.7 a sample from π is given (black = 1, white = -1) for $n = 30$ at the so-called *critical temperature* $T = 2/\ln(1 + \sqrt{2}) \approx 2.269$.

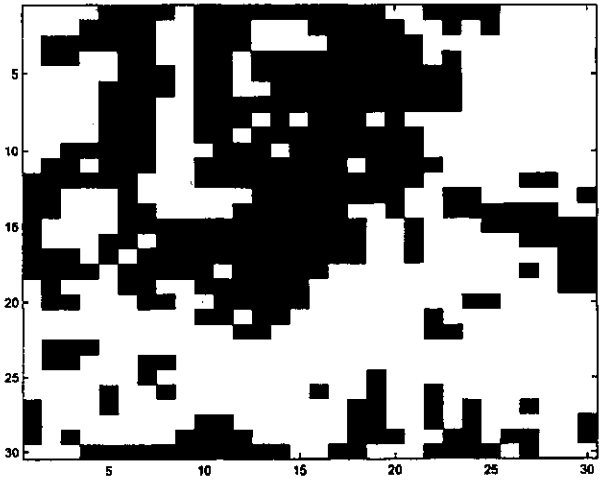

Figure 6.7 Ising configuration at the critical temperature.

We next define the *Potts model* — which can be viewed as a generalization of the Ising model — and explain how to generate samples from this extended model and thus, in particular, how to generate Figure 6.7.

6.5.2 Potts Model

Let $\{1, \ldots, J\}$ be an enumeration of spatial positions (sites), and let ψ_{ij} be some symmetrical and positive function relating the sites to each other, for example,

$$\psi_{ij} = \begin{cases} \beta\,(>0) & \text{if } i \text{ and } j \text{ are neighbors} \\ 0 & \text{otherwise.} \end{cases} \tag{6.13}$$

Assign to each site i a "color" x_i. Suppose there are K such colors, labeled $\{1, \ldots, K\}$. Define $\mathbf{x} = (x_1, \ldots, x_J)$ and let $\mathscr{X}$ be the space of such configurations. On $\mathscr{X}$ we define

the target pdf $f(\mathbf{x}) \propto \mathrm{e}^{H(\mathbf{x})}$ with

$$H(\mathbf{x}) = \sum_{i<j} \psi_{ij}\, I_{\{x_i = x_j\}} .$$

To see that the Ising model is a special case of the Potts model, define $x_i = I_{\{s_i = 1\}}$ and ψ_{ij} as in (6.13), with $\beta = 4/T$. Then

$$\frac{1}{T}\sum_{i \leftrightarrow j} s_i\, s_j = \frac{1}{T}\sum_{i \leftrightarrow j} 2\left(I_{\{x_i = x_j\}} - \frac{1}{2}\right) = \sum_{i<j} \psi_{ij}\, I_{\{x_i = x_j\}} + \text{const} ,$$

so that $\pi(\mathbf{s}) = f(\mathbf{x})$.

Next, we show how to generate a sample from the target pdf $f(\mathbf{x})$. To do so, we define auxiliary random variables $Y_{ij}, 1 \leqslant i < j \leqslant J$, such that conditional on $\mathbf{X} = \mathbf{x}$ the $\{Y_{ij}\}$ are independent, and each Y_{ij} is uniformly distributed on the interval $[0, a_{ij}]$, with $a_{ij} = \exp(\psi_{ij}\, I_{\{x_i = x_j\}}) \geqslant 1$. In other words, the conditional pdf of $\mathbf{Y} = \{Y_{ij}\}$ given $\mathbf{X} = \mathbf{x}$ is

$$f(\mathbf{y} \,|\, \mathbf{x}) = \prod_{i<j} \frac{I_{\{y_{ij} \leqslant a_{ij}\}}}{a_{ij}} = \prod_{i<j} I_{\{y_{ij} \leqslant a_{ij}\}}\, \mathrm{e}^{-H(\mathbf{x})} .$$

The significance of this is that the joint pdf of $\mathbf{X}$ and $\mathbf{Y}$ is now simply

$$f(\mathbf{x}, \mathbf{y}) = f(\mathbf{x})\, f(\mathbf{y}|\mathbf{x}) \propto \begin{cases} 1 & \text{if } y_{ij} \leqslant a_{ij}, \text{ for all } i < j , \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $(\mathbf{X}, \mathbf{Y})$ is *uniformly* distributed. More importantly, because $f(\mathbf{x} \,|\, \mathbf{y}) \propto f(\mathbf{x}, \mathbf{y})$, we find that $\mathbf{X} \,|\, \mathbf{y}$ is uniformly distributed over the set $\mathscr{A} = \{\mathbf{x} : y_{ij} \leqslant \exp(\psi_{ij}\, I_{\{x_i = x_j\}}) \text{ for all } i < j\}$. Now, either $y_{ij} \in [0, 1]$ or $y_{ij} \in (1, \mathrm{e}^{\psi_{ij}}]$. In the former case, for any $\mathbf{x} \in \mathscr{A}$, the coordinates x_i and x_j range over all the colors, and, by the uniformity, each color is equally likely. But in the latter case, x_i must be equal to x_j. Thus, for a given $\mathbf{y}$, the sites i, j (with $i < j$) for which $y_{ij} > 1$ can be gathered into clusters, and within each such cluster the sites have identical colors. Moreover, given $\mathbf{y}$, the colors within the clusters are independent and uniformly distributed on $\{1, \ldots, K\}$. The same holds for the colors of the remaining positions, which can be viewed as one-cluster sites.

Hence, we can easily generate both $\mathbf{X} \,|\, \mathbf{y}$ and $\mathbf{Y} \,|\, \mathbf{x}$. As a consequence, we can use the Gibbs sampler to (approximately) sample from $f(\mathbf{x}, \mathbf{y})$; that is, we iteratively sample from $f(\mathbf{x} \,|\, \mathbf{y})$ and $f(\mathbf{y} \,|\, \mathbf{x})$. Finally, to obtain a sample $\mathbf{X}$ from $f(\mathbf{x})$, we generate $(\mathbf{X}, \mathbf{Y})$ via the Gibbs sampler and simply ignore $\mathbf{Y}$.

To simplify matters further, note that instead of the exact value Y_{ij} it suffices to know only the variable $B_{ij} = I_{\{Y_{ij} \geqslant 1\}}$. Given $\mathbf{X} = \mathbf{x}$, B_{ij} has a $\mathrm{Ber}(1 - \mathrm{e}^{-\psi_{ij}})$ distribution if $x_i = x_j$, and $B_{ij} = 0$ otherwise. This leads to the following so-called Swendsen–Wang algorithm.

Algorithm 6.5.1 (Swendsen–Wang)

1. *Given* $\{X_i\}$, *generate* $B_{ij} \sim \mathrm{Ber}(I_{\{X_i = X_j\}}(1 - e^{-\psi_{ij}}))$ *for* $1 \leqslant i < j \leqslant J$.
2. *Given* $\{B_{ij}\}$, *generate* $X_i, i = 1, \ldots, J$ *by clustering all the sites and choosing each cluster color independently and uniformly from* $\{1, \ldots, K\}$.

Remark 6.5.1 (Data Augmentation) The above idea of introducing an *auxiliary* variable $\mathbf{y}$ to make sampling from $f(\mathbf{x})$ easier is also known as *data augmentation*. The composition method described in Section 2.3.3 can be viewed as another example of data augmentation. Namely, suppose we want to sample from the mixture pdf

$$f(x) = \sum_{i=1}^{K} p_i\, f_i(x) .$$

Let Y be the discrete random variable taking values in $\{1, \ldots, K\}$ corresponding to the probabilities $\{p_i\}$. Using the composition method, it is easy to sample from the joint pdf of X and Y: first, draw Y according to $\{p_i\}$ and then sample X conditional on $Y = i$; that is, sample from $f_i(x)$. By simply ignoring Y, we obtain a sample from $f(x)$.

6.6 BAYESIAN STATISTICS

One of the main application areas of the MCMC method is Bayesian statistics. The mainstay of the Bayesian approach is Bayes' rule (1.6), which, in terms of pdfs, can be written as

$$f(\mathbf{y} \,|\, \mathbf{x}) = \frac{f(\mathbf{x} \,|\, \mathbf{y})\, f(\mathbf{y})}{\int f(\mathbf{x} \,|\, \mathbf{y})\, f(\mathbf{y})\, d\mathbf{y}} \propto f(\mathbf{x} \,|\, \mathbf{y})\, f(\mathbf{y}) . \tag{6.14}$$

In other words, for any two random variables $\mathbf{X}$ and $\mathbf{Y}$, the conditional distribution of $\mathbf{Y}$ given $\mathbf{X} = \mathbf{x}$ is proportional to the product of the conditional pdf of $\mathbf{X}$ given $\mathbf{Y} = \mathbf{y}$ and the pdf of $\mathbf{Y}$. Note that instead of writing f_X, f_Y, $f_{X|Y}$, and $f_{Y|X}$ in the formula above, we have used the *same letter* f for the pdf of $\mathbf{X}$, $\mathbf{Y}$, and the conditional pdfs. This particular style of notation is typical in Bayesian analysis and can be of great descriptive value, despite its apparent ambiguity. We will use this notation whenever we work in a Bayesian setting.

The significance of (6.14) becomes clear when it is employed in the context of Bayesian parameter estimation, sometimes referred to as *Bayesian learning*. The following example explains the ideas.

■ **EXAMPLE 6.7 Coin Flipping and Bayesian Learning**

Consider the basic random experiment in Example 1.1 on page 3, where we toss a biased coin n times. Suppose that the outcomes are $x_1, \ldots, x_n$, with $x_i = 1$ if the i-th toss is heads and $x_i = 0$ otherwise, $i = 1, \ldots, n$. Let p denote the probability of heads. We want to obtain information about p from the data $\mathbf{x} = (x_1, \ldots, x_n)$, for example, construct a CI.

The crucial idea is to summarize the information about p via a probability density $f(p)$. For example, if we know nothing about p, we take $f(p)$ uniformly distributed on the $(0, 1)$ interval, that is, $f(p) = 1, 0 \leqslant p \leqslant 1$. In effect, we treat p as a random variable. Now, obviously, the data $\mathbf{x}$ will affect our knowledge of p, and the way to update this information is to use Bayes' formula:

$$f(p \,|\, \mathbf{x}) \propto f(\mathbf{x} \,|\, p)\, f(p) .$$

The density $f(p)$ is called the *prior* density; $f(p \,|\, \mathbf{x})$ is called the *posterior density*; and $f(\mathbf{x} \,|\, p)$ is referred to as the *likelihood*. In our case, given p, the $\{X_i\}$ are independent and $\mathrm{Ber}(p)$ distributed, so that

$$f(\mathbf{x} \,|\, p) = \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} = p^s\,(1-p)^{n-s} ,$$

with $s = x_1 + \cdots + x_n$ representing the total number of successes. Thus, using a uniform prior ($f(p) = 1$) gives a posterior pdf

$$f(p \,|\, \mathbf{x}) = c\, p^s (1-p)^{n-s} ,$$

which is the pdf of the Beta$(s+1, n-s+1)$ distribution. The normalization constant is $c = (n+1)\binom{n}{s}$.

A Bayesian CI for p is now formed by taking the appropriate quantiles of the posterior pdf. As an example, suppose that $n = 100$ and $s = 1$. Then, a left one-sided 95% CI for p is $[0, 0.0461]$, where 0.0461 is the 0.95 quantile of the Beta$(2, 100)$ distribution. As an estimate for p, one often takes the value for which the pdf is maximal, the so-called *mode* of the pdf. In this case, the mode is 0.01, coinciding with the sample mean. Figure 6.8 gives a plot of the posterior pdf for this problem.

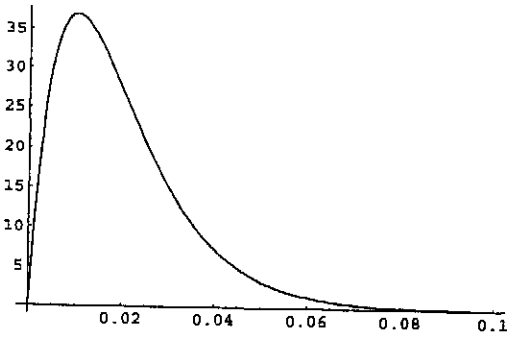

Figure 6.8 Posterior pdf for p, with $n = 100$ and $s = 1$.

Generalizing the previous example, a typical situation where MCMC (in particular Gibbs sampling) can be used in Bayesian statistics is the following. Suppose we want to sample from a posterior density $f(\boldsymbol{\theta} \,|\, \mathbf{x})$, where the data $\mathbf{x}$ are given (fixed) and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ is the parameter of interest. Suppose that it is easy to sample from $f(\theta_i \,|\, \theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_k, \mathbf{x})$ for all i. Then we can use the Gibbs sampler to obtain a sample $\boldsymbol{\Theta}$ from $f(\boldsymbol{\theta} \,|\, \mathbf{x})$. The following example, adapted from Gelman et al. [5], illustrates the general idea.

■ EXAMPLE 6.8 Poisson Disruption Problem

Suppose the random variables $X_1, \ldots, X_n$ describe the number of disasters in n subsequent years. In some random year K the rate of disasters changes from λ_1 to λ_2. Such a K is often called a *change point*. Our prior knowledge of λ_i is summarized by a Gamma(a_i, η_i), where shape parameter a_i is known. In turn, η_i is given by a Gamma(b_i, c_i) distribution, where both b_i and c_i are known. Let $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$ and $\boldsymbol{\eta} = (\eta_1, \eta_2)$. We are given the data $\mathbf{x} = (x_1, \ldots, x_n)$, and the objective is to simulate from the posterior distribution of $\boldsymbol{\theta} = (\lambda_1, \lambda_2, \eta_1, \eta_2, K)$ given $\mathbf{x}$.

For the model we have the following hierarchical structure:

1. K has some discrete pdf $f(K)$ on $1, \ldots, n$.
2. Given K, the $\{\eta_i\}$ are independent and have a Gamma(b_i, c_i) distribution for $i = 1, 2$.
3. Given K and $\boldsymbol{\eta}$, the $\{\lambda_i\}$ are independent and have a Gamma(a_i, η_i) distribution for $i = 1, 2$.
4. Given $K, \boldsymbol{\eta}$, and $\boldsymbol{\lambda}$, the $\{X_i\}$ are independent and have a Poi(λ_1) distribution for $i = 1, \ldots, K$, and a Poi(λ_2) distribution for $i = K+1, \ldots, n$.

It follows from point 4. that

$$\begin{aligned} f(\mathbf{x} \,|\, \boldsymbol{\lambda}, \boldsymbol{\eta}, K) &= \prod_{i=1}^{K} \mathrm{e}^{-\lambda_1} \frac{\lambda_1^{x_i}}{x_i!} \prod_{i=K+1}^{n} \mathrm{e}^{-\lambda_2} \frac{\lambda_2^{x_i}}{x_i!} \\ &= \mathrm{e}^{-\lambda_1 K} \lambda_1^{\sum_{i=1}^{K} x_i} \mathrm{e}^{-\lambda_2 (n-K)} \lambda_2^{\sum_{i=K+1}^{n} x_i} \prod_{i=1}^{n} \frac{1}{x_i!} . \end{aligned}$$

Moreover, by the product rule (1.4), the joint pdf is given by

$$\begin{aligned} f(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\eta}, K) \propto f(K)\, & \mathrm{e}^{-\lambda_1 K} \lambda_1^{\sum_{i=1}^{K} x_i} \mathrm{e}^{-\lambda_2 (n-K)} \lambda_2^{\sum_{i=K+1}^{n} x_i} \prod_{i=1}^{n} \frac{1}{x_i!} \\ & \times \mathrm{e}^{-\eta_1 \lambda_1} \lambda_1^{a_1 - 1} \eta_1^{a_1} \times \mathrm{e}^{-\eta_2 \lambda_2} \lambda_2^{a_2 - 1} \eta_2^{a_2} \\ & \times \mathrm{e}^{-c_1 \eta_1} \eta_1^{b_1 - 1} c_1^{b_1} \times \mathrm{e}^{-c_2 \eta_2} \eta_2^{b_2 - 1} c_2^{b_2} . \end{aligned}$$

As a consequence,

$$f(\lambda_1 \,|\, \lambda_2, \boldsymbol{\eta}, K, \mathbf{x}) \propto \mathrm{e}^{-\lambda_1 (K + \eta_1)} \lambda_1^{a_1 - 1 + \sum_{i=1}^{K} x_i} .$$

In other words, $(\lambda_1 \,|\, \lambda_2, \boldsymbol{\eta}, K, \mathbf{x}) \sim \text{Gamma}(a_1 + \sum_{i=1}^{K} x_i,\ K + \eta_1)$. In a similar way, we have

$$\begin{aligned} (\lambda_2 \,|\, \lambda_1, \boldsymbol{\eta}, K, \mathbf{x}) &\sim \text{Gamma}\left(a_2 + \textstyle\sum_{i=K+1}^{n} x_i,\ n - K + \eta_2\right) , \\ (\eta_1 \,|\, \boldsymbol{\lambda}, \eta_2, K, \mathbf{x}) &\sim \text{Gamma}(a_1 + b_1,\ \lambda_1 + c_1) , \\ (\eta_2 \,|\, \boldsymbol{\lambda}, \eta_1, K, \mathbf{x}) &\sim \text{Gamma}(a_2 + b_2,\ \lambda_2 + c_2) , \\ f(K \,|\, \boldsymbol{\lambda}, \boldsymbol{\eta}, \mathbf{x}) &\propto f(K)\, \mathrm{e}^{-K(\lambda_1 + \lambda_2)} (\lambda_1/\lambda_2)^{\sum_{i=1}^{K} x_i} , \end{aligned}$$

so that Gibbs sampling may be used to sample from the posterior pdf $f(\boldsymbol{\lambda}, \boldsymbol{\eta}, K \,|\, \mathbf{x})$.

6.7 * OTHER MARKOV SAMPLERS

There exist many variants of the Metropolis–Hastings and Gibbs samplers. However, all of the known MCMC algorithms can be described via the following framework. Consider a Markov chain $\{(\mathbf{X}_n, \mathbf{Y}_n), n = 0, 1, 2, \ldots\}$ on the set $\mathscr{X} \times \mathscr{Y}$, where $\mathscr{X}$ is the target set and $\mathscr{Y}$ is an auxiliary set. Let $f(\mathbf{x})$ be the target pdf. Each transition of the Markov chain consists of two parts. The first is $(\mathbf{x}, \bar{\mathbf{y}}) \to (\mathbf{x}, \mathbf{y})$, according to a transition matrix $\mathbf{Q}$; the second is $(\mathbf{x}, \mathbf{y}) \to (\mathbf{x}', \mathbf{y}')$, according to a transition matrix $\mathbf{R}$. In other words, the transition matrix $\mathbf{P}$ of the Markov chain is given by the product $\mathbf{Q}\,\mathbf{R}$. Both steps are illustrated in Figure 6.9 and further explained below.

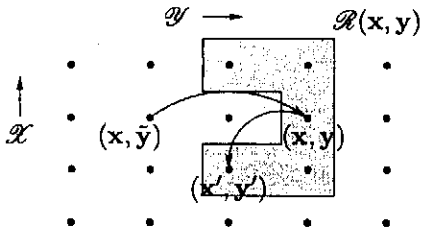

Figure 6.9 Each transition of the Markov chain consists of two steps: the Q-step, followed by the R-step.

The first step, the *Q-step*, changes the y-coordinate but leaves the x-coordinate intact. In particular, $\mathbf{Q}$ is of the form $\mathbf{Q}[(\mathbf{x},\tilde{\mathbf{y}}),(\mathbf{x},\mathbf{y})] = \mathbf{Q}_{\mathbf{x}}(\tilde{\mathbf{y}},\mathbf{y})$, where $\mathbf{Q}_{\mathbf{x}}$ is a transition matrix on $\mathscr{Y}$. Let $q_{\mathbf{x}}$ be a stationary distribution for $\mathbf{Q}_{\mathbf{x}}$, assuming that it exists.

The second step, the *R-step*, is determined by (a) the stationary distribution $q_{\mathbf{x}}$ and (b) a neighborhood structure on the set $\mathscr{X} \times \mathscr{Y}$. Specifically, we define for each point $(\mathbf{x},\mathbf{y})$ a set of neighbors $\mathscr{R}(\mathbf{x},\mathbf{y})$ such that *if* $(\mathbf{x}',\mathbf{y}')$ *is a neighbor of* $(\mathbf{x},\mathbf{y})$ *then the converse is also true*; see Figure 6.9, where the shaded area indicates the neighborhood set of $(\mathbf{x},\mathbf{y})$. The crucial step is now to define the transition matrix $\mathbf{R}$ as

$$\mathbf{R}[(\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')] = c(\mathbf{x},\mathbf{y})\, f(\mathbf{x}')\, q_{\mathbf{x}'}(\mathbf{y}') \quad \text{for all} \quad (\mathbf{x}',\mathbf{y}') \in \mathscr{R}(\mathbf{x},\mathbf{y}) \;,$$

where $c(\mathbf{x},\mathbf{y}) = \sum_{(\mathbf{x}',\mathbf{y}')\in\mathscr{R}(\mathbf{x},\mathbf{y})} f(\mathbf{x}')\, q_{\mathbf{x}'}(\mathbf{y}')$. Note that $c(\mathbf{x},\mathbf{y}) = c(\mathbf{x}',\mathbf{y}')$ when $(\mathbf{x},\mathbf{y})$ and $(\mathbf{x}',\mathbf{y}')$ belong to the same neighborhood set. With this choice of $\mathbf{Q}$ and $\mathbf{R}$ it can be shown (see Problem 6.15) that the Markov chain has a stationary distribution

$$\mu(\mathbf{x},\mathbf{y}) = f(\mathbf{x})\, q_{\mathbf{x}}(\mathbf{y}) \;, \tag{6.15}$$

which is also the limiting distribution, provided that the chain is irreducible and aperiodic. In particular, by ignoring the y-coordinate, we see that the limiting pdf of $\mathbf{X}_n$ is the required target $f(\mathbf{x})$. This leads to the following *generalized Markov sampler* [11].

Algorithm 6.7.1 (Generalized Markov Sampler) Starting with an arbitrary $(\mathbf{X}_0,\mathbf{Y}_0)$, perform the following steps iteratively:

[*Q*-step:] Given $(\mathbf{X}_n,\mathbf{Y}_n)$, generate $\mathbf{Y}$ from $\mathbf{Q}_{\mathbf{x}}(\mathbf{Y}_n,\mathbf{y})$.

[*R*-step:] Given $\mathbf{Y}$ generate $(\mathbf{X}_{n+1},\mathbf{Y}_{n+1})$ from $\mathbf{R}[(\mathbf{X}_n,\mathbf{Y}),(\mathbf{x},\mathbf{y})]$.

Remark 6.7.1 Denoting $\mathscr{R}^-(\mathbf{x},\mathbf{y}) = \mathscr{R}(\mathbf{x},\mathbf{y}) \setminus \{(\mathbf{x},\mathbf{y})\}$, the sampler can be generalized further (see [11]) by redefining $\mathbf{R}$ as

$$\mathbf{R}[(\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')] = \begin{cases} s(\mathbf{x},\mathbf{y})\, c(\mathbf{x},\mathbf{y})\, f(\mathbf{x}')\, q_{\mathbf{x}'}(\mathbf{y}') & \text{if } (\mathbf{x}',\mathbf{y}') \in \mathscr{R}^-(\mathbf{x},\mathbf{y}) \\ 1 - \displaystyle\sum_{(\mathbf{z},\mathbf{k})\in\mathscr{R}^-(\mathbf{x},\mathbf{y})} \mathbf{R}[(\mathbf{x},\mathbf{y}),(\mathbf{z},\mathbf{k})] & \text{if } (\mathbf{x}',\mathbf{y}') = (\mathbf{x},\mathbf{y}) \;, \end{cases} \tag{6.16}$$

where s is an arbitrary function such that, first, $s(\mathbf{x},\mathbf{y}) = s(\mathbf{x}',\mathbf{y}')$ for all $(\mathbf{x}',\mathbf{y}') \in \mathscr{R}(\mathbf{x},\mathbf{y})$ and, second, the quantities above are indeed probabilities.

The generalized Markov sampler framework makes it possible to obtain many different samplers in a simple and unified manner. We give two examples: the slice sampler and the reversible jump sampler.

6.7.1 Slice Sampler

Suppose we wish to generate samples from the pdf

$$f(\mathbf{x}) = b \prod_{k=1}^{m} f_k(\mathbf{x}) \;, \tag{6.17}$$

where b is a known or unknown constant and the $\{f_k\}$ are known positive functions — not necessarily densities. We employ Algorithm 6.7.1, where at the Q-step we generate, for a given $\mathbf{X} = \mathbf{x}$, a vector $\mathbf{Y} = (Y_1,\ldots,Y_m)$ by independently drawing each component Y_k from the uniform distribution on $[0, f_k(\mathbf{x})]$. Thus, $q_{\mathbf{x}}(\mathbf{y}) = 1/\prod_{k=1}^m f_k(\mathbf{x}) = b/f(\mathbf{x})$. Second, we let $\mathscr{R}(\mathbf{x},\mathbf{y}) = \{(\mathbf{x}',\mathbf{y}) : f_k(\mathbf{x}') \geqslant y_k,\ k = 1,\ldots,m\}$. Then, (note that $f(\mathbf{x}')\, q_{\mathbf{x}'}(\mathbf{y}) = b$)

$$\mathbf{R}[(\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y})] = \frac{1}{|\mathscr{R}(\mathbf{x},\mathbf{y})|} \;.$$

In other words, in the R-step, given $\mathbf{x}$ and $\mathbf{y}$, we draw $\mathbf{X}'$ uniformly from the set $\{\mathbf{x}' : f_k(\mathbf{x}') \geqslant y_k, k = 1,\ldots,m\}$. This gives the following *slice sampler*.

Algorithm 6.7.2 (Slice Sampler)

Let $f(\mathbf{x})$ *be of the form* (6.17).

1. *Initialize* $\mathbf{X}_1$. *Set* $t = 1$.
2. *For* $k = 1,\ldots,m$ *draw* $U_k \sim \mathsf{U}(0,1)$ *and let* $Y_k = U_k\, f_k(\mathbf{X}_t)$.
3. *Draw* $\mathbf{X}_{t+1}$ *uniformly from the set* $\{\mathbf{x} : f_k(\mathbf{x}) \geqslant Y_k, k = 1,\ldots,m\}$.
4. *Stop if a stopping criterion is met; otherwise, set* $t = t+1$ *and repeat from Step 2.*

■ **EXAMPLE 6.9 Slice Sampler**

Suppose we want to generate a sample from the target pdf

$$f(x) = c\,\frac{x\,\mathrm{e}^{-x}}{1+x}, \quad x \geqslant 0$$

using the slice sampler with $f_1(x) = x/(1+x)$ and $f_2(x) = \mathrm{e}^{-x}$.

Suppose that at iteration t, $X_{t-1} = z$, and u_1 and u_2 are generated in Step 2. In Step 3, X_t is drawn uniformly from the set $\{x : f_1(x)/f_1(z) \geqslant u_1,\ f_2(x)/f_2(z) \geqslant u_2\}$, which implies the bounds $x \geqslant \frac{u_1 z}{1+z-u_1 z}$ and $x \leqslant z - \ln u_2$. Since for $z > 0$ and $0 \leqslant u_1, u_2 \leqslant 1$, the latter bound is larger than the former, the interval to be drawn from in Step 3 is $(\frac{u_1 z}{1+z-u_1 z},\ z - \ln u_2)$. Figure 6.10 depicts a histogram of $N = 10^5$ samples generated via the slice sampler, along with the true pdf $f(x)$. We see that the two are in close agreement.

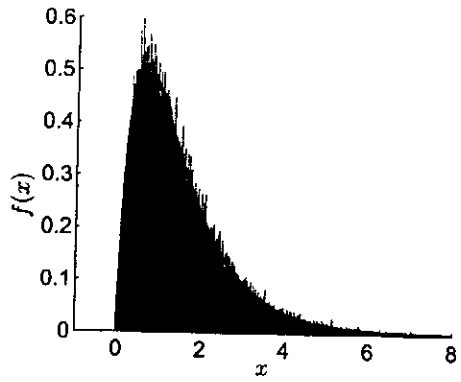

Figure 6.10 True density and histogram of samples produced by the slice sampler.

6.7.2 Reversible Jump Sampler

Reversible jump samplers [8] are useful for sampling from target spaces that contain vectors of different dimensions. This often occurs in Bayesian inference when different models for the data are considered.

■ **EXAMPLE 6.10 Regression Data**

Suppose some data $y_1, \ldots, y_n$ are the outcomes of independent random variables $\{Y_i\}$ of the form

$$Y_i = \sum_{j=0}^{M} \beta_j u_i^j + \varepsilon_i, \qquad \varepsilon_i \sim \mathsf{N}(0,1), \quad i = 1, \ldots, n\,, \tag{6.18}$$

where $u_1, \ldots, u_n$ are known variables, and $M \in \{0, \ldots, M_{\max}\}$ and the parameters $\{\beta_m\}$ are unknown. Let $\mathbf{y} = (y_1, \ldots, y_n)$ and $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_M)$. Taking uniform (i.e., constant) priors for $\{\beta_m\}$ and M, we have the joint pdf

$$f(\mathbf{y}, m, \boldsymbol{\beta}) \propto \exp\left[-\frac{1}{2}\sum_{i=1}^{n}\left(y_i - \sum_{j=0}^{m}\beta_j u_i^j\right)^2\right]. \tag{6.19}$$

Denoting $\mathbf{x} = (m, \boldsymbol{\beta})$, the objective is to draw from the posterior pdf $f(\mathbf{x}\,|\,\mathbf{y}) = f(m, \boldsymbol{\beta}\,|\,\mathbf{y})$. This yields information not only about the parameters, but also about which model (expressed by M) is more appropriate. However, note that the dimensionality of $\mathbf{x}$ depends crucially on m, so that standard Gibbs or Metropolis–Hastings sampling is not appropriate.

The reversible jump sampler jumps between spaces of different dimensionality according to a set of allowed jumps (also called *moves*). In the above example one could, for instance, allow only jumps between vectors that differ in dimension by at most 1; that is, $\beta_0 \to \beta_0'$, $\beta_0 \to (\beta_0', \beta_1')$, $(\beta_0, \beta_1) \to \beta_0'$, and so on.

To formulate the reversible jump sampler in the generalized Markov sampler framework, define $\mathscr{Y} = \mathscr{X} \times \mathscr{M}$, where $\mathscr{M}$ is the set of moves; write a generic element as $(\mathbf{z}, m)$. In

the Q-step we take $\mathbf{Q}_{\mathbf{x}}(\cdot, (\mathbf{z}, m)) = p_{\mathbf{x}}(m)\, q_m(\mathbf{x}, \mathbf{z})$. That is, a move of type m is selected according to some discrete pdf $p_{\mathbf{x}}(m)$. For example, the dimension of $\mathbf{x}$ is decreased, increased, or left unchanged. Then a new $\mathbf{z}$ is selected according to some transition function $q_m(\mathbf{x}, \mathbf{z})$. Note that the stationary pdf for the Q-step at $(\mathbf{z}, m)$ is $p_{\mathbf{x}}(m)\, q_m(\mathbf{x}, \mathbf{z})$. The R-step is determined by defining $\mathscr{R}(\mathbf{x}, (\mathbf{z}, m)) = \{(\mathbf{x}, (\mathbf{z}, m)), (\mathbf{z}, (\mathbf{x}, m'))\}$, where m' is the reverse move of m, that is, from $\mathbf{z}$ to $\mathbf{x}$. Then (6.16) reduces to

$$\mathbf{R}[(\mathbf{x}, (\mathbf{z}, m)), (\mathbf{z}, (\mathbf{x}, m'))] = \frac{s(\mathbf{x}, (\mathbf{z}, m))}{1 + 1/\varrho}\,, \tag{6.20}$$

with $\varrho = \frac{f(\mathbf{z})\, p_{\mathbf{z}}(m')\, q_{m'}(\mathbf{z}, \mathbf{x})}{f(\mathbf{x})\, p_{\mathbf{x}}(m)\, q_m(\mathbf{x}, \mathbf{z})}$. Taking $s(\mathbf{x}, (\mathbf{z}, m)) = \min\{1 + \varrho,\ 1 + 1/\varrho\}$, the right-hand side of (6.20) reduces further to $\min\{\varrho, 1\}$. The transition $(\mathbf{x}, (\mathbf{z}, m)) \to (\mathbf{z}, (\mathbf{x}, m'))$ may thus be interpreted as acceptance of the proposed element $\mathbf{z}$. In effect, $\mathbf{Q}$ is used to propose a new element in accordance with the move m and transition function q, and $\mathbf{R}$ is used to accept or reject it in accordance with the above acceptance ratio. The reversible jump sampler may thus be viewed as a generalization of the Metropolis–Hastings sampler. This gives the following algorithm (to be iterated).

Algorithm 6.7.3 (Reversible Jump Sampler)
Given the current state $\mathbf{X}_t$*:*

1. *Generate* $m \sim p_{\mathbf{X}_t}(m)$.
2. *Generate* $\mathbf{Z} \sim q_m(\mathbf{X}_t, \mathbf{z})$. *Let* m' *be the reverse move, that is, from* $\mathbf{Z}$ *to* $\mathbf{X}_t$.
3. *Generate* $U \sim \mathsf{U}(0,1)$ *and deliver*

$$\mathbf{X}_{t+1} = \begin{cases} \mathbf{Z}, & \text{if } U \le \alpha \\ \mathbf{X}_t, & \text{otherwise}\,, \end{cases} \tag{6.21}$$

where

$$\alpha = \min\left\{\frac{f(\mathbf{Z})\, p_{\mathbf{Z}}(m')\, q_{m'}(\mathbf{Z}, \mathbf{X}_t)}{f(\mathbf{X}_t)\, p_{\mathbf{X}_t}(m)\, q_m(\mathbf{X}_t, \mathbf{Z})},\ 1\right\}. \tag{6.22}$$

Remark 6.7.2 (Dimension Matching) When dealing with continuous random variables it is important to ensure that the transition densities are properly defined. Suppose that $\dim(\mathbf{x}) = d$ and $\dim(\mathbf{z}) = d' > d$. A possible way to generate a transition $\mathbf{x} \to \mathbf{z}$ is to first draw a $(d' - d)$-dimensional random vector $\mathbf{U}$ according to some density $g(\mathbf{u})$ and then let $\mathbf{z} = \phi(\mathbf{x}, \mathbf{U})$ for some bijection ϕ. This is known as *dimension matching* — the dimension of $(\mathbf{x}, \mathbf{u})$ must match that of $\mathbf{z}$. Note that by (1.20) the transition density is given by $q(\mathbf{x}, \mathbf{z}) = g(\mathbf{u})/|J_{(\mathbf{x},\mathbf{u})}(\phi)|$, where $|J_{(\mathbf{x},\mathbf{u})}(\phi)|$ is the absolute value of the determinant of the matrix of Jacobi of ϕ at $(\mathbf{x}, \mathbf{u})$.

■ **EXAMPLE 6.11 Example 6.10 (Continued)**

We illustrate the reversible jump sampler using regression data $\mathbf{y} = (y_1, \ldots, y_n)$ of the form (6.18), with $u_i = (i-1)/20$, $i = 1, \ldots, 101$, $\beta_0 = 1$, $\beta_1 = 0.3$, and $\beta_2 = -0.2$. The data are depicted in Figure 6.11. Although it is obvious that a constant model ($m = 0$) does not fit the data, it is not clear if a linear model ($m = 1$) or a quadratic model ($m = 2$) is more appropriate. To assess the different models, we

can run a reversible jump sampler to produce samples from the posterior pdf $f(\mathbf{x} \mid \mathbf{y})$, which (up to a normalization constant) is given by the right-hand side of (6.19). A very basic implementation is the following:

Procedure

1. Initialize $\mathbf{X}_1 = \mathbf{x} = (m', \boldsymbol{\beta}')$. Set $t = 1$.
2. Choose $m \in \{0, 1, 2\}$ with equal probability.
3. Generate $\boldsymbol{\beta}$ from an $(m+1)$-dimensional normal pdf g_m with independent components, with means 0 and variances σ^2. Let $\mathbf{z} = (m, \boldsymbol{\beta})$.
4. Generate $U \sim \mathsf{U}(0,1)$. If
$$U \leqslant \min\left\{\frac{f(\mathbf{z} \mid \mathbf{y})\, g_{m'}(\boldsymbol{\beta}')}{f(\mathbf{x} \mid \mathbf{y})\, g_m(\boldsymbol{\beta})}, 1\right\},$$
set $\mathbf{X}_{t+1} = \mathbf{z}$; otherwise, set $\mathbf{X}_{t+1} = \mathbf{x}$.
5. If $t = N$ stop; otherwise, set $t = t+1$, $\mathbf{x} = (m', \boldsymbol{\beta}') = \mathbf{X}_t$, and repeat from Step 2.

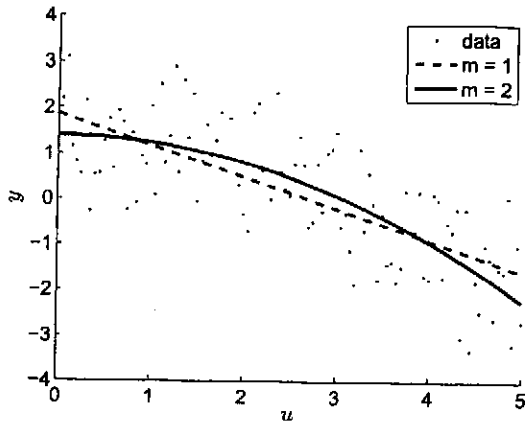

Figure 6.11 Regression data and fitted curves.

The above procedure, with $N = 10^5$ and $\sigma = 2$, produced 22,136 two-dimensional vectors $\boldsymbol{\beta}$ and 77,834 three-dimensional ones, giving posterior probabilities 0.221 and 0.778 for models 1 and 2, respectively. The posterior probability for the constant model was negligible (0.0003). This indicates that the quadratic model has the best fit. The regression parameters $\boldsymbol{\beta}$ are estimated via the sample means of the $\{\boldsymbol{\beta}_t\}$ for $m_t = 1$ or 2 and are found to be $(1.874, -0.691)$ and $(1.404, -0.011, -0.143)$. The corresponding regression curves are depicted in Figure 6.11.

6.8 SIMULATED ANNEALING

Simulated annealing is a popular optimization technique based on MCMC. This technique uses MCMC sampling to find a mode of a density $f(\mathbf{x})$ (a point $\mathbf{x}^*$ where $f(\mathbf{x})$ is maximal). It involves defining a family of densities of the form $f_T(\mathbf{x}) \propto [f(\mathbf{x})]^{1/T}$, where the parameter T is called the *temperature* of the distribution. MCMC sampling is used to draw a single element $\mathbf{X}^{(k)}$ from f_{T_k} for successively lower temperatures $T_1, T_2, \ldots$. Each element $\mathbf{X}^{(k)}$ is used as the initial element of the next chain. As the temperature is reduced, the distributions become sharply peaked at the global maxima of f. Thus, the $\{\mathbf{X}^{(k)}\}$ converge to a point. They can converge to a local maximum, but this possibility is reduced by careful selection of successive temperatures. The sequence of temperatures, or *annealing schedule*, is therefore critical to the success of the method. A common choice for the annealing schedule is a geometric progression, starting with a specified initial temperature and multiplying by a *cooling factor* in the interval $(0, 1)$ after each iteration.

Simulated annealing can also be applied to nonprobabilistic optimization problems. Given an objective function $S(\mathbf{x})$, one defines a Boltzmann distribution via the density $f(\mathbf{x}) \propto \mathrm{e}^{-S(\mathbf{x})}$ or $f(\mathbf{x}) \propto \mathrm{e}^{S(\mathbf{x})}$, depending on whether the objective is to minimize or maximize S. Global optima of S are then obtained by searching for the mode of the Boltzmann distribution. We illustrate the method via two worked examples, one based on the Metropolis–Hastings sampler and the other on the Gibbs sampler.

■ EXAMPLE 6.12 Traveling Salesman Problem

The traveling salesman problem (TSP) can be formulated as follows. Consider a weighted graph G with n nodes, labeled $1, 2, \ldots, n$. The nodes represent cities, and the edges represent the roads between the cities. Each edge from i to j has weight or cost c_{ij}, representing the length of the road. The problem is to find the shortest *tour* that visits all the cities exactly once except the starting city, which is also the terminating city. An example is given in Figure 6.12, where the bold lines form a possible tour.

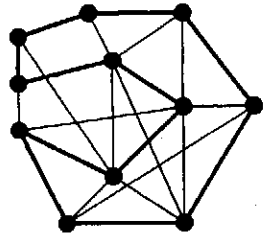

Figure 6.12 Find the shortest tour $\mathbf{x}$ visiting all nodes.

Without loss of generality, we may assume that the graph is *complete* (fully connected), because if it is not complete, we can always add some costs (distances) equal to $+\infty$. Let $\mathscr{X}$ be the set of all possible tours, and let $S(\mathbf{x})$ the total length of tour $\mathbf{x} \in \mathscr{X}$. We can represent each tour via a *permutation* of $(1, \ldots, n)$. For example, for $n = 4$, the permutation $(1, 3, 2, 4)$ represents the tour $1 \to 3 \to 2 \to 4 \to 1$. From now on, we identify a tour with its corresponding permutation. The objective

is thus to minimize

$$\min_{x\in\mathscr{X}} S(\mathbf{x}) \;=\; \min_{x\in\mathscr{X}} \left\{ \sum_{i=1}^{n-1} c_{x_i,x_{i+1}} + c_{x_n,1} \right\} . \tag{6.23}$$

Note that the number of elements in $\mathscr{X}$ is typically very large, because $|\mathscr{X}| = n!$.

The TSP can be solved via simulated annealing in the following way. First, we define the target pdf to be the Boltzmann pdf $f(\mathbf{x}) = ce^{-S(\mathbf{x})/T}$. Second, we define a neighborhood structure on the space of permutations $\mathscr{X}$ called *2-opt*. Here the neighbors of an arbitrary permutation $\mathbf{x}$ are found by (1) selecting two different indices from $\{1,\ldots,n\}$ and (2) reversing the path of $\mathbf{x}$ between those two indices. For example, if $\mathbf{x} = (1,2,\ldots,10)$ and indices 4 and 7 are selected, then $\mathbf{y} = (1,2,3,7,6,5,4,8,9,10)$; see Figure 6.13. Another example is: if $\mathbf{x} = (6,7,2,8,3,9,10,5,4,1)$ and indices 6 and 10 are selected, then $\mathbf{y} = (6,7,2,8,3,1,4,5,10,9)$.

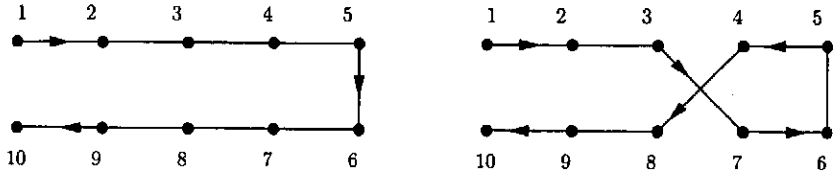

Figure 6.13 Illustration of the 2-opt neighborhood structure.

Third, we apply the Metropolis–Hastings algorithm to sample from the target. We need to supply a transition function $q(\mathbf{x},\mathbf{y})$ from $\mathbf{x}$ to one of its neighbors. Typically, the two indices for the 2-opt neighborhood are selected uniformly. This can be done, for example, by drawing a uniform permutation of $(1,\ldots,n)$ (see Section 2.8) and then selecting the first two elements of this permutation. The transition function is here constant: $q(\mathbf{x},\mathbf{y}) = q(\mathbf{y},\mathbf{x}) = 1/\binom{n}{2}$. It follows that in this case the acceptance probability is

$$\alpha = \min\left\{ \frac{f(\mathbf{y})}{f(\mathbf{x})}, 1 \right\} = \begin{cases} 1 & \text{if } S(\mathbf{y}) \leqslant S(\mathbf{x}) \\ e^{-(S(\mathbf{y})-S(\mathbf{x}))/T} & \text{if } S(\mathbf{y}) > S(\mathbf{x}) \,. \end{cases} \tag{6.24}$$

By gradually decreasing the temperature T, the Boltzmann distribution becomes more and more concentrated around the global minimizer. This leads to the following generic simulated annealing algorithm with Metropolis–Hastings sampling.

Algorithm 6.8.1 (Simulated Annealing: Metropolis–Hastings Sampling)

1. *Initialize the starting state* $\mathbf{X}_0$ *and temperature* T_0. *Set* $t = 0$.
2. *Generate a new state* $\mathbf{Y}$ *from the symmetric proposal* $q(\mathbf{X}_t,\mathbf{y})$.
3. *If* $S(\mathbf{Y}) < S(\mathbf{X}_t)$ *let* $\mathbf{X}_{t+1} = \mathbf{Y}$. *If* $S(\mathbf{Y}) \geqslant S(\mathbf{X}_t)$, *generate* $U \sim \mathrm{U}(0,1)$ *and let* $\mathbf{X}_{t+1} = \mathbf{Y}$ *if*
$$U \leqslant e^{-(S(\mathbf{Y})-S(\mathbf{X}_t))/T_t} ;$$
otherwise, let $\mathbf{X}_{t+1} = \mathbf{X}_t$.
4. *Select a new temperature* $T_{t+1} \leqslant T_t$, *increase* t *by 1, and repeat from Step 2 until stopping.*

A common choice in Step 4 is to take $T_{t+1} = \beta T_t$ for some $\beta < 1$ close to 1, such as $\beta = 0.99$.

■ **EXAMPLE 6.13** ***n*-Queens Problem**

In the n-queens problem the objective is to arrange n queens on a $n \times n$ chess board in such a way that no queen can capture another queen. An illustration is given in Figure 6.14 for the case $n = 8$. Note that the configuration in Figure 6.14 does not solve the problem. We take $n = 8$ from now on. Note that each row of the chess board must contain exactly one queen. Denote the position of the queen in the i-th row by x_i; then each configuration can be represented by a vector $\mathbf{x} = (x_1,\ldots,x_8)$. For example, $\mathbf{x} = (2,3,7,4,8,5,1,6)$ corresponds to the large configuration in Figure 6.14. Two other examples are given in the same figure. We can now formulate the problem of minimizing the function $S(\mathbf{x})$ representing the number of times the queens can capture each other. Thus $S(\mathbf{x})$ is the sum of the number of queens that can hit each other minus 1; see Figure 6.14, where $S(\mathbf{x}) = 2$ for the large configuration. Note that the minimal S value is 0. One of the optimal solutions is $\mathbf{x}^* = (5,1,8,6,3,7,2,4)$.

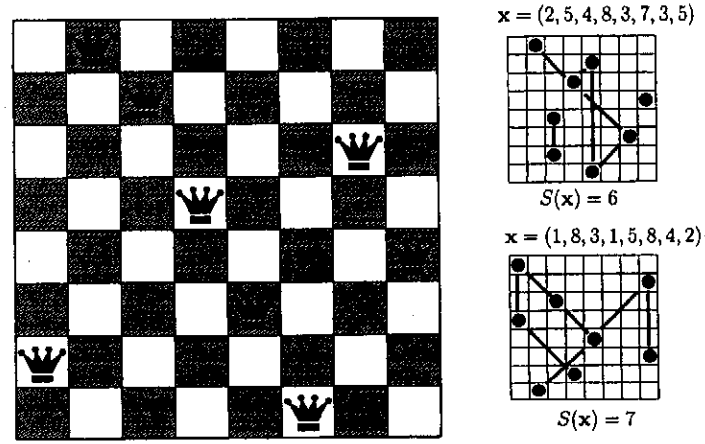

Figure 6.14 Position the eight queens such that no queen can capture another.

We show next how this optimization problem can be solved via simulated annealing using the Gibbs sampler. As in the previous TSP example, each iteration of the algorithm consists of sampling from the Boltzmann pdf $f(\mathbf{x}) = e^{-S(\mathbf{x})/T}$ via the Gibbs sampler, followed by decreasing the temperature. This leads to the following generic simulated annealing algorithm using Gibbs sampling.

Algorithm 6.8.2 (Simulated Annealing: Gibbs Sampling)

1. *Initialize the starting state* $\mathbf{X}_0$ *and temperature* T_0. *Set* $t = 0$.
2. *For a given* $\mathbf{X}_t$, *generate* $\mathbf{Y} = (Y_1, \ldots, Y_n)$ *as follows:*
   - i. *Draw* Y_1 *from the conditional pdf* $f(x_1 \mid X_{t,2}, \ldots, X_{t,n})$.
   - ii. *Draw* Y_i *from* $f(x_i \mid Y_1, \ldots, Y_{i-1}, X_{t,i+1}, \ldots, X_{t,n})$, $i = 2, \ldots, n-1$.
   - iii. *Draw* Y_n *from* $f(x_n \mid Y_1, \ldots, Y_{n-1})$.
3. *Let* $\mathbf{X}_{t+1} = \mathbf{Y}$.
4. *If* $S(\mathbf{X}_t) = 0$ *stop and display the solution; otherwise, select a new temperature* $T_{t+1} \leqslant T_t$, *increase* t *by 1, and repeat from Step 2.*

Note that in Step 2 each Y_i is drawn from a discrete distribution on $\{1, \ldots, n\}$ with probabilities proportional to $e^{-S(\mathbf{Z}_1)/T_t}, \ldots, e^{-S(\mathbf{Z}_n)/T_t}$, where each $\mathbf{Z}_k$ is equal to the vector $(Y_1, \ldots, Y_{i-1}, k, X_{t,i+1}, \ldots, X_{t,n})$.

Other MCMC samplers can be used in simulated annealing. For example, in the *hide-and-seek* algorithm [20] the general hit-and-run sampler (Section 6.3) is used. Research motivated by the use of hit-and-run and discrete hit-and-run in simulated annealing, has resulted in the development of a theoretically derived cooling schedule that uses the recorded values obtained during the course of the algorithm to adaptively update the temperature [22, 23].

6.9 PERFECT SAMPLING

Returning to the beginning of this chapter, suppose that we wish to generate a random variable X taking values in $\{1, \ldots, m\}$ according to a target distribution $\pi = \{\pi_i\}$. As mentioned, one of the main drawbacks of the MCMC method is that each sample X_t is only *asymptotically* distributed according to π, that is, $\lim_{t\to\infty} \mathbb{P}(X_t = i) = \pi_i$. In contrast, *perfect sampling* is an MCMC technique that produces exact samples from π.

Let $\{X_t\}$ be a Markov chain with state space $\{1, \ldots, m\}$, transition matrix P, and stationary distribution π. We wish to generate the $\{X_t,\ t = 0, -1, -2, \ldots\}$ in such a way that X_0 has the desired distribution. We can draw X_0 from the m-point distribution corresponding to the X_{-1}-th row of P, see Algorithm 2.7.1. This can be done via the IT method, which requires the generation of a random variable $U_0 \sim \mathsf{U}(0,1)$. Similarly, X_{-1} can be generated from X_{-2} and $U_{-1} \sim \mathsf{U}(0,1)$. In general, we see that for any negative time $-t$ the random variable X_0 depends on X_{-t} and the independent random variables $U_{-t+1}, \ldots, U_0 \sim \mathsf{U}(0,1)$.

Next, consider m dependent copies of the Markov chain, starting from each of the states $1, \ldots, m$ and using the *same* random numbers $\{U_i\}$ — similar to the CRV method. Then, if two paths coincide, or *coalesce*, at some time, from that time on, both paths will be identical.

The paths are said to be *coupled*. The main point of the perfect sampling method is that if the chain is ergodic (in particular, if it is aperiodic and irreducible), then *with probability 1 there exists a negative time* $-T$ *such that all* m *paths will have coalesced before or at time 0*. The situation is illustrated in Figure 6.15.

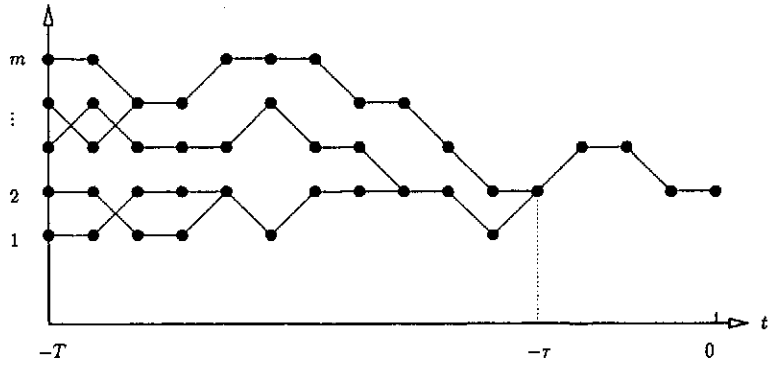

Figure 6.15 All Markov chains have coalesced at time $-\tau$.

Let $\mathbf{U}$ represent the vector of all $U_t, t \leqslant 0$. For each $\mathbf{U}$ we know there exists, with probability 1, a $-T(\mathbf{U}) < 0$ such that by time 0 all m coupled chains defined by $\mathbf{U}$ have coalesced. Moreover, if we start at time $-T$ a *stationary* version of the Markov chain, using again the same $\mathbf{U}$, this stationary chain must, at time $t = 0$, have coalesced with the other ones. Thus, any of the m chains has at time 0 the same distribution as the stationary chain, which is π.

Note that in order to construct T we do not need to know the whole (infinite vector) $\mathbf{U}$. Instead, we can work backward from $t = 0$ by generating U_{-1} first, and checking if $-T = -1$. If this is not the case, generate U_{-2} and check if $-T = -2$, and so on. This leads to the following algorithm, due to Propp and Wilson [18], called *coupling from the past*.

Algorithm 6.9.1 (Coupling from the Past)

1. *Generate* $U_0 \sim \mathsf{U}(0,1)$. *Set* $\mathbf{U}_0 = U_0$. *Set* $t = -1$.
2. *Generate* m *Markov chains, starting at* t *from each of the states* $1, \ldots, m$, *using the same random vector* $\mathbf{U}_{t+1}$.
3. *Check if all chains have coalesced before or at time 0. If so, return the common value of the chains at time 0 and stop; otherwise, generate* $U_t \sim \mathsf{U}(0,1)$, *let* $\mathbf{U}_t = (U_t, \mathbf{U}_{t+1})$, *set* $t = t - 1$, *and repeat from Step 2.*

Although perfect sampling seems indeed perfect in that it returns an exact sample from the target π rather than an approximate one, practical applications of the technique are, presently, quite limited. Not only is the technique difficult or impossible to use for most continuous simulation systems, it is also much more computationally intensive than simple MCMC.

PROBLEMS

6.1 Verify that the local balance equation (6.3) holds for the Metropolis–Hastings algorithm.

6.2 When running an MCMC algorithm, it is important to know when the transient (or *burn-in*) period has finished; otherwise, steady-state statistical analyses such as those in Section 4.3.2 may not be applicable. In practice this is often done via a visual inspection of the sample path. As an example, run the random walk sampler with normal target distribution $\mathsf{N}(10, 1)$ and proposal $Y \sim \mathsf{N}(x, 0.01)$. Take a sample size of $N = 5000$. Determine roughly when the process reaches stationarity.

6.3 A useful tool for examining the behavior of a stationary process $\{X_t\}$ obtained, for example, from an MCMC simulation, is the covariance function $R(t) = \mathrm{Cov}(X_t, X_0)$; see Example 6.4. Estimate the covariance function for the process in Problem 6.2 and plot the results. In Matlab's *signal processing* toolbox this is implemented under the M-function `xcov.m`. Try different proposal distributions of the form $\mathsf{N}(x, \sigma^2)$ and observe how the covariance function changes.

6.4 Implement the independence sampler with an $\mathsf{Exp}(1)$ target and an $\mathsf{Exp}(\lambda)$ proposal distribution for several values of λ. Similar to the importance sampling situation, things go awry when the sampling distribution gets too far from the target distribution, in this case when $\lambda > 2$. For each run, use a sample size of 10^5 and start with $x = 1$.

a) For each value $\lambda = 0.2, 1, 2$, and 5, plot a histogram of the data and compare it with the true pdf.

b) For each value of the above values of λ, calculate the sample mean and repeat this for 20 independent runs. Make a dotplot of the data (plot them on a line) and notice the differences. Observe that for $\lambda = 5$ most of the sample means are below 1, and thus underestimate the true expectation 1, but a few are significantly greater. Observe also the behavior of the corresponding auto-covariance functions, both between the different λs and, for $\lambda = 5$, within the 20 runs.

6.5 Implement the random walk sampler with an $\mathsf{Exp}(1)$ target distribution, where Z (in the proposal $Y = x + Z$) has a double exponential distribution with parameter λ. Carry out a study similar to that in Problem 6.4 for different values of λ, say $\lambda = 0.1, 1, 5, 20$. Observe that (in this case) the random walk sampler has a more stable behavior than the independence sampler.

6.6 Let $\mathbf{X} = (X, Y)^T$ be a random column vector with a bivariate normal distribution with expectation vector $\mathbf{0} = (0, 0)^T$ and covariance matrix

$$\Sigma = \begin{pmatrix} 1 & \varrho \\ \varrho & 1 \end{pmatrix} .$$

a) Show that $(Y \mid X = x) \sim \mathsf{N}(\varrho x, 1 - \varrho^2)$ and $(X \mid Y = y) \sim \mathsf{N}(\varrho y, 1 - \varrho^2)$.

b) Write a systematic Gibbs sampler to draw 10^4 samples from the bivariate distribution $\mathsf{N}(\mathbf{0}, \Sigma)$ and plot the data for $\varrho = 0, 0.7$ and 0.9.

6.7 A remarkable feature of the Gibbs sampler is that the conditional distributions in Algorithm 6.4.1 contain sufficient information to generate a sample from the joint one. The following result (by Hammersley and Clifford [9]) shows that it is possible to directly express the joint pdf in terms of the conditional ones. Namely,

$$f(x, y) = \frac{f_{Y|X}(y|x)}{\int \frac{f_{Y|X}(y|x)}{f_{X|Y}(x|y)}\, dy} .$$

Prove this. Generalize this to the n-dimensional case.

6.8 In the Ising model the *expected magnetization per spin* is given by

$$M(T) = \frac{1}{n^2}\, \mathbb{E}_{\pi_T} \left[\sum_i S_i \right] ,$$

where π_T is the Boltzmann distribution at temperature T. Estimate $M(T)$, for example via the Swendsen–Wang algorithm, for various values of $T \in [0, 5]$, and observe that the graph of $M(T)$ changes sharply around the critical temperature $T \approx 2.61$. Take $n = 20$ and use periodic boundaries.

6.9 Run Peter Young's Java applet in

`http://bartok.ucsc.edu/peter/java/ising/keep/ising.html`

to gain a better understanding of how the Ising model works.

6.10 As in Example 6.6, let $\mathscr{X}^* = \{\mathbf{x} : \sum_{i=1}^n x_i = m, x_i \in \{0, \ldots, m\}, i = 1, \ldots, n\}$. Show that this set has $\binom{m+n-1}{n-1}$ elements.

6.11 In a simple model for a closed queueing network with n queues and m customers, it is assumed that the service times are independent and exponentially distributed, say with rate μ_i for queue i, $i = 1, \ldots, n$. After completing service at queue i, the customer moves to queue j with probability p_{ij}. The $\{p_{ij}\}$ are the so-called *routing probabilities*.

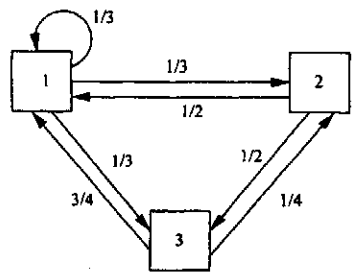

Figure 6.16 A closed queueing network.

It can be shown (see, for example, [12]) that the stationary distribution of the number of customers in the queues is of product form (6.10), with f_i being the pdf of the $\mathsf{G}(1 - y_i/\mu_i)$ distribution; thus, $f_i(x_i) \propto (y_i/\mu_i)^{x_i}$. Here the $\{y_i\}$ are constants that are obtained from the following set of *flow balance* equations:

$$y_i = \sum_j y_j\, p_{ji} , \quad i = 1, \ldots, n , \tag{6.25}$$

which has a one-dimensional solution space. Without loss of generality, y_1 can be set to 1 to obtain a unique solution.

Consider now the specific case of the network depicted in Figure 6.16, with $n = 3$ queues. Suppose the service rates are $\mu_1 = 2$, $\mu_2 = 1$, and $\mu_3 = 1$. The routing probabilities are given in the figure.

a) Show that a solution to (6.25) is $(y_1, y_2, y_3) = (1, 10/21, 4/7)$.

b) For $m = 50$ determine the exact normalization constant C.

c) Implement the procedure of Example 6.6 to estimate C via MCMC and compare the estimate for $m = 50$ with the exact value.

6.12 Let $X_1, \ldots, X_n$ be a random sample from the $\mathsf{N}(\mu, \sigma^2)$ distribution. Consider the following Bayesian model:

- $f(\mu, \sigma^2) = 1/\sigma^2$;
- $(\mathrm{x}_i \,|\, \mu, \sigma) \sim \mathsf{N}(\mu, \sigma^2)$, $i = 1, \ldots, n$ independently.

Note that the prior for (μ, σ^2) is *improper*. That is, it is not a pdf in itself, but by obstinately applying Bayes' formula, it does yield a proper posterior pdf. In some sense it conveys the least amount of information about μ and σ^2. Let $\mathbf{x} = (x_1, \ldots, x_n)$ represent the data. The posterior pdf is given by

$$f(\mu, \sigma^2 \,|\, \mathbf{x}) = \left(2\pi\sigma^2\right)^{-n/2} \exp\left\{-\frac{1}{2}\frac{\sum_i (x_i - \mu)^2}{\sigma^2}\right\} \frac{1}{\sigma^2} .$$

We wish to sample from this distribution via the Gibbs sampler.

a) Show that $(\mu \,|\, \sigma^2, \mathbf{x}) \sim \mathsf{N}(\bar{x},\, \sigma^2/n)$, where $\bar{x}$ is the sample mean.

b) Prove that

$$f(\sigma^2 \,|\, \mu, \mathbf{x}) \propto \frac{1}{(\sigma^2)^{n/2+1}} \exp\left(-\frac{n}{2}\frac{V_\mu}{\sigma^2}\right), \tag{6.26}$$

where $V_\mu = \sum_i (x_i - \mu)^2/n$ is the classical sample variance for known μ. In other words, $(1/\sigma^2 \,|\, \mu, \mathbf{x}) \sim \mathsf{Gamma}(n/2, nV_\mu/2)$.

c) Implement a Gibbs sampler to sample from the posterior distribution, taking $n = 100$. Run the sampler for 10^5 iterations. Plot the histograms of $f(\mu \,|\, \mathbf{x})$ and $f(\sigma^2 \,|\, \mathbf{x})$ and find the sample means of these posteriors. Compare them with the classical estimates.

d) Show that the true posterior pdf of μ given the data is given by

$$f(\mu \,|\, \mathbf{x}) \propto \left((\mu - \bar{x})^2 + V\right)^{-n/2},$$

where $V = \sum_i (x_i - \bar{x})^2/n$. (Hint: in order to evaluate the integral

$$f(\mu \,|\, \mathbf{x}) = \int_0^\infty f(\mu, \sigma^2 \,|\, \mathbf{x})\, d\sigma^2$$

write it first as $(2\pi)^{-n/2} \int_0^\infty t^{n/2-1} \exp(-\frac{1}{2}\, t\, c)\, dt$, where $c = n\, V_\mu$, by applying the change of variable $t = 1/\sigma^2$. Show that the latter integral is proportional to $c^{-n/2}$. Finally, apply the decomposition $V_\mu = (\bar{x} - \mu)^2 + V$.)

6.13 Suppose $f(\theta \,|\, \mathbf{x})$ is the posterior pdf for some Bayesian estimation problem. For example, θ could represent the parameters of a regression model based on the data $\mathbf{x}$. An important use for the posterior pdf is to make predictions about the distribution of other

random variables. For example, suppose the pdf of some random variable Y depends on θ via the conditional pdf $f(\mathrm{y} \,|\, \theta)$. The *predictive pdf* of Y given $\mathbf{x}$ is defined as

$$f(\mathrm{y} \,|\, \mathbf{x}) = \int f(\mathrm{y} \,|\, \theta) f(\theta \,|\, \mathbf{x})\, d\theta ,$$

which can be viewed as the expectation of $f(\mathrm{y} \,|\, \theta)$ under the posterior pdf. Therefore, we can use Monte Carlo simulation to approximate $f(\mathrm{y} \,|\, \mathbf{x})$ as

$$f(\mathrm{y} \,|\, \mathbf{x}) \approx \frac{1}{N} \sum_{i=1}^{N} f(\mathrm{y} \,|\, \theta_i) ,$$

where the sample $\{\theta_i, i = 1, \ldots, N\}$ is obtained from $f(\theta \,|\, \mathbf{x})$; for example, via MCMC.

As a concrete application, suppose that the independent measurement data: $-0.4326, -1.6656, 0.1253, 0.2877, -1.1465$ come from some $\mathsf{N}(\mu, \sigma^2)$ distribution. Define $\theta = (\mu, \sigma^2)$. Let $Y \sim \mathsf{N}(\mu, \sigma^2)$ be a new measurement. Estimate and draw the predictive pdf $f(y \,|\, \mathbf{x})$ from a sample $\theta_1, \ldots, \theta_N$ obtained via the Gibbs sampler of Problem 6.12. Take $N = 10{,}000$. Compare this with the "common-sense" Gaussian pdf with expectation $\bar{x}$ (sample mean) and variance s^2 (sample variance).

6.14 In the *zero-inflated Poisson* (ZIP) model, random data $X_1, \ldots, X_n$ are assumed to be of the form $X_i = R_i\, Y_i$, where the $\{Y_i\}$ have a $\mathsf{Poi}(\lambda)$ distribution and the $\{R_i\}$ have a $\mathsf{Ber}(p)$ distribution, all independent of each other. Given an outcome $\mathbf{x} = (x_1, \ldots, x_n)$, the objective is to estimate both λ and p. Consider the following hierarchical Bayes model:

- $p \sim \mathsf{U}(0, 1)$ (prior for p),
- $(\lambda \,|\, p) \sim \mathsf{Gamma}(a, b)$ (prior for λ),
- $(r_i \,|\, p, \lambda) \sim \mathsf{Ber}(p)$ independently (from the model above),
- $(x_i \,|\, \mathbf{r}, \lambda, p) \sim \mathsf{Poi}(\lambda r_i)$ independently (from the model above),

where $\mathbf{r} = (r_1, \ldots, r_n)$ and a and b are known parameters. It follows that

$$f(\mathbf{x}, \mathbf{r}, \lambda, p) = \frac{b^a \lambda^{a-1} \mathrm{e}^{-b\lambda}}{\Gamma(a)} \prod_{i=1}^{n} \frac{\mathrm{e}^{-\lambda r_i} (\lambda r_i)^{x_i}}{x_i!}\, p^{r_i} (1-p)^{1-r_i} .$$

We wish to sample from the posterior pdf $f(\lambda, p, \mathbf{r} \,|\, \mathbf{x})$ using the Gibbs sampler.

a) Show that

1. $(\lambda \,|\, p, \mathbf{r}, \mathbf{x}) \sim \mathsf{Gamma}(a + \sum_i x_i,\ b + \sum_i r_i)$.
2. $(p \,|\, \lambda, \mathbf{r}, \mathbf{x}) \sim \mathsf{Beta}(1 + \sum_i r_i,\ n + 1 - \sum_i r_i)$.
3. $(r_i \,|\, \lambda, p, \mathbf{x}) \sim \mathsf{Ber}\left(\frac{p\mathrm{e}^{-\lambda}}{p\mathrm{e}^{-\lambda} + (1-p) I_{\{x_i = 0\}}}\right)$.

b) Generate a random sample of size $n = 100$ for the ZIP model using parameters $p = 0.3$ and $\lambda = 2$.

c) Implement the Gibbs sampler, generate a large (dependent) sample from the posterior distribution and use this to construct 95% Bayesian CIs for p and λ using the data in b). Compare these with the true values.

6.15 * Show that μ in (6.15) satisfies the local balance equations

$$\mu(\mathbf{x}, \mathrm{y})\, \mathbf{R}[(\mathbf{x}, \mathrm{y}), (\mathbf{x}', \mathrm{y}')] = \mu(\mathbf{x}', \mathrm{y}')\, \mathbf{R}[(\mathbf{x}', \mathrm{y}'), (\mathbf{x}, \mathrm{y})] .$$

Thus μ is stationary with respect to $\mathbf{R}$, that is, $\mu\mathbf{R} = \mu$. Show that μ is also stationary with respect to $\mathbf{Q}$. Show, finally, that μ is stationary with respect to $\mathbf{P} = \mathbf{QR}$.

6.16 * This is to show that the systematic Gibbs sampler is a special case of the generalized Markov sampler. Take $\mathscr{Y}$ to be the set of indices $\{1, \ldots, n\}$, and define for the Q-step

$$\mathbf{Q}_{\mathbf{x}}(y, y') = \begin{cases} 1 & \text{if } y' = y + 1 \text{ or } y' = 1, y = n \\ 0 & \text{otherwise.} \end{cases}$$

Let the set of possible transitions $\mathscr{R}(\mathbf{x}, y)$ be the set of vectors $\{(\mathbf{x}', y)\}$ such that all coordinates of $\mathbf{x}'$ are the same as those of $\mathbf{x}$ except for possibly the y-th coordinate.

a) Show that the stationary distribution of $\mathbf{Q}_{\mathbf{x}}$ is $q_{\mathbf{x}}(y) = 1/n$, for $y = 1, \ldots, n$.

b) Show that

$$\mathbf{R}[(\mathbf{x}, y), (\mathbf{x}', y)] = \frac{f(\mathbf{x}')}{\sum_{(\mathbf{z}, y) \in \mathscr{R}(\mathbf{x}, y)} f(\mathbf{z})}, \quad \text{for } (\mathbf{x}', y) \in \mathscr{R}(\mathbf{x}, y) .$$

c) Compare with Algorithm 6.4.1.

6.17 * Prove that the Metropolis–Hastings algorithm is a special case of the generalized Markov sampler. (Hint: let the auxiliary set $\mathscr{Y}$ be a copy of the target set $\mathscr{X}$, let $\mathbf{Q}_{\mathbf{x}}$ correspond to the transition function of the Metropolis–Hastings algorithm (that is, $\mathbf{Q}_{\mathbf{x}}(\cdot, \mathbf{y}) = q(\mathbf{x}, \mathbf{y})$), and define $\mathscr{R}(\mathbf{x}, \mathbf{y}) = \{(\mathbf{x}, \mathbf{y}), (\mathbf{y}, \mathbf{x})\}$. Use arguments similar to those for the Markov jump sampler (see (6.20)) to complete the proof.)

6.18 * Barker's and Hastings' MCMC algorithms differ from the symmetric Metropolis sampler only in that they define the acceptance ratio $\alpha(\mathbf{x}, \mathbf{y})$ to be respectively $f(\mathbf{y})/(f(\mathbf{x}) + f(\mathbf{y}))$ and $s(\mathbf{x}, \mathbf{y})/(1 + 1/\varrho(\mathbf{x}, \mathbf{y}))$ instead of $\min\{f(\mathbf{y})/f(\mathbf{x}), 1\}$. Here $\varrho(\mathbf{x}, \mathbf{y})$ is defined in (6.6) and s is any symmetric function such that $0 \leqslant \alpha(\mathbf{x}, \mathbf{y}) \leqslant 1$. Show that both are special cases of the generalized Markov sampler. (Hint: take $\mathscr{Y} = \mathscr{X}$.)

6.19 Implement the simulated annealing algorithm for the n-queens problem suggested in Example 6.13. How many solutions can you find?

6.20 Implement the Metropolis–Hastings based simulated annealing algorithm for the TSP in Example 6.12. Run the algorithm on some test problems in

`http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`

6.21 Write a simulated annealing algorithm based on the random walk sampler to maximize the function

$$S(x) = \left| \frac{\sin^8(10x) + \cos^5(5x + 1)}{x^2 - x + 1} \right|, \quad x \in \mathbb{R} .$$

Use a $\mathrm{N}(x, \sigma^2)$ proposal function, given the current state x. Start with $x = 0$. Plot the current best function value against the number of evaluations of S for various values of σ and various annealing schedules. Repeat the experiments several times to assess what works best.

Further Reading

MCMC is one of the principal tools of statistical computing and Bayesian analysis. A comprehensive discussion of MCMC techniques can be found in [19], and practical applications are discussed in [7]. For more details on the use of MCMC in Bayesian analysis, we refer to [5]. A classical reference on simulated annealing is [1]. More general global search algorithms may be found in [25]. An influential paper on stationarity detection in Markov chains, which is closely related to perfect sampling, is [3].

REFERENCES

1. E. H. L. Aarts and J. H. M. Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons, Chichester, 1989.
2. D. J. Aldous and J. Fill. *Reversible Markov Chains and Random Walks on Graphs*. In preparation. `http://www.stat.berkeley.edu /users/aldous/book.html`, 2007.
3. S. Asmussen, P. W. Glynn, and H. Thorisson. Stationary detection in the initial transient problem. *ACM Transactions on Modeling and Computer Simulation*, 2(2):130–157, 1992.
4. S. Baumert, A. Ghate, S Kiatsupaibul, Y. Shen, R. L. Smith, and Z. B. Zabinsky. A discrete hit-and-run algorithm for generating multivariate distributions over arbitrary finite subsets of a lattice. Technical report, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, 2006.
5. A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall, New York, 2nd edition, 2003.
6. S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on PAMI*, 6:721–741, 1984.
7. W.R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, New York, 1996.
8. P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.
9. J. Hammersley and M. Clifford. Markov fields on finite graphs and lattices. Unpublished manuscript, 1970.
10. W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:92–109, 1970.
11. J. M. Keith, D. P. Kroese, and D. Bryant. A generalized Markov chain sampler. *Methodology and Computing in Applied Probability*, 6(1):29–53, 2004.
12. F. P. Kelly. *Reversibility and Stochastic Networks*. Wiley, Chichester, 1979.
13. J. S. Liu. *Monte Carlo Strategies in Scientifi c Computing*. Springer-Verlag, New York, 2001.
14. L. Lovász. Hit-and-run mixes fast. *Mathematical Programming*, 86:443–461, 1999.
15. L. Lovász and S. S. Vempala. Hit-and-run is fast and fun. Technical report, Microsoft Research, SMS-TR, 2003.
16. L. Lovász and S. Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006.
17. M. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.
18. J. G. Propp and D. B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 1 & 2:223–252, 1996.
19. C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, New York, 2nd edition, 2004.

20. H. E. Romeijn and R. L. Smith. Simulated annealing for constrained global optimization. *Journal of Global Optimization*, 5:101–126, 1994.

21. S. M. Ross. *Simulation*. Academic Press, New York, 3rd edition, 2002.

22. Y. Shen. *Annealing Adaptive Search with Hit-and-Run Sampling Methods for Stochastic Global Optimization Algorithms*. PhD thesis, University of Washington, 2005.

23. Y. Shen, S. Kiatsupaibul, Z. B. Zabinsky, and R. L. Smith. An analytically derived cooling schedule for simulated annealing. *Journal of Global Optimization*, 38(3):333–365, 2007.

24. R. L. Smith. Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32:1296–1308, 1984.

25. Z. B. Zabinsky. *Stochastic Adaptive Search for Global Optimization*. Kluwer Academic Publishers, Dordrecht, 2003.

26. Z. B. Zabinsky, R. L. Smith, J. F. McDonald, H. E. Romeijn, and D. E. Kaufman. Improving hit-and-run for global optimization. *Journal of Global Optimization*, 3:171–192, 1993.

CHAPTER 7

SENSITIVITY ANALYSIS AND MONTE CARLO OPTIMIZATION

7.1 INTRODUCTION

As discussed in Chapter 3, many real-world complex systems in science and engineering can be modeled as *discrete-event systems*. The behavior of such systems is identified via a sequence of discrete events, which causes the system to change from one state to another. Examples include traffic systems, flexible manufacturing systems, computer-communications systems, inventory systems, production lines, coherent lifetime systems, PERT networks, and flow networks. A discrete-event system can be classified as either *static* or *dynamic*. The former are called *discrete-event static systems* (DESS), while the latter are called *discrete-event dynamic systems* (DEDS). The main difference is that DESS do not evolve over time, while DEDS do. The PERT network is a typical example of a DESS, with the sample performance being, for example, the shortest path in the network. A queueing network, such as the Jackson network in Section 3.3.1, is an example of a DEDS, with the sample performance being, for example, the delay (waiting time of a customer) in the network. In this chapter we shall deal mainly with DESS. For a comprehensive study of both DESS and DEDS the reader is referred to [11], [16], and [20].

Because of their complexity, the performance evaluation of discrete-event systems is usually studied by simulation, and it is often associated with the estimation of the performance or response function $\ell(\mathbf{u}) = \mathbf{E}_{\mathbf{u}}[H(\mathbf{X})]$, where the distribution of the sample performance $H(\mathbf{X})$ depends on the control or reference parameter $\mathbf{u} \in \mathscr{V}$. *Sensitivity analysis* is concerned with evaluating sensitivities (gradients, Hessians, etc.) of the response function $\ell(\mathbf{u})$ with respect to parameter vector $\mathbf{u}$, and it is based on the score function and the Fisher infor-

Simulation and the Monte Carlo Method, Second Edition. By R.Y. Rubinstein and D. P. Kroese