INTRODUCTION TO MONTE CARLO METHODS

D.J.C. MACKAY
Department of Physics, Cambridge University.
Cavendish Laboratory, Madingley Road,
Cambridge, CB3 0HE. United Kingdom.

ABSTRACT

This chapter describes a sequence of Monte Carlo methods: **importance sampling**, **rejection sampling**, the **Metropolis method**, and **Gibbs sampling**. For each method, we discuss whether the method is expected to be useful for high–dimensional problems such as arise in inference with graphical models. After the methods have been described, the terminology of Markov chain Monte Carlo methods is presented. The chapter concludes with a discussion of advanced methods, including methods for reducing random walk behaviour.

For details of Monte Carlo methods, theorems and proofs and a full list of references, the reader is directed to Neal (1993), Gilks, Richardson and Spiegelhalter (1996), and Tanner (1996).

1. The problems to be solved

The aims of Monte Carlo methods are to solve one or both of the following problems.

Problem 1: to generate samples $\{\mathbf{x}^{(r)}\}_{r=1}^{R}$ from a given probability distribution $P(\mathbf{x})$.[1]

Problem 2: to estimate expectations of functions under this distribution, for example

$$\Phi = \langle \phi(\mathbf{x}) \rangle \equiv \int d^N\mathbf{x}\, P(\mathbf{x})\phi(\mathbf{x}). \tag{1}$$

[1]Please note that I will use the word "sample" in the following sense: a sample from a distribution $P(\mathbf{x})$ is a single realization $\mathbf{x}$ whose probability distribution is $P(\mathbf{x})$. This contrasts with the alternative usage in statistics, where "sample" refers to a collection of realizations $\{\mathbf{x}\}$.

The probability distribution $P(\mathbf{x})$, which we will call the **target density**, might be a distribution from statistical physics or a conditional distribution arising in data modelling — for example, the posterior probability of a model's parameters given some observed data. We will generally assume that $\mathbf{x}$ is an N–dimensional vector with real components x_n, but we will sometimes consider discrete spaces also.

We will concentrate on the first problem (sampling), because if we have solved it, then we can solve the second problem by using the random samples $\{\mathbf{x}^{(r)}\}_{r=1}^{R}$ to give the estimator

$$\hat{\Phi} \equiv \frac{1}{R} \sum_{r} \phi(\mathbf{x}^{(r)}). \tag{2}$$

Clearly if the vectors $\{\mathbf{x}^{(r)}\}_{r=1}^{R}$ are generated from $P(\mathbf{x})$ then the expectation of $\hat{\Phi}$ is Φ. Also, as the number of samples R increases, the variance of $\hat{\Phi}$ will decrease as $\frac{\sigma^2}{R}$, where σ^2 is the variance of ϕ,

$$\sigma^2 = \int d^N\mathbf{x}\, P(\mathbf{x})(\phi(\mathbf{x}) - \Phi)^2. \tag{3}$$

This is one of the important properties of Monte Carlo methods.

> **The accuracy of the Monte Carlo estimate (equation (2)) is independent of the dimensionality of the space sampled.** To be precise, the variance of $\hat{\Phi}$ goes as $\frac{\sigma^2}{R}$. So regardless of the dimensionality of $\mathbf{x}$, it may be that as few as a dozen independent samples $\{\mathbf{x}^{(r)}\}$ suffice to estimate Φ satisfactorily.

We will find later, however, that high dimensionality can cause other difficulties for Monte Carlo methods. Obtaining independent samples from a given distribution $P(\mathbf{x})$ is often not easy.

1.1. WHY IS SAMPLING FROM $P(\mathbf{x})$ HARD?

We will assume that the density from which we wish to draw samples, $P(\mathbf{x})$, can be evaluated, at least to within a multiplicative constant; that is, we can evaluate a function $P^*(\mathbf{x})$ such that

$$P(\mathbf{x}) = P^*(\mathbf{x})/Z. \tag{4}$$

If we can evaluate $P^*(\mathbf{x})$, why can we not easily solve problem 1? Why is it in general difficult to obtain samples from $P(\mathbf{x})$? There are two difficulties. The first is that we typically do not know the normalizing constant

$$Z = \int d^N\mathbf{x}\, P^*(\mathbf{x}). \tag{5}$$

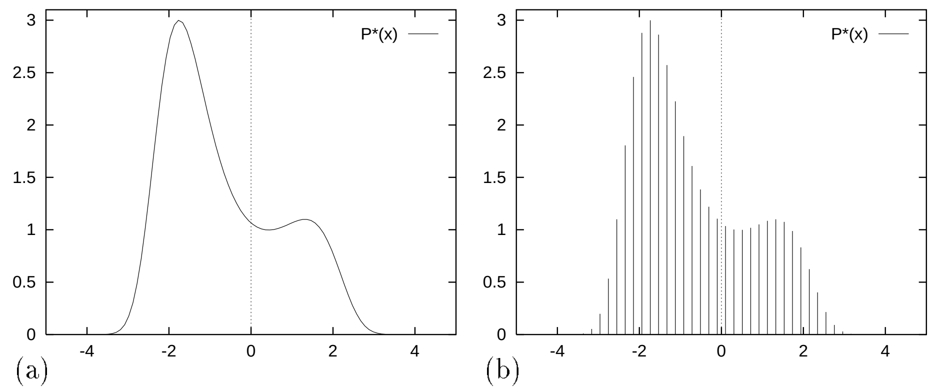

Figure 1. (a) The function $P^*(x) = \exp\left[0.4(x-0.4)^2 - 0.08x^4\right]$. How to draw samples from this density? (b) The function $P^*(x)$ evaluated at a discrete set of uniformly spaced points $\{x_i\}$. How to draw samples from this discrete distribution?

The second is that, even if we did know Z, the problem of drawing samples from $P(\mathbf{x})$ is still a challenging one, especially in high–dimensional spaces. There are only a few high–dimensional densities from which it is easy to draw samples, for example the Gaussian distribution.[2]

Let us start from a simple one–dimensional example. Imagine that we wish to draw samples from the density $P(x) = P^*(x)/Z$ where

$$P^*(x) = \exp\left[0.4(x-0.4)^2 - 0.08x^4\right], \quad x \in (-\infty, \infty). \tag{6}$$

We can plot this function (figure 1a). But that does not mean we can draw samples from it. To give ourselves a simpler problem, we could discretize the variable x and ask for samples from the discrete probability distribution over a set of uniformly spaced points $\{x_i\}$ (figure 1b). How could we solve this problem? If we evaluate $p_i^* = P^*(x_i)$ at each point x_i, we can compute

$$Z = \sum_i p_i^* \tag{7}$$

and

$$p_i = p_i^*/Z \tag{8}$$

and we can then sample from the probability distribution $\{p_i\}$ using various methods based on a source of random bits. But what is the cost of this procedure, and how does it scale with the dimensionality of the space,

[2]A sample from a univariate Gaussian can be generated by computing $\cos(2\pi u_1)\sqrt{2\log(1/u_2)}$, where u_1 and u_2 are uniformly distributed in $(0,1)$.

N? Let us concentrate on the initial cost of evaluating Z. To compute Z (equation (7)) we have to visit every point in the space. In figure 1b there are 50 uniformly spaced points in one dimension. If our system had N dimensions, $N = 1000$ say, then the corresponding number of points would be 50^{1000}, an unimaginable number of evaluations of P^*. Even if each component x_n only took two discrete values, the number of evaluations of P^* would be 2^{1000}, a number that is still horribly huge, equal to the fourth power of the number of particles in the universe.

One system with 2^{1000} states is a collection of 1000 spins, for example, a 30×30 fragment of an Ising model (or 'Boltzmann machine' or 'Markov field') (Yeomans 1992) whose probability distribution is proportional to

$$P^*(\mathbf{x}) = \exp\left[-\beta E(\mathbf{x})\right] \tag{9}$$

where $x_n \in \{\pm 1\}$ and

$$E(\mathbf{x}) = -\left[\frac{1}{2}\sum_{m,n} J_{mn} x_m x_n + \sum_n H_n x_n\right]. \tag{10}$$

The energy function $E(\mathbf{x})$ is readily evaluated for any $\mathbf{x}$. But if we wish to evaluate this function at *all* states $\mathbf{x}$, the computer time required would be 2^{1000} function evaluations.

The Ising model is a simple model which has been around for a long time, but the task of generating samples from the distribution $P(\mathbf{x}) = P^*(\mathbf{x})/Z$ is still an active research area as evidenced by the work of Propp and Wilson (1996).

1.2. UNIFORM SAMPLING

Having agreed that we cannot visit every location $\mathbf{x}$ in the state space, we might consider trying to solve the second problem (estimating the expectation of a function $\phi(\mathbf{x})$) by drawing random samples $\{\mathbf{x}^{(r)}\}_{r=1}^{R}$ *uniformly* from the state space and evaluating $P^*(\mathbf{x})$ at those points. Then we could introduce Z_R, defined by

$$Z_R = \sum_{r=1}^{R} P^*(\mathbf{x}^{(r)}), \tag{11}$$

and estimate $\Phi = \int d^N\mathbf{x}\, \phi(\mathbf{x}) P(\mathbf{x})$ by

$$\hat{\Phi} = \sum_{r=1}^{R} \phi(\mathbf{x}^{(r)}) \frac{P^*(\mathbf{x}^{(r)})}{Z_R}. \tag{12}$$

Is anything wrong with this strategy? Well, it depends on the functions $\phi(\mathbf{x})$ and $P^*(\mathbf{x})$. Let us assume that $\phi(\mathbf{x})$ is a benign, smoothly varying function and concentrate on the nature of $P^*(\mathbf{x})$. A high–dimensional distribution is often concentrated in a small region of the state space known as its typical set T, whose volume is given by $|T| \simeq 2^{H(\mathbf{X})}$, where $H(\mathbf{X})$ is the Shannon–Gibbs entropy of the probability distribution $P(\mathbf{x})$,

$$H(\mathbf{X}) = \sum_{\mathbf{x}} P(\mathbf{x}) \log_2 \frac{1}{P(\mathbf{x})}. \tag{13}$$

If almost all the probability mass is located in the typical set and $\phi(\mathbf{x})$ is a benign function, the value of $\Phi = \int d^N\mathbf{x}\ \phi(\mathbf{x})P(\mathbf{x})$ will be principally determined by the values that $\phi(\mathbf{x})$ takes on in the typical set. So uniform sampling will only stand a chance of giving a good estimate of Φ if we make the number of samples R sufficiently large that we are likely to hit the typical set a number of times. So, how many samples are required? Let us take the case of the Ising model again. The total size of the state space is 2^N states, and the typical set has size 2^H. So each sample has a chance of $2^H/2^N$ of falling in the typical set. The number of samples required to hit the typical set once is thus of order

$$R_{\text{min}} \simeq 2^{N-H}. \tag{14}$$

So, what is H? At high temperatures, the probability distribution of an Ising model tends to a uniform distribution and the entropy tends to $H_{\text{max}} = N$ bits, so R_{min} is of order 1. Under these conditions, uniform sampling may well be a satisfactory technique for estimating Φ. But high temperatures are not of great interest. Considerably more interesting are intermediate temperatures such as the critical temperature at which the Ising model melts from an ordered phase to a disordered phase. At this temperature the entropy of an Ising model is roughly $N/2$ bits. For this probability distribution the number of samples required simply to hit the typical set once is of order

$$R_{\text{min}} \simeq 2^{N-N/2} = 2^{N/2} \tag{15}$$

which for $N = 1000$ is about 10^{150}. This is roughly the square of the number of particles in the universe. Thus uniform sampling is utterly useless for the study of Ising models of modest size. And in most high–dimensional problems, if the distribution $P(\mathbf{x})$ is not actually uniform, uniform sampling is unlikely to be useful.

1.3. OVERVIEW

Having established that drawing samples from a high–dimensional distribution $P(\mathbf{x}) = P^*(\mathbf{x})/Z$ is difficult even if $P^*(\mathbf{x})$ is easy to evaluate, we will

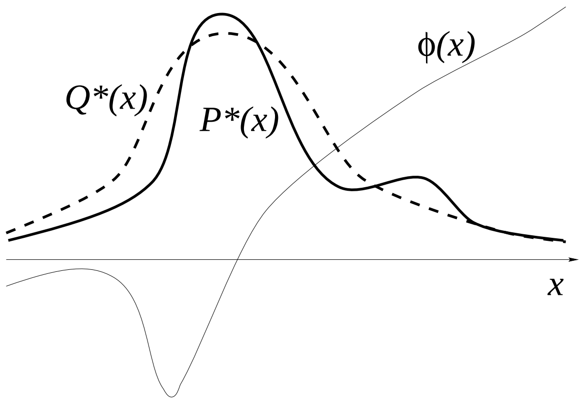

Figure 2. Functions involved in importance sampling. We wish to estimate the expectation of $\phi(x)$ under $P(x) \propto P^*(x)$. We can generate samples from the simpler distribution $Q(x) \propto Q^*(x)$. We can evaluate Q^* and P^* at any point.

now study a sequence of Monte Carlo methods: **importance sampling**, **rejection sampling**, the **Metropolis method**, and **Gibbs sampling**.

2. Importance sampling

Importance sampling is not a method for generating samples from $P(\mathbf{x})$ (problem 1); it is just a method for estimating the expectation of a function $\phi(\mathbf{x})$ (problem 2). It can be viewed as a generalization of the uniform sampling method.

For illustrative purposes, let us imagine that the target distribution is a one–dimensional density $P(x)$. It is assumed that we are able to evaluate this density, at least to within a multiplicative constant; thus we can evaluate a function $P^*(x)$ such that

$$P(x) = P^*(x)/Z. \tag{16}$$

But $P(x)$ is too complicated a function for us to be able to sample from it directly. We now assume that we have a simpler density $Q(x)$ which we can evaluate to within a multiplicative constant (that is, we can evaluate $Q^*(x)$, where $Q(x) = Q^*(x)/Z_Q$), and from which we can generate samples. An example of the functions P^*, Q^* and ϕ is shown in figure 2. We call Q the *sampler density*.

In importance sampling, we generate R samples $\{x^{(r)}\}_{r=1}^{R}$ from $Q(x)$. If these points were samples from $P(x)$ then we could estimate Φ by equation (2). But when we generate samples from Q, values of x where $Q(x)$ is greater than $P(x)$ will be over–represented in this estimator, and points

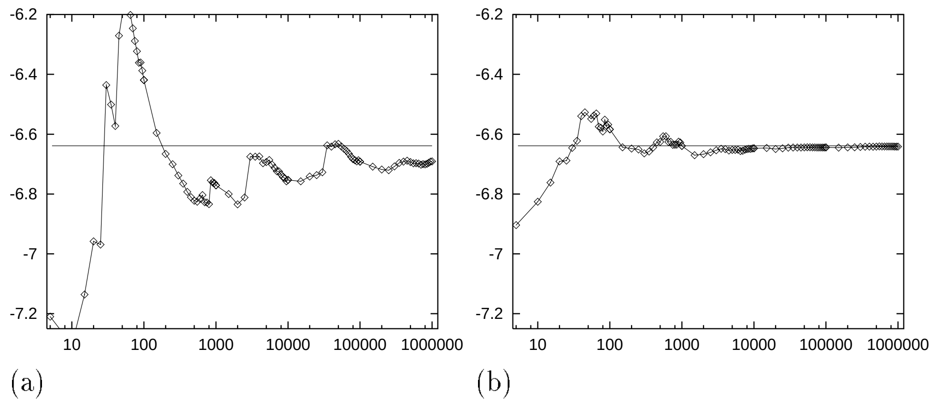

Figure 3. Importance sampling in action: a) using a Gaussian sampler density; b) using a Cauchy sampler density. Horizontal axis shows number of samples on a log scale. Vertical axis shows the estimate $\hat{\Phi}$. The horizontal line indicates the true value of Φ.

where $Q(x)$ is less than $P(x)$ will be under–represented. To take into account the fact that we have sampled from the wrong distribution, we introduce 'weights'

$$w_r \equiv \frac{P^*(x^{(r)})}{Q^*(x^{(r)})} \tag{17}$$

which we use to adjust the 'importance' of each point in our estimator thus:

$$\hat{\Phi} \equiv \frac{\sum_r w_r \phi(x^{(r)})}{\sum_r w_r}. \tag{18}$$

If $Q(x)$ is non–zero for all x where $P(x)$ is non–zero, it can be proved that the estimator $\hat{\Phi}$ converges to Φ, the mean value of $\phi(x)$, as R increases.

A practical difficulty with importance sampling is that it is hard to estimate how reliable the estimator $\hat{\Phi}$ is. The variance of $\hat{\Phi}$ is hard to estimate, because the empirical variances of the quantities w_r and $w_r\phi(x^{(r)})$ are not necessarily a good guide to the true variances of the numerator and denominator in equation (18). If the proposal density $Q(x)$ is small in a region where $|\phi(x)P^*(x)|$ is large then it is quite possible, even after many points $x^{(r)}$ have been generated, that none of them will have fallen in that region. This leads to an estimate of Φ that is drastically wrong, and no indication in the empirical variance that the true variance of the estimator $\hat{\Phi}$ is large.

2.1. CAUTIONARY ILLUSTRATION OF IMPORTANCE SAMPLING

In a toy problem related to the modelling of amino acid probability distributions with a one–dimensional variable x I evaluated a quantity of interest using importance sampling. The results using a Gaussian sampler and a Cauchy sampler are shown in figure 3. The horizontal axis shows the number of samples on a log scale. In the case of the Gaussian sampler, after about 500 samples had been evaluated one might be tempted to call a halt; but evidently there are infrequent samples that make a huge contribution to $\hat{\Phi}$, and the value of the estimate at 500 samples is wrong. Even after a million samples have been taken, the estimate has still not settled down close to the true value. In contrast, the Cauchy sampler does not suffer from glitches and converges (on the scale shown here) after about 5000 samples.

This example illustrates the fact that an importance sampler should have **heavy tails**.

2.2. IMPORTANCE SAMPLING IN MANY DIMENSIONS

We have already observed that care is needed in one–dimensional importance sampling problems. Is importance sampling a useful technique in spaces of higher dimensionality, say $N = 1000$?

Consider a simple case–study where the target density $P(\mathbf{x})$ is a uniform distribution inside a sphere,

$$P^*(\mathbf{x}) = \begin{cases} 1 & 0 \leq \rho(\mathbf{x}) \leq R_P \\ 0 & \rho(\mathbf{x}) > R_P \end{cases}, \qquad (19)$$

where $\rho(\mathbf{x}) \equiv (\sum_i x_i^2)^{1/2}$, and the proposal density is a Gaussian centred on the origin,

$$Q(\mathbf{x}) = \prod_i \text{Normal}(x_i; 0, \sigma^2). \qquad (20)$$

An importance sampling method will be in trouble if the estimator $\hat{\Phi}$ is dominated by a few large weights w_r. What will be the typical range of values of the weights w_r? By the central limit theorem, if ρ is the distance from the origin of a sample from Q, the quantity ρ^2 has a roughly Gaussian distribution with mean and standard deviation:

$$\rho^2 \sim N\sigma^2 \pm \sqrt{2N}\sigma^2. \qquad (21)$$

Thus almost all samples from Q lie in a 'typical set' with distance from the origin very close to $\sqrt{N}\sigma$. Let us assume that σ is chosen such that the typical set of Q lies inside the sphere of radius R_P. [If it does not, then the law of large numbers implies that almost all the samples generated from Q

will fall outside R_P and will have weight zero.] Then we know that most samples from Q will have a value of Q that lies in the range

$$\frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{N}{2} \pm \frac{\sqrt{2N}}{2}\right). \tag{22}$$

Thus the weights $w_r = P^*/Q$ will typically have values in the range

$$(2\pi\sigma^2)^{N/2} \exp\left(\frac{N}{2} \pm \frac{\sqrt{2N}}{2}\right). \tag{23}$$

So if we draw a hundred samples, what will the typical range of weights be? We can roughly estimate the ratio of the largest weight to the median weight by doubling the standard deviation in equation (23). The largest weight and the median weight will typically be in the ratio:

$$\frac{w_r^{\max}}{w_r^{\text{med}}} = \exp\left(\sqrt{2N}\right). \tag{24}$$

In $N = 1000$ dimensions therefore, the largest weight after one hundred samples is likely to be roughly 10^{19} times greater than the median weight. Thus an importance sampling estimate for a high–dimensional problem will very likely be utterly dominated by a few samples with huge weights.

In conclusion, importance sampling in high dimensions often suffers from two difficulties. First, we clearly need to obtain samples that lie in the typical set of P, and this may take a long time unless Q is a good approximation to P. Second, even if we obtain samples in the typical set, the weights associated with those samples are likely to vary by large factors, because the probabilities of points in a typical set, although similar to each other, still differ by factors of order $\exp(\sqrt{N})$.

3. Rejection sampling

We assume again a one–dimensional density $P(x) = P^*(x)/Z$ that is too complicated a function for us to be able to sample from it directly. We assume that we have a simpler *proposal density* $Q(x)$ which we can evaluate (within a multiplicative factor Z_Q, as before), and which we can generate samples from. We further assume that we know the value of a constant c such that

$$\text{for all } x,\ cQ^*(x) > P^*(x). \tag{25}$$

A schematic picture of the two functions is shown in figure 4a.

We generate two random numbers. The first, x, is generated from the proposal density $Q(x)$. We then evaluate $cQ^*(x)$ and generate a uniformly

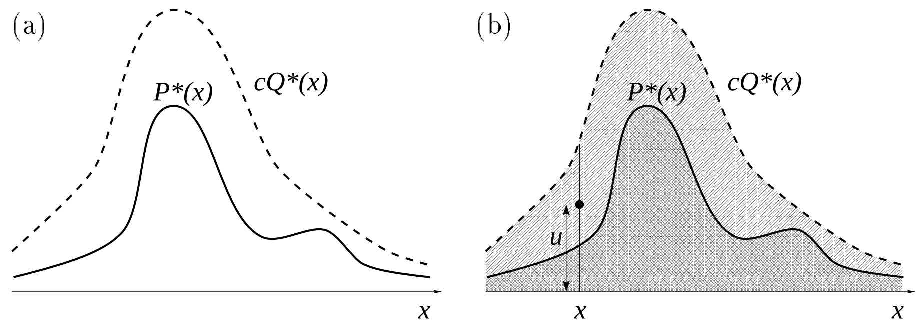

Figure 4. Rejection sampling. a) The functions involved in rejection sampling. We desire samples from $P(x) \propto P^*(x)$. We are able to draw samples from $Q(x) \propto Q^*(x)$, and we know a value c such that $cQ^*(x) > P^*(x)$ for all x. b) A point (x, u) is generated at random in the lightly shaded area under the curve $cQ^*(x)$. If this point also lies below $P^*(x)$ then it is accepted.

distributed random variable u from the interval $[0, cQ^*(x)]$. These two random numbers can be viewed as selecting a point in the two–dimensional plane as shown in figure 4b.

We now evaluate $P^*(x)$ and accept or reject the sample x by comparing the value of u with the value of $P^*(x)$. If $u > P^*(x)$ then x is rejected; otherwise it is accepted, which means that we add x to our set of samples $\{x^{(r)}\}$. The value of u is discarded.

Why does this procedure generate samples from $P(x)$? The proposed point (x, u) comes with uniform probability from the lightly shaded area underneath the curve $cQ^*(x)$ as shown in figure 4b. The rejection rule rejects all the points that lie above the curve $P^*(x)$. So the points (x, u) that are accepted are uniformly distributed in the heavily shaded area under $P^*(x)$. This implies that the probability density of the x–coordinates of the accepted points must be proportional to $P^*(x)$, so the samples must be independent samples from $P(x)$.

Rejection sampling will work best if Q is a good approximation to P. If Q is very different from P then c will necessarily have to be large and the frequency of rejection will be large.

3.1. REJECTION SAMPLING IN MANY DIMENSIONS

In a high–dimensional problem it is very likely that the requirement that cQ^* be an upper bound for P^* will force c to be so huge that acceptances will be very rare indeed. Finding such a value of c may be difficult too, since in many problems we don't know beforehand where the modes of P^* are located or how high they are.

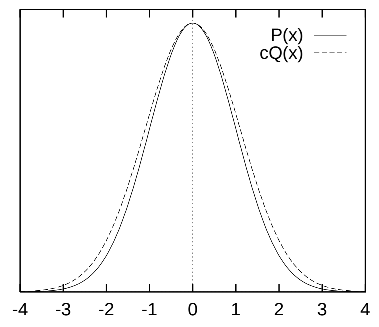

Figure 5. A Gaussian $P(x)$ and a slightly broader Gaussian $Q(x)$ scaled up by a factor c such that $cQ(x) \geq P(x)$.

As a case study, consider a pair of N–dimensional Gaussian distributions with mean zero (figure 5). Imagine generating samples from one with standard deviation σ_Q and using rejection sampling to obtain samples from the other whose standard deviation is σ_P. Let us assume that these two standard deviations are close in value — say, σ_Q is one percent larger than σ_P. [σ_Q must be larger than σ_P because if this is not the case, there is no c such that cQ upper–bounds P for all $\mathbf{x}$.] So, what is the value of c if the dimensionality is $N = 1000$? The density of $Q(\mathbf{x})$ at the origin is $1/(2\pi\sigma_Q^2)^{N/2}$, so for cQ to upper–bound P we need to set

$$c = \frac{(2\pi\sigma_Q^2)^{N/2}}{(2\pi\sigma_P^2)^{N/2}} = \exp\left(N \log \frac{\sigma_Q}{\sigma_P}\right). \tag{26}$$

With $N = 1000$ and $\frac{\sigma_Q}{\sigma_P} = 1.01$, we find $c = \exp(10) \simeq 20,000$. What will the rejection rate be for this value of c? The answer is immediate: since the acceptance rate is the ratio of the volume under the curve $P(\mathbf{x})$ to the volume under $cQ(\mathbf{x})$, the fact that P and Q are normalized implies that the acceptance rate will be $1/c$. For our case study, this is $1/20,000$. In general, c grows exponentially with the dimensionality N.

Rejection sampling, therefore, whilst a useful method for one–dimensional problems, is not a practical technique for generating samples from high–dimensional distributions $P(\mathbf{x})$.

4. The Metropolis method

Importance sampling and rejection sampling only work well if the proposal density $Q(x)$ is similar to $P(x)$. In large and complex problems it is difficult to create a single density $Q(x)$ that has this property.

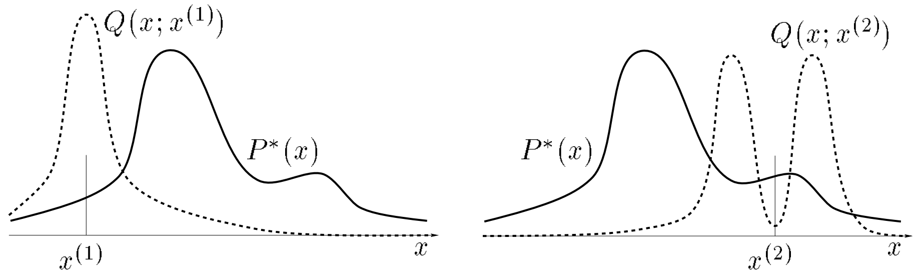

Figure 6. Metropolis method in one dimension. The proposal distribution $Q(x'; x)$ is here shown as having a shape that changes as x changes, though this is not typical of the proposal densities used in practice.

The Metropolis algorithm instead makes use of a proposal density Q *which depends on the current state* $x^{(t)}$. The density $Q(x'; x^{(t)})$ might in the simplest case be a simple distribution such as a Gaussian centred on the current $x^{(t)}$. The proposal density $Q(x'; x)$ can be any fixed density. It is not necessary for $Q(x'; x^{(t)})$ to look at all similar to $P(x)$. An example of a proposal density is shown in figure 6; this figure shows the density $Q(x'; x^{(t)})$ for two different states $x^{(1)}$ and $x^{(2)}$.

As before, we assume that we can evaluate $P^*(x)$ for any x. A tentative new state x' is generated from the proposal density $Q(x'; x^{(t)})$. To decide whether to accept the new state, we compute the quantity

$$a = \frac{P^*(x')}{P^*(x^{(t)})} \frac{Q(x^{(t)}; x')}{Q(x'; x^{(t)})}. \tag{27}$$

If $a \geq 1$ then the new state is accepted.
Otherwise, the new state is accepted with probability a. (28)

If the step is accepted, we set $x^{(t+1)} = x'$. If the step is rejected, then we set $x^{(t+1)} = x^{(t)}$. Note the difference from rejection sampling: in rejection sampling, rejected points are discarded and have no influence on the list of samples $\{x^{(r)}\}$ that we collected. Here, a rejection causes the current state to be written onto the list of points another time.

Notation: I have used the superscript $r = 1 \ldots R$ to label points that are *independent* samples from a distribution, and the superscript $t = 1 \ldots T$ to label the sequence of states in a Markov chain. It is important to note that a Metropolis simulation of T iterations does not produce T independent samples from the target distribution P. The samples are correlated.

To compute the acceptance probability we need to be able to compute the probability ratios $P(x')/P(x^{(t)})$ and $Q(x^{(t)}; x')/Q(x'; x^{(t)})$. If the proposal density is a simple symmetrical density such as a Gaussian centred on the current point, then the latter factor is unity, and the Metropolis

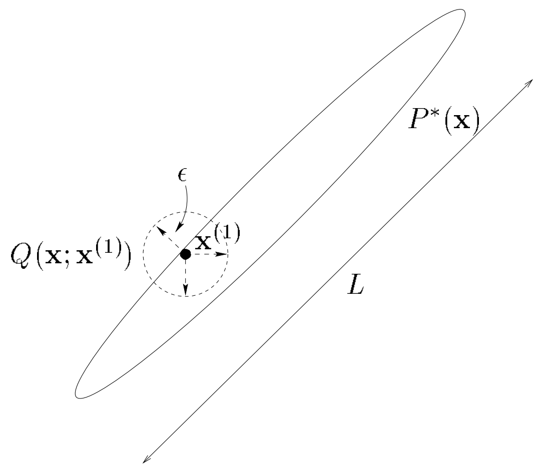

Figure 7. Metropolis method in two dimensions, showing a traditional proposal density that has a sufficiently small step size ϵ that the acceptance frequency will be about 0.5.

method simply involves comparing the value of the target density at the two points. The general algorithm for asymmetric Q, given above, is often called the Metropolis–Hastings algorithm.

It can be shown that for any positive Q (that is, any Q such that $Q(x';x) > 0$ for all x, x'), as $t \to \infty$, the probability distribution of $x^{(t)}$ tends to $P(x) = P^*(x)/Z$. [This statement should not be seen as implying that Q *has* to assign positive probability to every point x' — we will discuss examples later where $Q(x';x) = 0$ for some x, x'; notice also that we have said nothing about how rapidly the convergence to $P(x)$ takes place.]

The Metropolis method is an example of a '**Markov chain Monte Carlo**' method (abbreviated MCMC). In contrast to rejection sampling where the accepted points $\{x^{(r)}\}$ are independent samples from the desired distribution, Markov chain Monte Carlo methods involve a Markov process in which a sequence of states $\{x^{(t)}\}$ is generated, each sample $x^{(t)}$ having a probability distribution that depends on the previous value, $x^{(t-1)}$. Since successive samples are correlated with each other, the Markov chain may have to be run for a considerable time in order to generate samples that are effectively independent samples from P.

Just as it was difficult to estimate the variance of an importance sampling estimator, so it is difficult to assess whether a Markov chain Monte Carlo method has 'converged', and to quantify how long one has to wait to obtain samples that are effectively independent samples from P.

4.1. DEMONSTRATION OF THE METROPOLIS METHOD

The Metropolis method is widely used for high–dimensional problems. Many implementations of the Metropolis method employ a proposal distribution with a length scale ϵ that is short relative to the length scale L of the probable region (figure 7). A reason for choosing a small length scale is that for most high–dimensional problems, a large random step from a typical point (that is, a sample from $P(\mathbf{x})$) is very likely to end in a state which has very low probability; such steps are unlikely to be accepted. If ϵ is large, movement around the state space will only occur when a transition to a state which has very low probability is actually accepted, or when a large random step chances to land in another probable state. So the rate of progress will be slow, unless small steps are used.

The disadvantage of small steps, on the other hand, is that the Metropolis method will explore the probability distribution by a *random walk*, and random walks take a long time to get anywhere. Consider a one–dimensional random walk, for example, on each step of which the state moves randomly to the left or to the right with equal probability. After T steps of size ϵ, the state is only likely to have moved a distance about $\sqrt{T}\epsilon$. Recall that the first aim of Monte Carlo sampling is to generate a number of *independent* samples from the given distribution (a dozen, say). If the largest length scale of the state space is L, then we have to simulate a random–walk Metropolis method for a time $T \simeq (L/\epsilon)^2$ before we can expect to get a sample that is roughly independent of the initial condition — and that's assuming that every step is accepted: if only a fraction f of the steps are accepted on average, then this time is increased by a factor $1/f$.

Rule of thumb: lower bound on number of iterations of a Metropolis method. If the largest length scale of the space of probable states is L, a Metropolis method whose proposal distribution generates a random walk with step size ϵ must be run for at least $T \simeq (L/\epsilon)^2$ iterations to obtain an independent sample.

This rule of thumb only gives a lower bound; the situation may be much worse, if, for example, the probability distribution consists of several islands of high probability separated by regions of low probability.

To illustrate how slow the exploration of a state space by random walk is, figure 8 shows a simulation of a Metropolis algorithm for generating samples from the distribution:

$$P(x) = \begin{cases} \frac{1}{21} & x \in \{0, 1, 2 \ldots, 20\} \\ 0 & \text{otherwise} \end{cases} . \tag{29}$$

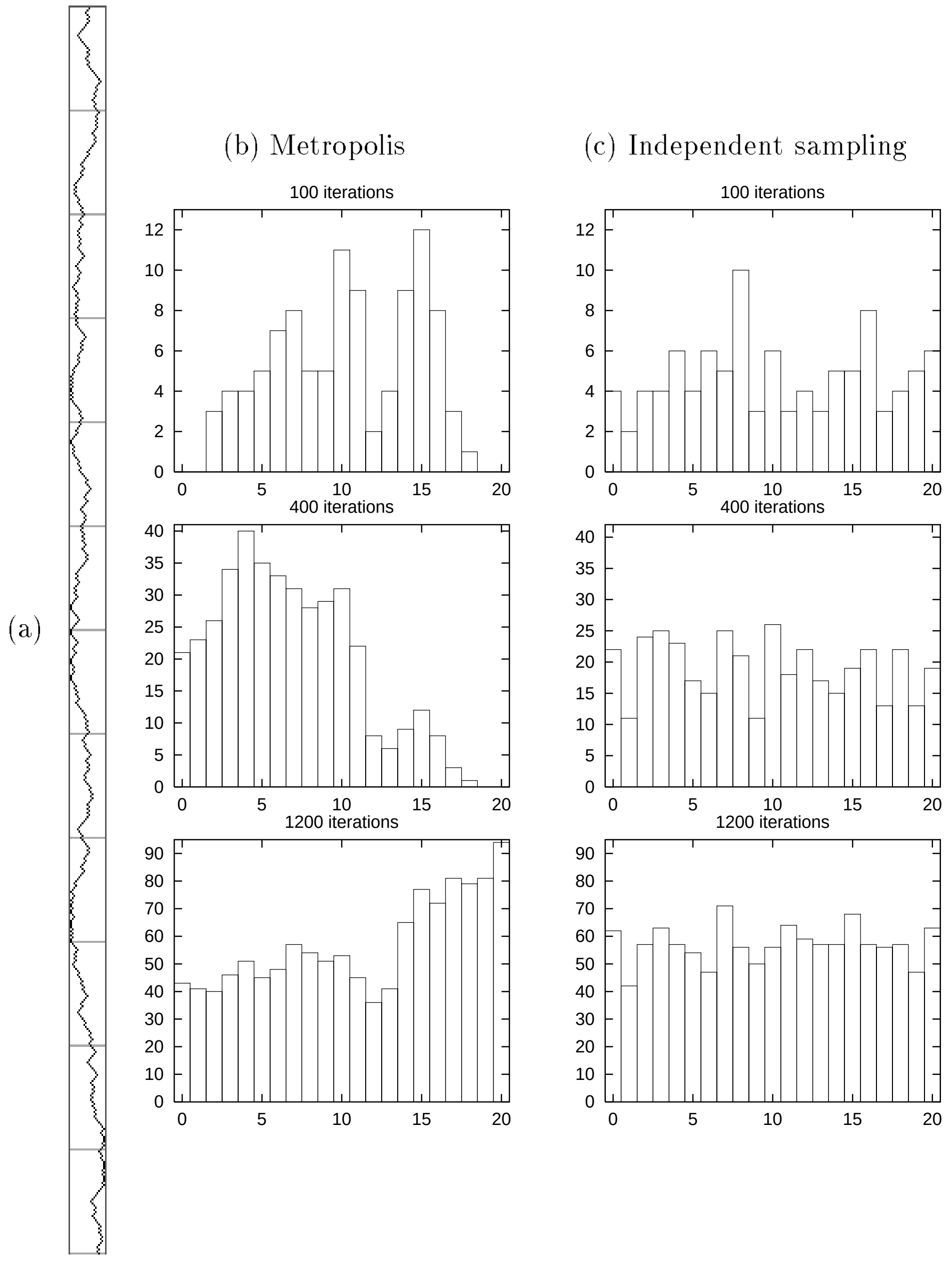

Figure 8. Metropolis method for a toy problem. (a) The state sequence for $t = 1 \ldots 600$. Horizontal direction = states from 0 to 20; vertical direction = time from 1 to 600; the cross bars mark time intervals of duration 50. (b) Histogram of occupancy of the states after 100, 400 and 1200 iterations. (c) For comparison, histograms resulting when successive points are drawn *independently* from the target distribution.

The proposal distribution is

$$Q(x';x) = \begin{cases} \frac{1}{2} & x' = x \pm 1 \\ 0 & \text{otherwise} \end{cases} . \tag{30}$$

Because the target distribution $P(x)$ is uniform, rejections will occur only when the proposal takes the state to $x' = -1$ or $x' = 21$.

The simulation was started in the state $x_0 = 10$ and its evolution is shown in figure 8a. How long does it take to reach one of the end states $x = 0$ and $x = 20$? Since the distance is 10 steps the rule of thumb above predicts that it will typically take a time $T \simeq 100$ iterations to reach an end state. This is confirmed in the present example. The first step into an end state occurs on the 178th iteration. How long does it take to visit *both* end states? The rule of thumb predicts about 400 iterations are required to traverse the whole state space. And indeed the first encounter with the other end state takes place on the 540th iteration. Thus effectively independent samples are only generated by simulating for about four hundred iterations.

This simple example shows that it is important to try to abolish random walk behaviour in Monte Carlo methods. A systematic exploration of the toy state space $\{0, 1, 2, \ldots 20\}$ could get around it, using the same step sizes, in about twenty steps instead of four hundred!

4.2. METROPOLIS METHOD IN HIGH DIMENSIONS

The rule of thumb that we discussed above, giving a lower bound on the number of iterations of a random walk Metropolis method, also applies to higher dimensional problems. Consider the simplest case of a target distribution that is a Gaussian, and a proposal distribution that is a spherical Gaussian of standard deviation in each direction equal to ϵ. Without loss of generality, we can assume that the target distribution is a separable distribution aligned with the axes $\{x_n\}$, and that it has standard deviations $\{\sigma_n\}$ in the different directions n. Let $\sigma^{\max}$ and $\sigma^{\min}$ be the largest and smallest of these standard deviations. Let us assume that ϵ is adjusted such that the acceptance probability is close to 1. Under this assumption, each variable x_n evolves independently of all the others, executing a random walk with step sizes about ϵ. The time taken to generate effectively independent samples from the target distribution will be controlled by the largest lengthscale $\sigma^{\max}$; just as in the previous section, where we needed at least $T \simeq (L/\epsilon)^2$ iterations to obtain an independent sample, here we need $T \simeq (\sigma^{\max}/\epsilon)^2$.

Now how big can ϵ be? The bigger it is, the smaller this number T becomes, but if ϵ is too big — bigger than $\sigma^{\min}$ — then the acceptance rate will fall sharply. It seems plausible that the optimal ϵ must be similar to

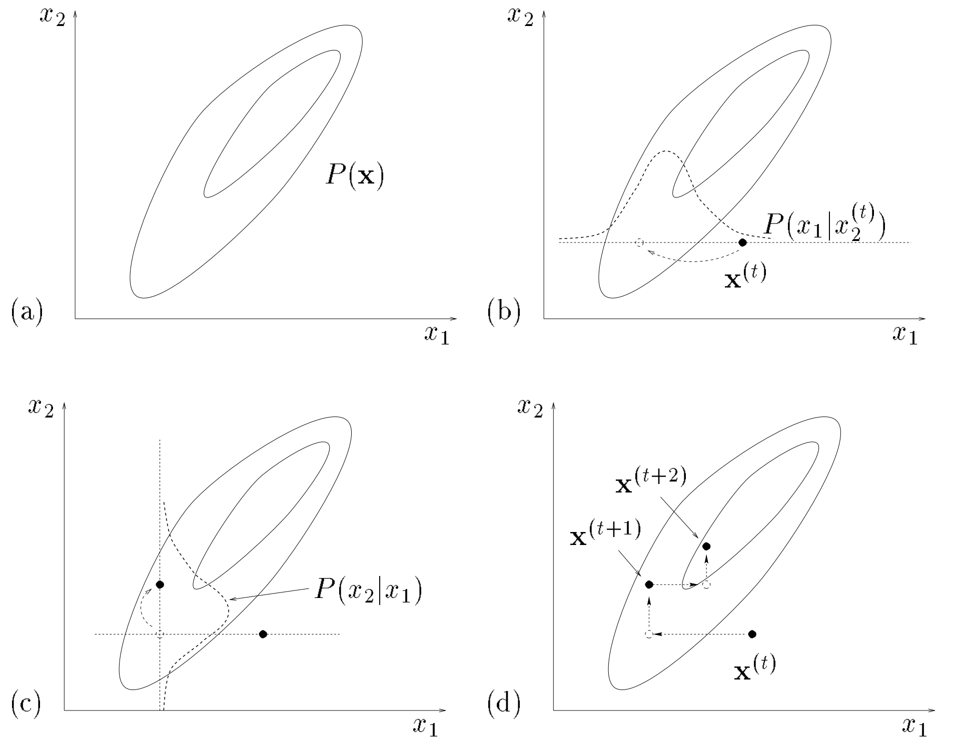

Figure 9. Gibbs sampling. (a) The joint density $P(\mathbf{x})$ from which samples are required. (b) Starting from a state $\mathbf{x}^{(t)}$, x_1 is sampled from the conditional density $P(x_1|x_2^{(t)})$. (c) A sample is then made from the conditional density $P(x_2|x_1)$. (d) A couple of iterations of Gibbs sampling.

$\sigma^{\min}$. Strictly, this may not be true; in special cases where the second smallest σ_n is significantly greater than $\sigma^{\min}$, the optimal ϵ may be closer to that second smallest σ_n. But our rough conclusion is this: where simple spherical proposal distributions are used, we will need at least $T \simeq (\sigma^{\max}/\sigma^{\min})^2$ iterations to obtain an independent sample, where $\sigma^{\max}$ and $\sigma^{\min}$ are the longest and shortest lengthscales of the target distribution.

This is good news and bad news. It is good news because, unlike the cases of rejection sampling and importance sampling, there is no catastrophic dependence on the dimensionality N. But it is bad news in that all the same, this quadratic dependence on the lengthscale ratio may force us to make very lengthy simulations.

Fortunately, there are methods for suppressing random walks in Monte Carlo simulations, which we will discuss later.

5. Gibbs sampling

We introduced importance sampling, rejection sampling and the Metropolis method using one–dimensional examples. Gibbs sampling, also known as the *heat bath method*, is a method for sampling from distributions over at least two dimensions. It can be viewed as a Metropolis method in which the proposal distribution Q is defined in terms of the *conditional* distributions of the joint distribution $P(\mathbf{x})$. It is assumed that whilst $P(\mathbf{x})$ is too complex to draw samples from directly, its conditional distributions $P(x_i|\{x_j\}_{j\neq i})$ are tractable to work with. For many graphical models (but not all) these one–dimensional conditional distributions are straightforward to sample from. Conditional distributions that are not of standard form may still be sampled from by adaptive rejection sampling if the conditional distribution satisfies certain convexity properties (Gilks and Wild 1992).

Gibbs sampling is illustrated for a case with two variables $(x_1, x_2) = \mathbf{x}$ in figure 9. On each iteration, we start from the current state $\mathbf{x}^{(t)}$, and x_1 is sampled from the conditional density $P(x_1|x_2)$, with x_2 fixed to $x_2^{(t)}$. A sample x_2 is then made from the conditional density $P(x_2|x_1)$, using the new value of x_1. This brings us to the new state $\mathbf{x}^{(t+1)}$, and completes the iteration.

In the general case of a system with K variables, a single iteration involves sampling one parameter at a time:

$$\begin{aligned} x_1^{(t+1)} &\sim P(x_1|x_2^{(t)}, x_3^{(t)}, \ldots x_K^{(t)}) \quad (31)\\ x_2^{(t+1)} &\sim P(x_2|x_1^{(t+1)}, x_3^{(t)}, \ldots x_K^{(t)}) \quad (32)\\ x_3^{(t+1)} &\sim P(x_3|x_1^{(t+1)}, x_2^{(t+1)}, \ldots x_K^{(t)}), \text{etc.} \quad (33) \end{aligned}$$

Gibbs sampling can be viewed as a Metropolis method which has the property that every proposal is always accepted. Because Gibbs sampling is a Metropolis method, the probability distribution of $\mathbf{x}^{(t)}$ tends to $P(\mathbf{x})$ as $t \to \infty$, as long as $P(\mathbf{x})$ does not have pathological properties.

5.1. GIBBS SAMPLING IN HIGH DIMENSIONS

Gibbs sampling suffers from the same defect as simple Metropolis algorithms — the state space is explored by a random walk, unless a fortuitous parameterization has been chosen which makes the probability distribution $P(\mathbf{x})$ separable. If, say, two variables x_1 and x_2 are strongly correlated, having marginal densities of width L and conditional densities of width ϵ, then it will take at least about $(L/\epsilon)^2$ iterations to generate an independent sample from the target density. However Gibbs sampling involves no adjustable parameters, so it is an attractive strategy when one wants to get a model running quickly. An excellent software package, `BUGS`, is available which makes it easy to set up almost arbitrary probabilistic models and simulate them by Gibbs sampling (Thomas, Spiegelhalter and Gilks 1992).

6. Terminology for Markov chain Monte Carlo methods

We now spend a few moments sketching the theory on which the Metropolis method and Gibbs sampling are based.

A **Markov chain** can be specified by an **initial** probability distribution $p^{(0)}(\mathbf{x})$ and a **transition probability** $T(\mathbf{x}';\mathbf{x})$.

The probability distribution of the state at the $(t+1)$th iteration of the Markov chain is given by

$$p^{(t+1)}(\mathbf{x}') = \int d^N\mathbf{x}\, T(\mathbf{x}';\mathbf{x})p^{(t)}(\mathbf{x}). \tag{34}$$

We construct the chain such that:

1. The desired distribution $P(\mathbf{x})$ is the **invariant distribution** of the chain.

   A distribution $\pi(\mathbf{x})$ is an invariant distribution of $T(\mathbf{x}';\mathbf{x})$ if

   $$\pi(\mathbf{x}') = \int d^N\mathbf{x}\, T(\mathbf{x}';\mathbf{x})\pi(\mathbf{x}). \tag{35}$$

2. The chain must also be **ergodic**, that is,

   $$p^{(t)}(\mathbf{x}) \rightarrow \pi(\mathbf{x}) \text{ as } t \rightarrow \infty, \text{ for any } p^{(0)}(\mathbf{x}). \tag{36}$$

It is often convenient to construct T by mixing or concatenating simple **base transitions** B all of which satisfy

$$P(\mathbf{x}') = \int d^N\mathbf{x}\, B(\mathbf{x}';\mathbf{x})P(\mathbf{x}), \tag{37}$$

for the desired density $P(\mathbf{x})$. These base transitions need not be individually ergodic.

Many useful transition probabilities satisfy the **detailed balance** property:

$$T(\mathbf{x}';\mathbf{x})P(\mathbf{x}) = T(\mathbf{x};\mathbf{x}')P(\mathbf{x}'), \text{ for all } \mathbf{x} \text{ and } \mathbf{x}'. \tag{38}$$

This equation says that if we pick a state from the target density P and make a transition under T to another state, it is just as likely that we will pick $\mathbf{x}$ and go from $\mathbf{x}$ to $\mathbf{x}'$ as it is that we will pick $\mathbf{x}'$ and go from $\mathbf{x}'$ to $\mathbf{x}$. Markov chains that satisfy detailed balance are also called **reversible** Markov chains. The reason why the detailed balance property is of interest is that detailed balance implies invariance of the distribution $P(\mathbf{x})$ under the Markov chain T (the proof of this is left as an exercise for the reader). Proving that detailed balance holds is often a key step when proving that a Markov chain Monte Carlo simulation will converge to the desired distribution. The Metropolis method and Gibbs sampling method both satisfy detailed balance, for example. Detailed balance is not an essential condition, however, and we will see later that irreversible Markov chains can be useful in practice.

7. Practicalities

Can we predict how long a Markov chain Monte Carlo simulation will take to equilibrate? By considering the random walks involved in a Markov chain Monte Carlo simulation we can obtain simple *lower bounds* on the time required for convergence. But predicting this time more precisely is a difficult problem, and most of the theoretical results are of little practical use.
Can we diagnose or detect convergence in a running simulation? This is also a difficult problem. There are a few practical tools available, but none of them is perfect (Cowles and Carlin 1996).
Can we speed up the convergence time and time between independent samples of a Markov chain Monte Carlo method? Here, there is good news.

7.1. SPEEDING UP MONTE CARLO METHODS

7.1.1. *Reducing random walk behaviour in Metropolis methods*

The hybrid Monte Carlo method reviewed in Neal (1993) is a Metropolis method applicable to continuous state spaces which makes use of gradient information to reduce random walk behaviour.

For many systems, the probability $P(\mathbf{x})$ can be written in the form

$$P(\mathbf{x}) = \frac{e^{-E(\mathbf{x})}}{Z} \tag{39}$$

where not only $E(\mathbf{x})$, but also its gradient with respect to $\mathbf{x}$ can be readily evaluated. It seems wasteful to use a simple random–walk Metropolis method when this gradient is available — the gradient indicates which direction one should go in to find states with higher probability!

In the hybrid Monte Carlo method, the state space $\mathbf{x}$ is augmented by *momentum variables* $\mathbf{p}$, and there is an alternation of two types of proposal. The first proposal randomizes the momentum variable, leaving the state $\mathbf{x}$ unchanged. The second proposal changes both $\mathbf{x}$ and $\mathbf{p}$ using simulated Hamiltonian dynamics as defined by the Hamiltonian

$$H(\mathbf{x}, \mathbf{p}) = E(\mathbf{x}) + K(\mathbf{p}), \tag{40}$$

```
g = gradE ( x ) ;            # set gradient using initial x
E = findE ( x ) ;            # set objective function too

for l = 1:L                  # loop L times
   p = randn ( size(x) ) ;   # initial momentum is Normal(0,1)
   H = p' * p / 2 + E ;      # evaluate H(x,p)

   xnew = x
   gnew = g ;
   for tau = 1:Tau           # make Tau `leapfrog' steps

      p = p - epsilon * gnew / 2 ; # make half-step in p
      xnew = xnew + epsilon * p ;  # make step in x

      gnew = gradE ( xnew ) ;      # find new gradient
      p = p - epsilon * gnew / 2 ; # make half-step in p

   endfor

   Enew = findE ( xnew ) ;   # find new value of H
   Hnew = p' * p / 2 + Enew ;
   dH = Hnew - H ;           # Decide whether to accept

   if ( dH < 0 )                accept = 1 ;
   elseif ( rand() < exp(-dH) ) accept = 1 ;
   else                         accept = 0 ;
   endif

   if ( accept )
      g = gnew ;   x = xnew ;    E = Enew ;
   endif
endfor
```

Figure 10. Octave source code for the hybrid Monte Carlo method.

where $K(\mathbf{p})$ is a 'kinetic energy' such as $K(\mathbf{p}) = \mathbf{p}^{\mathrm{T}}\mathbf{p}/2$. These two proposals are used to create (asymptotically) samples from the joint density

$$P_H(\mathbf{x}, \mathbf{p}) = \frac{1}{Z_H} \exp[-H(\mathbf{x}, \mathbf{p})] = \frac{1}{Z_H} \exp[-E(\mathbf{x})] \exp[-K(\mathbf{p})]. \qquad (41)$$

This density is separable, so it is clear that the marginal distribution of $\mathbf{x}$ is the desired distribution $\exp[-E(\mathbf{x})]/Z$. So, simply discarding the momen-

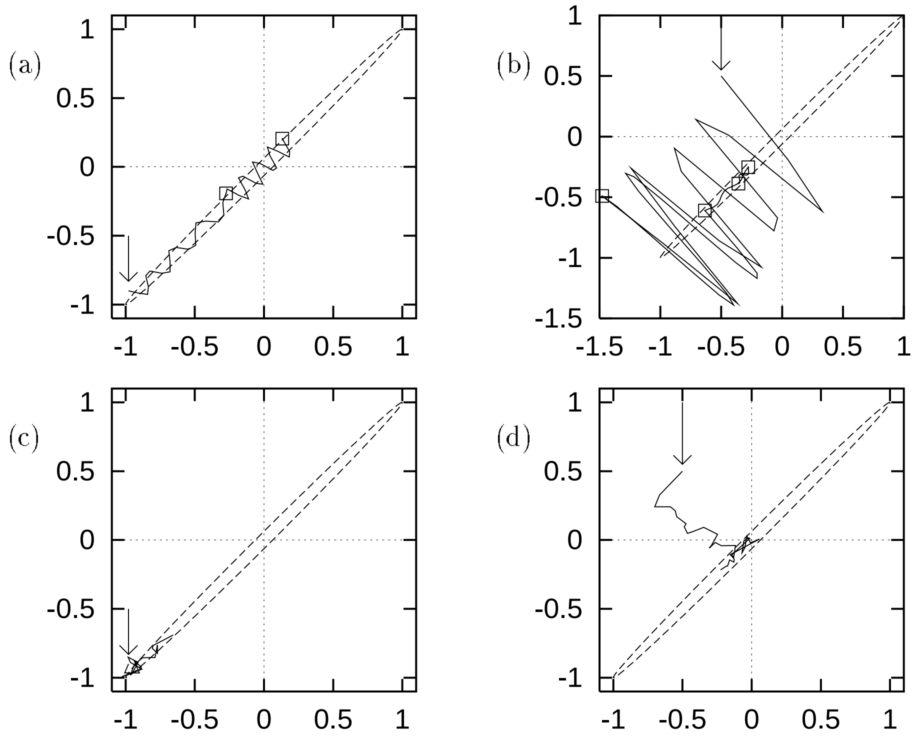

Figure 11. (a,b) Hybrid Monte Carlo used to generate samples from a bivariate Gaussian with correlation $\rho = 0.998$. (c,d) Random–walk Metropolis method for comparison. (a) Starting from the state indicated by the arrow, the continuous line represents two successive trajectories generated by the Hamiltonian dynamics. The squares show the endpoints of these two trajectories. Each trajectory consists of `Tau` = 19 'leapfrog' steps with `epsilon` = 0.055. After each trajectory, the momentum is randomized. Here, both trajectories are accepted; the errors in the Hamiltonian were +0.016 and −0.06 respectively. (b) The second figure shows how a sequence of four trajectories converges from an initial condition, indicated by the arrow, that is not close to the typical set of the target distribution. The trajectory parameters `Tau` and `epsilon` were randomized for each trajectory using uniform distributions with means 19 and 0.055 respectively. The first trajectory takes us to a new state, $(-1.5, -0.5)$, similar in energy to the first state. The second trajectory happens to end in a state nearer the bottom of the energy landscape. Here, since the potential energy E is smaller, the kinetic energy $K = \mathbf{p}^2/2$ is necessarily larger than it was at the start. When the momentum is randomized for the third trajectory, its magnitude becomes much smaller. After the fourth trajectory has been simulated, the state appears to have become typical of the target density. (c) A random–walk Metropolis method using a Gaussian proposal density with radius such that the acceptance rate was 58% in this simulation. The number of proposals was 38 so the total amount of computer time used was similar to that in (a). The distance moved is small because of random walk behaviour. (d) A random–walk Metropolis method given a similar amount of computer time to (b).

tum variables, we will obtain a sequence of samples $\{\mathbf{x}^{(t)}\}$ which asymptotically come from $P(\mathbf{x})$.

The first proposal draws a new momentum from the Gaussian density $\exp[-K(\mathbf{p})]/Z_K$. During the second, dynamical proposal, the momentum variable determines where the state $\mathbf{x}$ goes, and the *gradient* of $E(\mathbf{x})$ determines how the momentum $\mathbf{p}$ changes, in accordance with the equations

$$\dot{\mathbf{x}} = \mathbf{p} \tag{42}$$

$$\dot{\mathbf{p}} = -\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}}. \tag{43}$$

Because of the persistent motion of $\mathbf{x}$ in the direction of the momentum $\mathbf{p}$, during each dynamical proposal, the state of the system tends to move a distance that goes *linearly* with the computer time, rather than as the square root.

If the simulation of the Hamiltonian dynamics is numerically perfect then the proposals are accepted every time, because the total energy $H(\mathbf{x},\mathbf{p})$ is a constant of the motion and so a in equation (27) is equal to one. If the simulation is imperfect, because of finite step sizes for example, then some of the dynamical proposals will be rejected. The rejection rule makes use of the change in $H(\mathbf{x},\mathbf{p})$, which is zero if the simulation is perfect. The occasional rejections ensure that asymptotically, we obtain samples $(\mathbf{x}^{(t)},\mathbf{p}^{(t)})$ from the required joint density $P_H(\mathbf{x},\mathbf{p})$.

The source code in figure 10 describes a hybrid Monte Carlo method which uses the 'leapfrog' algorithm to simulate the dynamics on the function `findE(x)`, whose gradient is found by the function `gradE(x)`. Figure 11 shows this algorithm generating samples from a bivariate Gaussian whose energy function is $E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}$ with

$$\mathbf{A} = \begin{bmatrix} 250.25 & -249.75 \\ -249.75 & 250.25 \end{bmatrix}. \tag{44}$$

7.1.2. *Overrelaxation*

The method of '**overrelaxation**' is a similar method for reducing random walk behaviour in Gibbs sampling. Overrelaxation was originally introduced for systems in which all the conditional distributions are Gaussian. (There are joint distributions that are *not* Gaussian whose conditional distributions *are* all Gaussian, for example, $P(x,y) = \exp(-x^2y^2)/Z$.)

In ordinary Gibbs sampling, one draws the new value $x_i^{(t+1)}$ of the current variable x_i from its conditional distribution, ignoring the old value $x_i^{(t)}$. This leads to lengthy random walks in cases where the variables are strongly correlated, as illustrated in the left hand panel of figure 12.

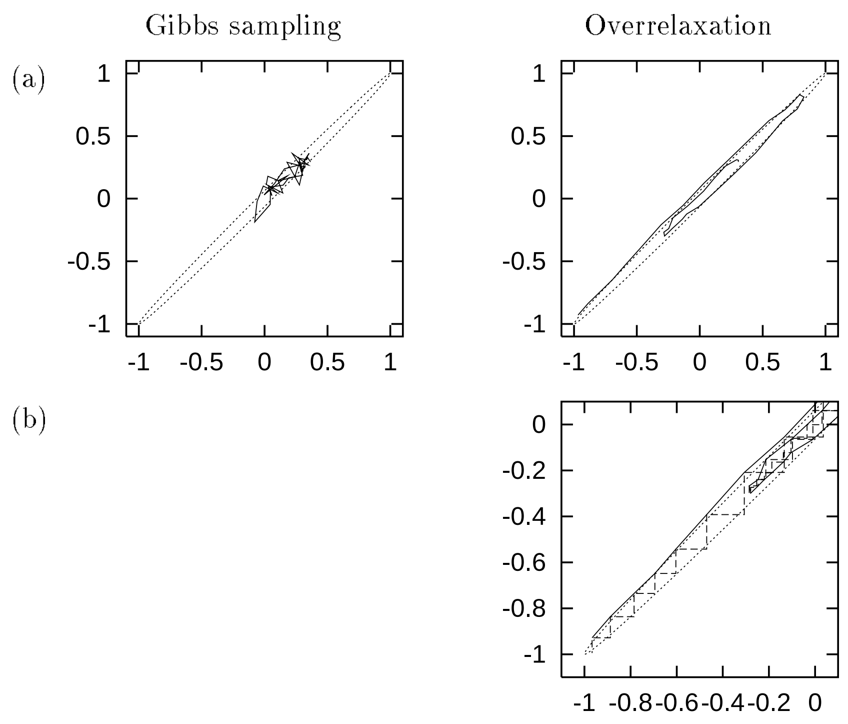

Figure 12. Overrelaxation contrasted with Gibbs sampling for a bivariate Gaussian with correlation $\rho = 0.998$. (a) The state sequence for 40 iterations, each iteration involving one update of both variables. The overrelaxation method had $\alpha = -0.98$. (This excessively large value is chosen to make it easy to see how the overrelaxation method reduces random walk behaviour.) The dotted line shows the contour $\mathbf{x}^{\mathrm{T}}\Sigma^{-1}\mathbf{x} = 1$. (b) Detail of (a), showing the two steps making up each iteration. (After Neal (1995).)

In Adler's (1981) overrelaxation method, one instead samples $x_i^{(t+1)}$ from a Gaussian that is biased to the opposite side of the conditional distribution. If the conditional distribution of x_i is Normal(μ, σ^2) and the current value of x_i is $x_i^{(t)}$, then Adler's method sets x_i to

$$x_i^{(t+1)} = \mu + \alpha(x_i^{(t)} - \mu) + (1 - \alpha^2)^{1/2}\sigma\nu, \tag{45}$$

where $\nu \sim$ Normal$(0, 1)$ and α is a parameter between -1 and 1, commonly set to a negative value.

The transition matrix $T(\mathbf{x}'; \mathbf{x})$ defined by this procedure does not satisfy detailed balance. The individual transitions for the individual coordinates just described *do* satisfy detailed balance, but when we form a chain by applying them in a fixed sequence, the overall chain is not reversible. If, say, two variables are positively correlated, then they will (on a short timescale) evolve in a directed manner instead of by random walk, as shown in figure 12. This may significantly reduce the time required to obtain effectively independent samples. This method is still a valid sampling strategy — it converges to the target density $P(\mathbf{x})$ — because it is made up of transitions that satisfy detailed balance.

The overrelaxation method has been generalized by Neal (1995, and this volume) whose 'ordered overrelaxation' method is applicable to any system where Gibbs sampling is used. For practical purposes this method may speed up a simulation by a factor of ten or twenty.

7.1.3. *Simulated annealing*

A third technique for speeding convergence is **simulated annealing**. In simulated annealing, a 'temperature' parameter is introduced which, when large, allows the system to make transitions which would be improbable at temperature 1. The temperature may be initially set to a large value and reduced gradually to 1. It is hoped that this procedure reduces the chance of the simulation's becoming stuck in an unrepresentative probability island.

We asssume that we wish to sample from a distribution of the form

$$P(\mathbf{x}) = \frac{e^{-E(\mathbf{x})}}{Z} \tag{46}$$

where $E(\mathbf{x})$ can be evaluated. In the simplest simulated annealing method, we instead sample from the distribution

$$P_T(\mathbf{x}) = \frac{1}{Z(T)} e^{-\frac{E(\mathbf{x})}{T}} \tag{47}$$

and decrease T gradually to 1.

Often the energy function can be separated into two terms,

$$E(\mathbf{x}) = E_0(\mathbf{x}) + E_1(\mathbf{x}), \tag{48}$$

of which the first term is 'nice' (for example, a separable function of $\mathbf{x}$) and the second is 'nasty'. In these cases, a better simulated annealing method might make use of the distribution

$$P'_T(\mathbf{x}) = \frac{1}{Z'(T)} e^{-E_0(\mathbf{x}) - \frac{E_1(\mathbf{x})}{T}} \tag{49}$$

with T gradually decreasing to 1. In this way, the distribution at high temperatures reverts to a well–behaved distribution defined by E_0.

Simulated annealing is often used as an optimization method, where the aim is to find an $\mathbf{x}$ that minimizes $E(\mathbf{x})$, in which case the temperature is decreased to zero rather than to 1. As a Monte Carlo method, simulated annealing as described above doesn't sample exactly from the right distribution; the closely related 'simulated tempering' methods (Marinari and Parisi 1992) correct the biases introduced by the annealing process by making the temperature itself a random variable that is updated in Metropolis fashion during the simulation.

7.2. CAN THE NORMALIZING CONSTANT BE EVALUATED?

If the target density $P(\mathbf{x})$ is given in the form of an unnormalized density $P^*(\mathbf{x})$ with $P(\mathbf{x}) = \frac{1}{Z}P^*(\mathbf{x})$, the value of Z may well be of interest. Monte Carlo methods do not readily yield an estimate of this quantity, and it is an area of active research to find ways of evaluating it. Techniques for evaluating Z include:

1. Importance sampling (reviewed by Neal (1993)).
2. 'Thermodynamic integration' during simulated annealing, the 'acceptance ratio' method, and 'umbrella sampling' (reviewed by Neal (1993)).
3. 'Reversible jump Markov chain Monte Carlo' (Green 1995).

Perhaps the best way of dealing with Z, however, is to find a solution to one's task that does not require that Z be evaluated. In Bayesian data modelling one can avoid the need to evaluate Z — which would be important for model comparison — by not having more than one model. Instead of using several models (differing in complexity, for example) and evaluating their relative posterior probabilities, one can make a single **hierarchical** model having, for example, various continuous hyperparameters which play a role similar to that played by the distinct models (Neal 1996).

7.3. THE METROPOLIS METHOD FOR BIG MODELS

Our original description of the Metropolis method involved a joint updating of all the variables using a proposal density $Q(\mathbf{x}'; \mathbf{x})$. For big problems it may be more efficient to use several proposal distributions $Q^{(b)}(\mathbf{x}'; \mathbf{x})$, each of which updates only some of the components of $\mathbf{x}$. Each proposal is individually accepted or rejected, and the proposal distributions are repeatedly run through in sequence.

In the Metropolis method, the proposal density $Q(\mathbf{x}'; \mathbf{x})$ typically has a number of parameters that control, for example, its 'width'. These parameters are usually set by trial and error with the rule of thumb being that one aims for a rejection frequency of about 0.5. It is *not* valid to have the width parameters be dynamically updated during the simulation in a way that depends on the history of the simulation. Such a modification of the proposal density would violate the detailed balance condition which guarantees that the Markov chain has the correct invariant distribution.

7.4. GIBBS SAMPLING IN BIG MODELS

Our description of Gibbs sampling involved sampling one parameter at a time, as described in equations (31–33). For big problems it may be more efficient to sample *groups* of variables jointly, that is to use several proposal distributions:

$$x_1^{(t+1)} \ldots x_a^{(t+1)} \sim P(x_1 \ldots x_a | x_{a+1}^{(t)} \ldots x_K^{(t)}) \tag{50}$$

$$x_{a+1}^{(t+1)} \ldots x_b^{(t+1)} \sim P(x_{a+1} \ldots x_b | x_1^{(t+1)} \ldots x_a^{(t+1)}, x_{b+1}^{(t)} \ldots x_K^{(t)}), \text{ etc..} \tag{51}$$

7.5. HOW MANY SAMPLES ARE NEEDED?

At the start of this chapter, we observed that the variance of an estimator $\hat{\Phi}$ depends only on the number of independent samples R and the value of

$$\sigma^2 = \int d^N\mathbf{x}\, P(\mathbf{x})(\phi(\mathbf{x}) - \Phi)^2. \tag{52}$$

We have now discussed a variety of methods for generating samples from $P(\mathbf{x})$. How many independent samples R should we aim for?

In many problems, we really only need about twelve independent samples from $P(\mathbf{x})$. Imagine that $\mathbf{x}$ is an unknown vector such as the amount of corrosion present in each of 10,000 underground pipelines around Sicily, and $\phi(\mathbf{x})$ is the total cost of repairing those pipelines. The distribution $P(\mathbf{x})$ describes the probability of a state $\mathbf{x}$ given the tests that have been carried out on some pipelines and the assumptions about the physics of corrosion. The quantity Φ is the expected cost of the repairs. The quantity σ^2 is the variance of the cost — σ measures by how much we should expect the actual cost to differ from the expectation Φ.

Now, how accurately would a manager like to know Φ? I would suggest there is little point in knowing Φ to a precision finer than about $\sigma/3$. After all, the true cost is likely to differ by $\pm\sigma$ from Φ. If we obtain $R = 12$ independent samples from $P(\mathbf{x})$, we can estimate Φ to a precision of $\sigma/\sqrt{12}$ — which is smaller than $\sigma/3$. So twelve samples suffice.

7.6. ALLOCATION OF RESOURCES

Assuming we have decided how many independent samples R are required, an important question is how one should make use of one's limited computer resources to obtain these samples.

A typical Markov chain Monte Carlo experiment involves an initial period in which control parameters of the simulation such as step sizes may be adjusted. This is followed by a 'burn in' period during which we hope the

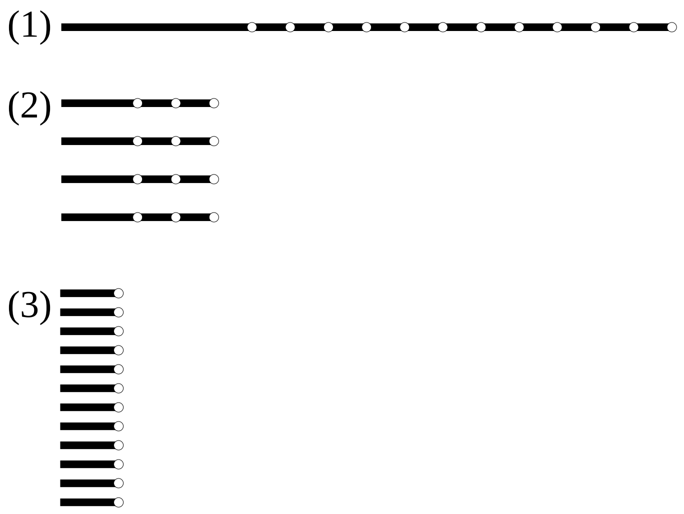

Figure 13. Three possible Markov Chain Monte Carlo strategies for obtaining twelve samples using a fixed amount of computer time. Computer time is represented by horizontal lines; samples by white circles. (1) A single run consisting of one long 'burn in' period followed by a sampling period. (2) Four medium–length runs with different initial conditions and a medium–length burn in period. (3) Twelve short runs.

simulation 'converges' to the desired distribution. Finally, as the simulation continues, we record the state vector occasionally so as to create a list of states $\{\mathbf{x}^{(r)}\}_{r=1}^{R}$ that we hope are roughly independent samples from $P(\mathbf{x})$.

There are several possible strategies (figure 13).

1. Make one long run, obtaining all R samples from it.
2. Make a few medium length runs with different initial conditions, obtaining some samples from each.
3. Make R short runs, each starting from a different random initial condition, with the only state that is recorded being the final state of each simulation.

The first strategy has the best chance of attaining 'convergence'. The last strategy may have the advantage that the correlations between the recorded samples are smaller. The middle path appears to be popular with Markov chain Monte Carlo experts because it avoids the inefficiency of discarding burn–in iterations in many runs, while still allowing one to detect problems with lack of convergence that would not be apparent from a single run.

7.7. PHILOSOPHY

One curious defect of these Monte Carlo methods — which are widely used by Bayesian statisticians — is that they are all non–Bayesian. They involve computer experiments from which *estimators* of quantities of interest are derived. These estimators depend on the sampling distributions that were used to generate the samples. In contrast, an alternative Bayesian approach to the problem would use the results of our computer experiments to infer the properties of the target function $P(\mathbf{x})$ and generate predictive distributions for quantities of interest such as Φ. This approach would give answers which would depend only on the computed values of $P^*(\mathbf{x}^{(r)})$ at the points $\{\mathbf{x}^{(r)}\}$; the answers would not depend on how those points were chosen.

It remains an open problem to create a Bayesian version of Monte Carlo methods.

8. Summary

- Monte Carlo methods are a powerful tool that allow one to implement any probability distribution that can be expressed in the form $P(\mathbf{x}) = \frac{1}{Z}P^*(\mathbf{x})$.
- Monte Carlo methods can answer virtually any query related to $P(\mathbf{x})$ by putting the query in the form

$$\int \phi(\mathbf{x})P(\mathbf{x}) \simeq \frac{1}{R}\sum_r \phi(\mathbf{x}^{(r)}). \tag{53}$$

- In high–dimensional problems the only satisfactory methods are those based on Markov chain Monte Carlo: the Metropolis method and Gibbs sampling.
- Simple Metropolis algorithms, although widely used, perform poorly because they explore the space by a slow random walk. More sophisticated Metropolis algorithms such as hybrid Monte Carlo make use of proposal densities that give faster movement through the state space. The efficiency of Gibbs sampling is also troubled by random walks. The method of ordered overrelaxation is a general purpose technique for suppressing them.

ACKNOWLEDGEMENTS

This presentation of Monte Carlo methods owes a great deal to Wally Gilks and David Spiegelhalter. I thank Radford Neal for teaching me about Monte Carlo methods and for giving helpful comments on the manuscript.

References

Adler, S. L.: 1981, Over-relaxation method for the Monte-Carlo evaluation of the partition function for multiquadratic actions, *Physical Review D-Particles and Fields* **23**(12), 2901–2904.

Cowles, M. K. and Carlin, B. P.: 1996, Markov-chain Monte-Carlo convergence diagnostics — a comparative review, *Journal of the American Statistical Association* **91**(434), 883–904.

Gilks, W. and Wild, P.: 1992, Adaptive rejection sampling for Gibbs sampling, *Applied Statistics* **41**, 337–348.

Gilks, W. R., Richardson, S. and Spiegelhalter, D. J.: 1996, *Markov Chain Monte Carlo in Practice*, Chapman and Hall.

Green, P. J.: 1995, Reversible jump Markov chain Monte Carlo computation and Bayesian model determination, *Biometrika* **82**, 711–732.

Marinari, E. and Parisi, G.: 1992, Simulated tempering - a new Monte-Carlo scheme, *Europhysics Letters* **19**(6), 451–458.

Neal, R. M.: 1993, Probabilistic inference using Markov chain Monte Carlo methods, *Technical Report CRG–TR–93–1*, Dept. of Computer Science, University of Toronto.

Neal, R. M.: 1995, Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation, *Technical Report 9508*, Dept. of Statistics, University of Toronto.

Neal, R. M.: 1996, *Bayesian Learning for Neural Networks*, number 118 in *Lecture Notes in Statistics*, Springer, New York.

Propp, J. G. and Wilson, D. B.: 1996, Exact sampling with coupled Markov chains and applications to statistical mechanics, *Random Structures and Algorithms* **9**(1-2), 223–252.

Tanner, M. A.: 1996, *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions*, Springer Series in Statistics, 3rd edn, Springer Verlag.

Thomas, A., Spiegelhalter, D. J. and Gilks, W. R.: 1992, BUGS: A program to perform Bayesian inference using Gibbs sampling, *in* J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith (eds), *Bayesian Statistics 4*, Clarendon Press, Oxford, pp. 837–842.

Yeomans, J.: 1992, *Statistical mechanics of phase transitions*, Clarendon Press, Oxford.

For a full bibliography and a more thorough review of Monte Carlo methods, the reader is encouraged to consult Neal (1993), Gilks et al. (1996), and Tanner (1996).