About Chapter 29

The last couple of chapters have assumed that a Gaussian approximation to the probability distribution we are interested in is adequate. What if it is not? We have already seen an example – clustering – where the likelihood function is multimodal, and has nasty unboundedly high spikes in certain locations in the parameter space; so maximizing the posterior probability and fitting a Gaussian is not always going to work. This difficulty with Laplace's method is one motivation for being interested in Monte Carlo methods. In fact, Monte Carlo methods provide a general-purpose set of tools with applications in Bayesian data modelling and many other fields.

This chapter describes a sequence of methods: *importance sampling*, *rejection sampling*, the *Metropolis method*, *Gibbs sampling* and *slice 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. [A graphical model is a probabilistic model in which dependencies and independencies of variables are represented by edges in a graph whose nodes are the variables.] Along the way, the terminology of Markov chain Monte Carlo methods is presented. The subsequent chapter gives a discussion of advanced 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 (1993b), Gilks *et al.* (1996), and Tanner (1996).

In this chapter 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 $\{\mathbf{x}\}$.

When we discuss transition probability matrices, I will use a right-multiplication convention: I like my matrices to act to the right, preferring

$$\mathbf{u} = \mathbf{M}\mathbf{v} \tag{29.1}$$

to

$$\mathbf{u}^{\mathsf{T}} = \mathbf{v}^{\mathsf{T}}\mathbf{M}^{\mathsf{T}}. \tag{29.2}$$

A transition probability matrix T_{ij} or $T_{i|j}$ specifies the probability, given the current state is j, of making the transition from j to i. The columns of $\mathbf{T}$ are probability vectors. If we write down a transition probability density, we use the same convention for the order of its arguments: $T(x';x)$ is a transition probability density from x to x'. This unfortunately means that you have to get used to reading from right to left – the sequence xyz has probability $T(z;y)T(y;x)\pi(x)$.

29

Monte Carlo Methods

▶ 29.1 The problems to be solved

Monte Carlo methods are computational techniques that make use of random numbers. 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 $P(\mathbf{x})$.

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

$$\Phi = \langle \phi(\mathbf{x}) \rangle \equiv \int \mathrm{d}^N \mathbf{x} \; P(\mathbf{x}) \phi(\mathbf{x}). \tag{29.3}$$

The probability distribution $P(\mathbf{x})$, which we 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.

Simple examples of functions $\phi(\mathbf{x})$ whose expectations we might be interested in include the first and second moments of quantities that we wish to predict, from which we can compute means and variances; for example if some quantity t depends on $\mathbf{x}$, we can find the mean and variance of t under $P(\mathbf{x})$ by finding the expectations of the functions $\phi_1(\mathbf{x}) = t(\mathbf{x})$ and $\phi_2(\mathbf{x}) = (t(\mathbf{x}))^2$,

$$\Phi_1 \equiv \mathcal{E}[\phi_1(\mathbf{x})] \;\; \text{and} \;\; \Phi_2 \equiv \mathcal{E}[\phi_2(\mathbf{x})], \tag{29.4}$$

then using

$$\bar{t} = \Phi_1 \;\; \text{and} \;\; \mathrm{var}(t) = \Phi_2 - \Phi_1^2. \tag{29.5}$$

It is assumed that $P(\mathbf{x})$ is sufficiently complex that we cannot evaluate these expectations by exact methods; so we are interested in Monte Carlo methods.

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

$$\hat{\Phi} \equiv \frac{1}{R} \sum_r \phi(\mathbf{x}^{(r)}). \tag{29.6}$$

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 σ^2/R, where σ^2 is the variance of ϕ,

$$\sigma^2 = \int \mathrm{d}^N \mathbf{x} \; P(\mathbf{x}) (\phi(\mathbf{x}) - \Phi)^2. \tag{29.7}$$

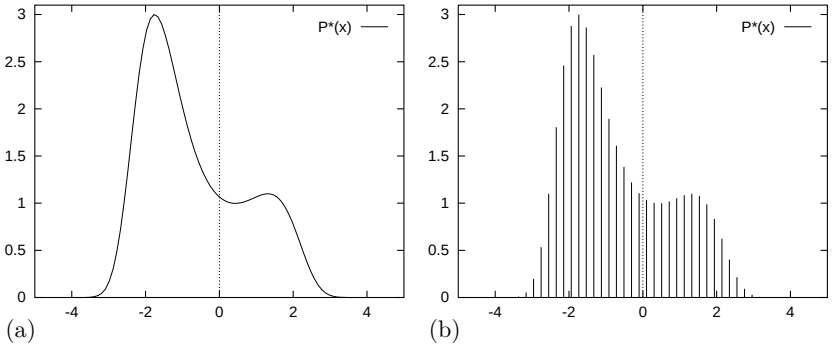

Figure 29.1. (a) The function $P^*(x) = \exp\left[0.4(x-0.4)^2 - 0.08x^4\right]$. How to draw samples from this density? (b) The function $P^*(x)$ evaluated at a discrete set of uniformly spaced points $\{x_i\}$. How to draw samples from this discrete distribution?

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

> The accuracy of the Monte Carlo estimate (29.6) depends only on the variance of ϕ, not on the dimensionality of the space sampled. To be precise, the variance of $\hat{\Phi}$ goes as σ^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.

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

$$P(\mathbf{x}) = P^*(\mathbf{x})/Z. \tag{29.8}$$

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

$$Z = \int \mathrm{d}^N\mathbf{x}\ P^*(\mathbf{x}). \tag{29.9}$$

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, because there is no obvious way to sample from P without enumerating most or all of the possible states. Correct samples from P will by definition tend to come from places in $\mathbf{x}$-space where $P(\mathbf{x})$ is big; how can we identify those places where $P(\mathbf{x})$ is big, without evaluating $P(\mathbf{x})$ *everywhere*? There are only a few high-dimensional densities from which it is easy to draw samples, for example the Gaussian distribution.

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

$$P^*(x) = \exp\left[0.4(x-0.4)^2 - 0.08x^4\right], \quad x \in (-\infty, \infty). \tag{29.10}$$

We can plot this function (figure 29.1a). But that does not mean we can draw samples from it. To start with, we don't know the normalizing constant Z.

To give ourselves a simpler problem, we could discretize the variable x and ask for samples from the discrete probability distribution over a finite set of uniformly spaced points $\{x_i\}$ (figure 29.1b). How could we solve this problem? If we evaluate $p_i^* = P^*(x_i)$ at each point x_i, we can compute

$$Z = \sum_i p_i^* \tag{29.11}$$

and

$$p_i = p_i^*/Z \tag{29.12}$$

and we can then sample from the probability distribution $\{p_i\}$ using various methods based on a source of random bits (see section 6.3). But what is the cost of this procedure, and how does it scale with the dimensionality of the space, N? Let us concentrate on the initial cost of evaluating Z (29.11). To compute Z we have to visit every point in the space. In figure 29.1b 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 took only two discrete values, the number of evaluations of P^* would be 2^{1000}, a number that is still horribly huge. If every electron in the universe (there are about 2^{266} of them) were a 1000 gigahertz computer that could evaluate P^* for a trillion (2^{40}) states every second, and if we ran those 2^{266} computers for a time equal to the age of the universe (2^{58} seconds), they would still only visit 2^{364} states. We'd have to wait for more than $2^{636} \simeq 10^{190}$ universe ages to elapse before all 2^{1000} states had been visited.

Systems with 2^{1000} states are two a penny.[⋆] One example is a collection of 1000 spins such as a 30×30 fragment of an Ising model whose probability distribution is proportional to

$$P^*(\mathbf{x}) = \exp[-\beta E(\mathbf{x})] \tag{29.13}$$

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

$$E(\mathbf{x}) = -\left[\frac{1}{2}\sum_{m,n} J_{mn} x_m x_n + \sum_n H_n x_n\right]. \tag{29.14}$$

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; the first 'exact' samples from this distribution were created in the pioneering work of Propp and Wilson (1996), as we'll describe in Chapter 32.

[⋆] Translation for American readers: 'such systems are a dime a dozen'; incidentally, this equivalence (10c = 6p) shows that the correct exchange rate between our currencies is £1.00 = $1.67.

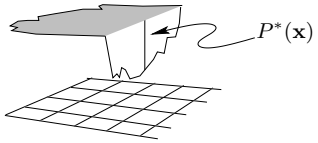

Figure 29.2. A lake whose depth at $\mathbf{x} = (x, y)$ is $P^*(\mathbf{x})$.

A useful analogy

Imagine the tasks of drawing random water samples from a lake and finding the average plankton concentration (figure 29.2). The depth of the lake at $\mathbf{x} = (x, y)$ is $P^*(\mathbf{x})$, and we assert (in order to make the analogy work) that the plankton concentration is a function of $\mathbf{x}$, $\phi(\mathbf{x})$. The required average concentration is an integral like (29.3), namely

$$\Phi = \langle \phi(\mathbf{x}) \rangle \equiv \frac{1}{Z} \int \mathrm{d}^N \mathbf{x} \; P^*(\mathbf{x}) \phi(\mathbf{x}), \tag{29.15}$$

where $Z = \int \mathrm{d}x\, \mathrm{d}y\, P^*(\mathbf{x})$ is the volume of the lake. You are provided with a boat, a satellite navigation system, and a plumbline. Using the navigator, you can can take your boat to any desired location $\mathbf{x}$ on the map; using the plumbline you can measure $P^*(\mathbf{x})$ at that point. You can also measure the plankton concentration there.

Problem 1 is to draw $1\,\mathrm{cm}^3$ water samples at random from the lake, in such a way that each sample is equally likely to come from any point within the lake. Problem 2 is to find the average plankton concentration.

These are difficult problems to solve because at the outset we know nothing about the depth $P^*(\mathbf{x})$. Perhaps much of the volume of the lake is contained in narrow, deep underwater canyons (figure 29.3), in which case, to correctly sample from the lake and correctly estimate Φ our method must implicitly discover the canyons and find their volume relative to the rest of the lake. Difficult problems, yes; nevertheless, we'll see that clever Monte Carlo methods can solve them.

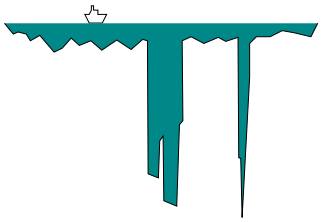

Figure 29.3. A slice through a lake that includes some canyons.

Uniform sampling

Having accepted that we cannot exhaustively 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 a normalizing constant Z_R, defined by

$$Z_R = \sum_{r=1}^{R} P^*(\mathbf{x}^{(r)}), \tag{29.16}$$

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

$$\hat{\Phi} = \sum_{r=1}^{R} \phi(\mathbf{x}^{(r)}) \frac{P^*(\mathbf{x}^{(r)})}{Z_R}. \tag{29.17}$$

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 $P^*(\mathbf{x})$. As we learnt in Chapter 4, 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 entropy of the probability distribution $P(\mathbf{x})$. If almost all the probability mass is located in the typical set and $\phi(\mathbf{x})$ is a benign function, the value of $\Phi = \int \mathrm{d}^N\mathbf{x}\ \phi(\mathbf{x})P(\mathbf{x})$ 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 at least once or twice. So, how many samples are required? Let us take the case of the Ising model again. (Strictly, the Ising model may not be a good example, since it doesn't necessarily have a typical set, as defined in Chapter 4; the definition of a typical set was that all states had log probability close to the entropy, which for an Ising model would mean that the *energy* is very close to the *mean energy*; but in the vicinity of phase transitions, the variance of energy, also known as the heat capacity, may diverge, which means that the energy of a random state is not necessarily expected to be very close to the mean energy.) 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.

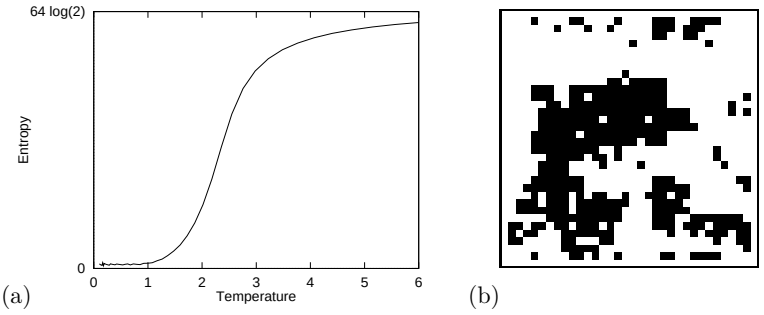

Figure 29.4. (a) Entropy of a 64-spin Ising model as a function of temperature. (b) One state of a 1024-spin Ising model.

The number of samples required to hit the typical set once is thus of order

$$R_{\min} \simeq 2^{N-H}. \tag{29.18}$$

So, what is H? At high temperatures, the probability distribution of an Ising model tends to a uniform distribution and the entropy tends to $H_{\max} = N$ bits, which means $R_{\min}$ 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. The critical temperature of an infinite Ising model, at which it melts, is $\theta_c = 2.27$. At this temperature the entropy of an Ising model is roughly $N/2$ bits (figure 29.4). For this probability distribution the number of samples required simply to hit the typical set once is of order

$$R_{\min} \simeq 2^{N-N/2} = 2^{N/2}, \tag{29.19}$$

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.

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

▶ 29.2 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 $P(x)$. Let us assume that we are able to evaluate this density at any chosen point $\mathbf{x}$, at least to within a multiplicative constant; thus we can evaluate a function $P^*(x)$ such that

$$P(x) = P^*(x)/Z. \tag{29.20}$$

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 $Q(x)$ from which we *can* generate samples and which we can evaluate to within a multiplicative constant (that is, we can evaluate $Q^*(x)$, where $Q(x) = Q^*(x)/Z_Q$). An example of the functions P^*, Q^* and ϕ is shown in figure 29.5. 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 (29.6). 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 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*

$$w_r \equiv \frac{P^*(x^{(r)})}{Q^*(x^{(r)})} \tag{29.21}$$

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

$$\hat{\Phi} \equiv \frac{\sum_r w_r \phi(x^{(r)})}{\sum_r w_r}. \tag{29.22}$$

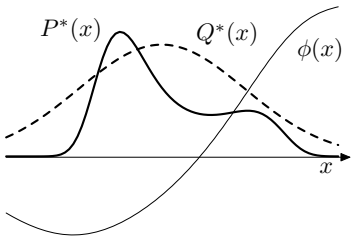

Figure 29.5. Functions involved in importance sampling. We wish to estimate the expectation of $\phi(x)$ under $P(x) \propto P^*(x)$. We can generate samples from the simpler distribution $Q(x) \propto Q^*(x)$. We can evaluate Q^* and P^* at any point.

▷ Exercise 29.1.[2, p.386] Prove that, if $Q(x)$ is non-zero for all x where $P(x)$ is non-zero, the estimator $\hat{\Phi}$ converges to Φ, the mean value of $\phi(x)$, as R increases. What is the variance of this estimator, asymptotically? Hint: consider the statistics of the numerator and the denominator separately. Is the estimator $\hat{\Phi}$ an unbiased estimator for small R?

A practical difficulty with importance sampling is that it is hard to estimate how reliable the estimator $\hat{\Phi}$ is. The variance of the estimator is unknown beforehand, because it depends on an integral over x of a function involving $P^*(x)$. And 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 (29.22). 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. In this case the estimate of Φ would be drastically wrong, and there would be no indication in the *empirical* variance that the true variance of the estimator $\hat{\Phi}$ is large.

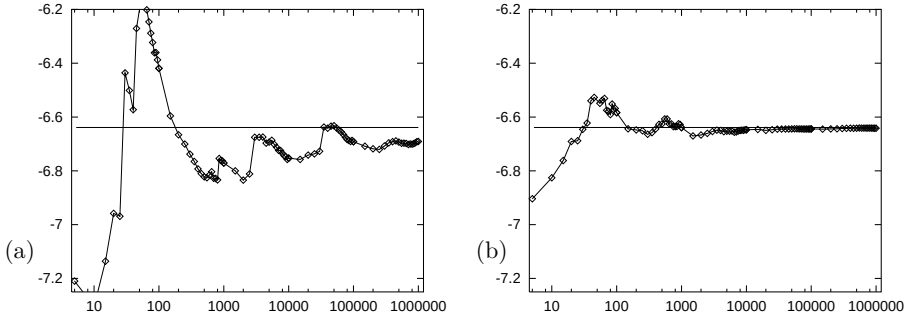

Figure 29.6. Importance sampling in action: (a) using a Gaussian sampler density; (b) using a Cauchy sampler density. Vertical axis shows the estimate $\hat{\Phi}$. The horizontal line indicates the true value of Φ. Horizontal axis shows number of samples on a log scale.

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 29.6. 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; it converges (on the scale shown here) after about 5000 samples.

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

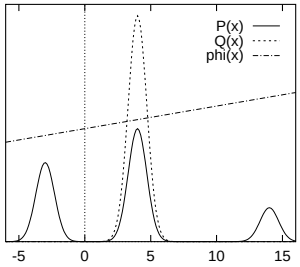

Figure 29.7. A multimodal distribution $P^*(x)$ and a unimodal sampler $Q(x)$.

Exercise 29.2.[2, p.387] Consider the situation where $P^*(x)$ is multimodal, consisting of several widely separated peaks. (Probability distributions like this arise frequently in statistical data modelling.) Discuss whether it is a wise strategy to do importance sampling using a sampler $Q(x)$ that is a unimodal distribution fitted to one of these peaks. Assume that the function $\phi(x)$ whose mean Φ is to be estimated is a smoothly varying function of x such as $mx + c$. Describe the typical evolution of the estimator $\hat{\Phi}$ as a function of the number of samples R.

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,

$$P^*(\mathbf{x}) = \begin{cases} 1 & 0 \leq \rho(\mathbf{x}) \leq R_P \\ 0 & \rho(\mathbf{x}) > R_P, \end{cases} \tag{29.23}$$

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

$$Q(\mathbf{x}) = \prod_i \text{Normal}(x_i; 0, \sigma^2). \tag{29.24}$$

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? We know from our discussions of typical sequences in Part I – see exercise 6.14 (p.124), for example – that 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:

$$\rho^2 \sim N\sigma^2 \pm \sqrt{2N}\sigma^2. \tag{29.25}$$

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

$$\frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{N}{2} \pm \frac{\sqrt{2N}}{2}\right). \tag{29.26}$$

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

$$(2\pi\sigma^2)^{N/2} \exp\left(\frac{N}{2} \pm \frac{\sqrt{2N}}{2}\right). \tag{29.27}$$

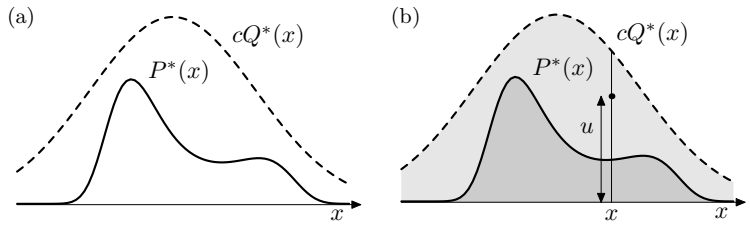

Figure 29.8. Rejection sampling. (a) The functions involved in rejection sampling. We desire samples from $P(x) \propto P^*(x)$. We are able to draw samples from $Q(x) \propto Q^*(x)$, and we know a value c such that $cQ^*(x) > P^*(x)$ for all x. (b) A point (x, u) is generated at random in the lightly shaded area under the curve $cQ^*(x)$. If this point also lies below $P^*(x)$ then it is accepted.

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 (29.27). The largest weight and the median weight will typically be in the ratio:

$$\frac{w_r^{\max}}{w_r^{\text{med}}} = \exp\left(\sqrt{2N}\right). \tag{29.28}$$

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 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 $\exp(\sqrt{N})$, so the weights will too, unless Q is a near-perfect approximation to P.

▶ 29.3 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 from which we can generate samples. We further assume that we know the value of a constant c such that

$$c\,Q^*(x) > P^*(x), \text{ for all } x. \tag{29.29}$$

A schematic picture of the two functions is shown in figure 29.8a.

We generate two random numbers. The first, x, is generated from the proposal density $Q(x)$. We then evaluate $c\,Q^*(x)$ and generate a uniformly distributed random variable u from the interval $[0, c\,Q^*(x)]$. These two random numbers can be viewed as selecting a point in the two-dimensional plane as shown in figure 29.8b.

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 $c\,Q^*(x)$ as shown in figure 29.8b. 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, for cQ to exceed P everywhere, c will necessarily have to be large and the frequency of rejection will be large.

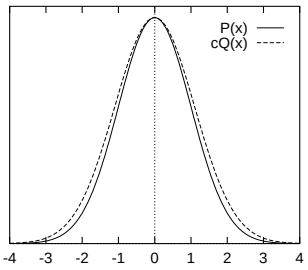

Figure 29.9. A Gaussian $P(x)$ and a slightly broader Gaussian $Q(x)$ scaled up by a factor c such that $cQ(x) \geq P(x)$.

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 know neither where the modes of P^* are located nor how high they are.

As a case study, consider a pair of N-dimensional Gaussian distributions with mean zero (figure 29.9). 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 1% 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 exceeds P for all $\mathbf{x}$.] So, what value of c is required 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 exceed P we need to set

$$c = \frac{(2\pi\sigma_Q^2)^{N/2}}{(2\pi\sigma_P^2)^{N/2}} = \exp\left(N \ln \frac{\sigma_Q}{\sigma_P}\right). \tag{29.30}$$

With $N = 1000$ and $\frac{\sigma_Q}{\sigma_P} = 1.01$, we find $c = \exp(10) \simeq 20{,}000$. What will the acceptance 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 both normalized here implies that the acceptance rate will be $1/c$, for example, 1/20,000. In general, c grows exponentially with the dimensionality N, so the acceptance rate is expected to be exponentially small in N.

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

▶ 29.4 The Metropolis–Hastings method

Importance sampling and rejection sampling work well only 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.

The Metropolis–Hastings algorithm instead makes use of a proposal density Q *which depends on the current state* $x^{(t)}$. The density $Q(x'; x^{(t)})$ might 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 from which we can draw samples. In contrast to importance sampling and rejection sampling, it is not necessary that $Q(x'; x^{(t)})$ look at all similar to $P(x)$ in order for the algorithm to be practically useful. An example of a proposal density is shown in figure 29.10; 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

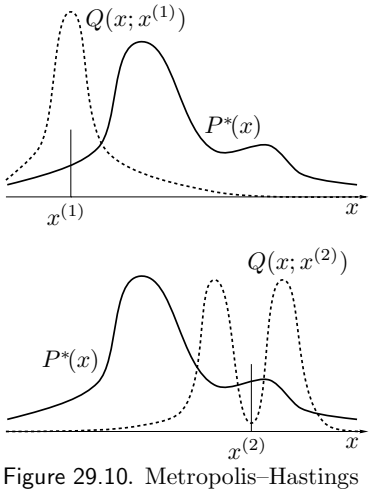

Figure 29.10. Metropolis–Hastings method in one dimension. The proposal distribution $Q(x'; x)$ is here shown as having a shape that changes as x changes, though this is not typical of the proposal densities used in practice.

whether to accept the new state, we compute the quantity

$$a = \frac{P^*(x')}{P^*(x^{(t)})} \frac{Q(x^{(t)}; x')}{Q(x'; x^{(t)})}. \tag{29.31}$$

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

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 again onto the list.

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–Hastings simulation of T iterations does not produce T *independent* samples from the target distribution P. The samples are correlated.

To compute the acceptance probability (29.31) 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–Hastings method simply involves comparing the value of the target density at the two points. This special case is sometimes called the Metropolis method. However, with apologies to Hastings, I will call the general Metropolis–Hastings algorithm for asymmetric Q 'the Metropolis algorithm' since I believe important ideas deserve short names.

Convergence of the Metropolis method to the target density

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.

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

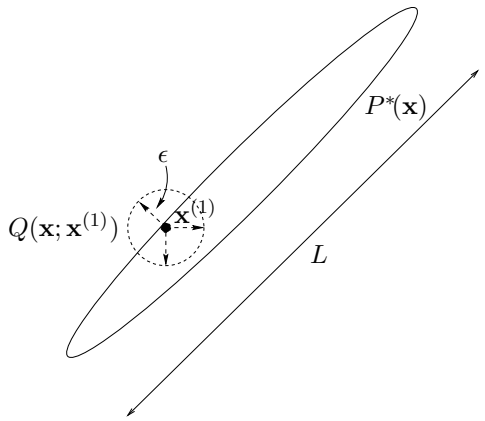

Figure 29.11. Metropolis method in two dimensions, showing a traditional proposal density that has a sufficiently small step size ϵ that the acceptance frequency will be about 0.5.

with a length scale ϵ that is short relative to the longest length scale L of the probable region (figure 29.11). 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 that has very low probability; such steps are unlikely to be accepted. If ϵ is large, movement around the state space will only occur when such a transition to a low-probability state is actually accepted, or when a large random step chances to land in another probable state. So the rate of progress will be slow if large 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 a random walk takes a long time to get anywhere, especially if the walk is made of small steps.

Exercise 29.3.[1] Consider a one-dimensional random walk, on each step of which the state moves randomly to the left or to the right with equal probability. Show that after T steps of size ϵ, the state is likely to have moved only a distance about $\sqrt{T}\epsilon$. (Compute the root mean square distance travelled.)

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

$$T \simeq (L/\epsilon)^2 \tag{29.32}$$

iterations to obtain an independent sample.

This rule of thumb gives only 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.

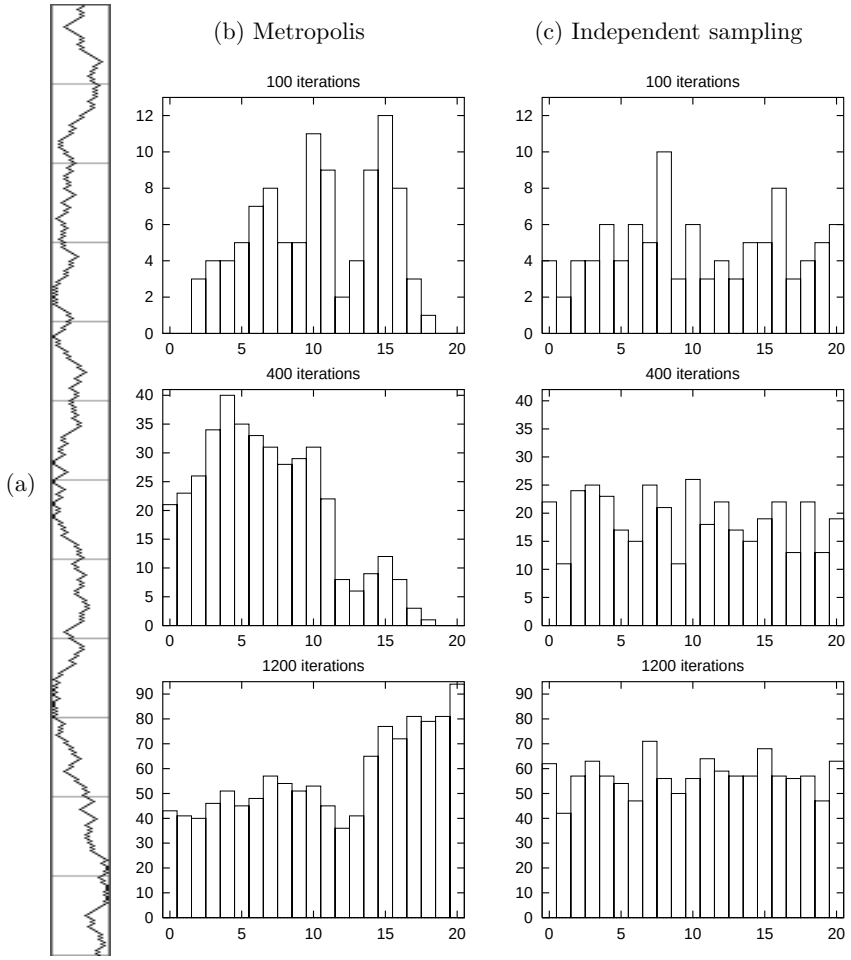

Figure 29.12. Metropolis method for a toy problem. (a) The state sequence for $t = 1, \ldots, 600$. Horizontal direction = states from 0 to 20; vertical direction = time from 1 to 600; the cross bars mark time intervals of duration 50. (b) Histogram of occupancy of the states after 100, 400 and 1200 iterations. (c) For comparison, histograms resulting when successive points are drawn *independently* from the target distribution.

To illustrate how slowly a random walk explores a state space, figure 29.12 shows a simulation of a Metropolis algorithm for generating samples from the distribution:

$$P(x) = \begin{cases} 1/21 & x \in \{0, 1, 2, \ldots, 20\} \\ 0 & \text{otherwise.} \end{cases} \tag{29.33}$$

The proposal distribution is

$$Q(x'; x) = \begin{cases} 1/2 & x' = x \pm 1 \\ 0 & \text{otherwise.} \end{cases} \tag{29.34}$$

Because the target distribution $P(x)$ is uniform, rejections 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 29.12a. 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 (29.32) 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 per independent sample.

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. Methods for reducing random walk behaviour are discussed in the next chapter.

Metropolis method in high dimensions

The rule of thumb (29.32), which gives a lower bound on the number of iterations of a random walk Metropolis method, also applies to higher dimensional problems. Consider the simple case of a target distribution that is an N-dimensional Gaussian, and a proposal distribution that is a spherical Gaussian of standard deviation ϵ in each direction. 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 deviation σ_n in direction 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 frequency is close to 1. Under this assumption, each variable x_n evolves independently of all the others, executing a random walk with step size 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 $\sigma^{\min}$. 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.

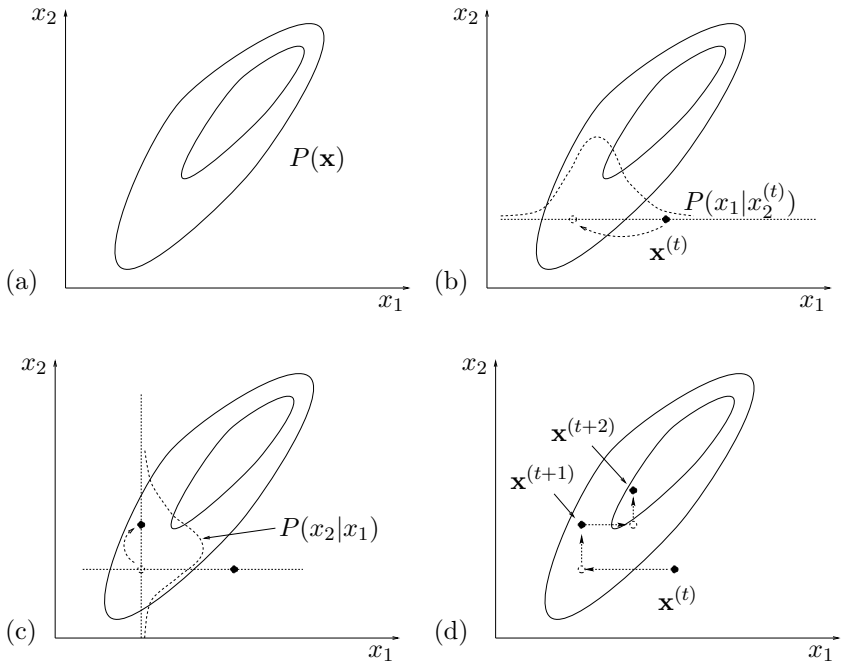

Figure 29.13. Gibbs sampling. (a) The joint density $P(\mathbf{x})$ from which samples are required. (b) Starting from a state $\mathbf{x}^{(t)}$, x_1 is sampled from the conditional density $P(x_1|x_2^{(t)})$. (c) A sample is then made from the conditional density $P(x_2|x_1)$. (d) A couple of iterations of Gibbs sampling.

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. Our computer *will* give useful answers in a time shorter than the age of the universe. But it is bad news all the same, because this quadratic dependence on the lengthscale-ratio may still force us to make very lengthy simulations.

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

▶ 29.5 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* or 'Glauber dynamics', is a method for sampling from distributions over at least two dimensions. Gibbs sampling can be viewed as a Metropolis method in which a sequence of proposal distributions Q are 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. For example, if a Gaussian distribution for some variables $\mathbf{d}$ has an unknown mean $\mathbf{m}$, and the prior distribution of $\mathbf{m}$ is Gaussian, then the conditional distribution of $\mathbf{m}$ given $\mathbf{d}$ is also Gaussian. 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 29.13. 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:

$$x_1^{(t+1)} \sim P(x_1 \,|\, x_2^{(t)}, x_3^{(t)}, \ldots, x_K^{(t)}) \quad (29.35)$$
$$x_2^{(t+1)} \sim P(x_2 \,|\, x_1^{(t+1)}, x_3^{(t)}, \ldots, x_K^{(t)}) \quad (29.36)$$
$$x_3^{(t+1)} \sim P(x_3 \,|\, x_1^{(t+1)}, x_2^{(t+1)}, \ldots, x_K^{(t)}), \text{ etc.} \quad (29.37)$$

Convergence of Gibbs sampling to the target density

▷ Exercise 29.4.[2] Show that a single variable-update of Gibbs sampling can be viewed as a Metropolis method with target density $P(\mathbf{x})$, and that this Metropolis method 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.

▷ Exercise 29.5.[2, p.387] Discuss whether the syndrome decoding problem for a $(7, 4)$ Hamming code can be solved using Gibbs sampling. The syndrome decoding problem, if we are to solve it with a Monte Carlo approach, is to draw samples from the posterior distribution of the noise vector $\mathbf{n} = (n_1, \ldots, n_n, \ldots, n_N)$,

$$P(\mathbf{n} \,|\, \mathbf{f}, \mathbf{z}) = \frac{1}{Z} \prod_{n=1}^{N} f_n^{n_n} (1 - f_n)^{(1-n_n)} \mathbb{1}[\mathbf{Hn} = \mathbf{z}], \quad (29.38)$$

where f_n is the normalized likelihood for the nth transmitted bit and $\mathbf{z}$ is the observed syndrome. The factor $\mathbb{1}[\mathbf{Hn} = \mathbf{z}]$ is 1 if $\mathbf{n}$ has the correct syndrome $\mathbf{z}$ and 0 otherwise.

What about the syndrome decoding problem for any linear error-correcting code?

Gibbs sampling in high dimensions

Gibbs sampling suffers from the same defect as simple Metropolis algorithms – the state space is explored by a slow random walk, unless a fortuitous parameterization has been chosen that 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. Figure 30.3, p.392, illustrates the slow progress made by Gibbs sampling when $L \gg \epsilon$.

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`, makes it easy to set up almost arbitrary probabilistic models and simulate them by Gibbs sampling (Thomas *et al.*, 1992).[1]

[1] `http://www.mrc-bsu.cam.ac.uk/bugs/`

▶ 29.6 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. We denote by $p^{(t)}(\mathbf{x})$ the probability distribution of the state of a Markov chain simulator. (To visualize this distribution, imagine running an infinite collection of identical simulators in parallel.) Our aim is to find a Markov chain such that as $t \to \infty$, $p^{(t)}(\mathbf{x})$ tends to the desired distribution $P(\mathbf{x})$.

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, $p^{(t+1)}(\mathbf{x})$, is given by

$$p^{(t+1)}(\mathbf{x}') = \int \mathrm{d}^N\mathbf{x}\ T(\mathbf{x}'; \mathbf{x}) p^{(t)}(\mathbf{x}). \tag{29.39}$$

Example 29.6. An example of a Markov chain is given by the Metropolis demonstration of section 29.4 (figure 29.12), for which the transition probability is

$$\mathbf{T} = \begin{bmatrix}
1/2 & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & . & 1/2 \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1/2 & 1/2
\end{bmatrix}$$

and the initial distribution was

$$p^{(0)}(x) = \begin{bmatrix} . & . & . & . & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . \end{bmatrix}. \tag{29.40}$$

The probability distribution $p^{(t)}(x)$ of the state at the tth iteration is shown for $t = 0$, 1, 2, 3, 5, 10, 100, 200, 400 in figure 29.14; an equivalent sequence of distributions is shown in figure 29.15 for the chain that begins in initial state $x_0 = 17$. Both chains converge to the target density, the uniform density, as $t \to \infty$.

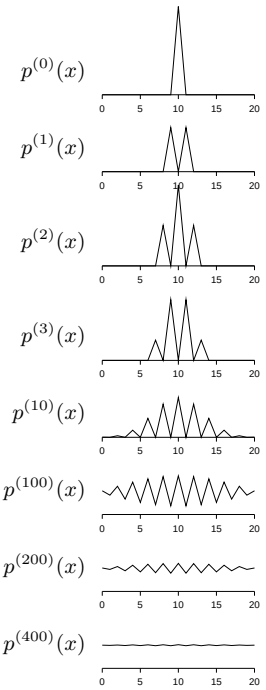

Figure 29.14. The probability distribution of the state of the Markov chain of example 29.6.

Required properties

When designing a Markov chain Monte Carlo method, we construct a chain with the following properties:

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

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

   $$\pi(\mathbf{x}') = \int \mathrm{d}^N\mathbf{x}\ T(\mathbf{x}'; \mathbf{x}) \pi(\mathbf{x}). \tag{29.41}$$

   An invariant distribution is an eigenvector of the transition probability matrix that has eigenvalue 1.

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

$$p^{(t)}(\mathbf{x}) \to \pi(\mathbf{x}) \text{ as } t \to \infty, \text{ for any } p^{(0)}(\mathbf{x}). \tag{29.42}$$

A couple of reasons why a chain might not be ergodic are:

(a) Its matrix might be *reducible*, which means that the state space contains two or more subsets of states that can never be reached from each other. Such a chain has many invariant distributions; which one $p^{(t)}(\mathbf{x})$ would tend to as $t \to \infty$ would depend on the initial condition $p^{(0)}(\mathbf{x})$.

The transition probability matrix of such a chain has more than one eigenvalue equal to 1.

(b) The chain might have a *periodic* set, which means that, for some initial conditions, $p^{(t)}(\mathbf{x})$ doesn't tend to an invariant distribution, but instead tends to a periodic limit-cycle.

A simple Markov chain with this property is the random walk on the N-dimensional hypercube. The chain T takes the state from one corner to a randomly chosen adjacent corner. The unique invariant distribution of this chain is the uniform distribution over all 2^N states, but the chain is not ergodic; it is periodic with period two: if we divide the states into states with odd parity and states with even parity, we notice that every odd state is surrounded by even states and *vice versa*. So if the initial condition at time $t = 0$ is a state with even parity, then at time $t = 1$ – and at all odd times – the state must have odd parity, and at all even times, the state will be of even parity.

The transition probability matrix of such a chain has more than one eigenvalue with magnitude equal to 1. The random walk on the hypercube, for example, has eigenvalues equal to $+1$ and -1.

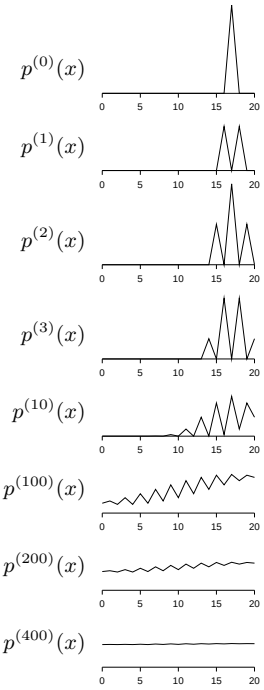

Figure 29.15. The probability distribution of the state of the Markov chain for initial condition $x_0 = 17$ (example 29.6 (p.374)).

Methods of construction of Markov chains

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

$$P(\mathbf{x}') = \int \mathrm{d}^N\mathbf{x}\ B(\mathbf{x}';\mathbf{x})P(\mathbf{x}), \tag{29.43}$$

for the desired density $P(\mathbf{x})$, i.e., they all have the desired density as an invariant distribution. These base transitions need not individually be ergodic.

T is a *mixture* of several base transitions $B_b(\mathbf{x}', \mathbf{x})$ if we make the transition by picking one of the base transitions at random, and allowing it to determine the transition, i.e.,

$$T(\mathbf{x}', \mathbf{x}) = \sum_b p_b B_b(\mathbf{x}', \mathbf{x}), \tag{29.44}$$

where $\{p_b\}$ is a probability distribution over the base transitions.

T is a *concatenation* of two base transitions $B_1(\mathbf{x}', \mathbf{x})$ and $B_2(\mathbf{x}', \mathbf{x})$ if we first make a transition to an intermediate state $\mathbf{x}''$ using B_1, and then make a transition from state $\mathbf{x}''$ to $\mathbf{x}'$ using B_2.

$$T(\mathbf{x}', \mathbf{x}) = \int \mathrm{d}^N\mathbf{x}''\ B_2(\mathbf{x}', \mathbf{x}'')B_1(\mathbf{x}'', \mathbf{x}). \tag{29.45}$$

Detailed balance

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

$$T(\mathbf{x}_a; \mathbf{x}_b)P(\mathbf{x}_b) = T(\mathbf{x}_b; \mathbf{x}_a)P(\mathbf{x}_a), \text{ for all } \mathbf{x}_b \text{ and } \mathbf{x}_a. \tag{29.46}$$

This equation says that if we pick (by magic) 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 $\mathbf{x}_b$ and go from $\mathbf{x}_b$ to $\mathbf{x}_a$ as it is that we will pick $\mathbf{x}_a$ and go from $\mathbf{x}_a$ to $\mathbf{x}_b$. 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, which is a necessary condition for the key property that we want from our MCMC simulation – that the probability distribution of the chain should converge to $P(\mathbf{x})$.

▷ Exercise 29.7.[2] Prove that detailed balance implies invariance of the distribution $P(\mathbf{x})$ under the Markov chain T.

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 satisfies 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, because they may have different random walk properties.

▷ Exercise 29.8.[2] Show that, if we concatenate two base transitions B_1 and B_2 which satisfy detailed balance, it is not necessarily the case that the T thus defined (29.45) satisfies detailed balance.

Exercise 29.9.[2] Does Gibbs sampling, with several variables all updated in a deterministic sequence, satisfy detailed balance?

▶ 29.7 Slice sampling

Slice sampling (Neal, 1997a; Neal, 2003) is a Markov chain Monte Carlo method that has similarities to rejection sampling, Gibbs sampling and the Metropolis method. It can be applied wherever the Metropolis method can be applied, that is, to any system for which the target density $P^*(\mathbf{x})$ can be evaluated at any point $\mathbf{x}$; it has the advantage over simple Metropolis methods that it is more robust to the choice of parameters like step sizes. The simplest version of slice sampling is similar to Gibbs sampling in that it consists of one-dimensional transitions in the state space; however there is no requirement that the one-dimensional conditional distributions be easy to sample from, nor that they have any convexity properties such as are required for adaptive rejection sampling. And slice sampling is similar to rejection sampling in that it is a method that asymptotically draws samples from the volume under the curve described by $P^*(\mathbf{x})$; but there is no requirement for an upper-bounding function.

I will describe slice sampling by giving a sketch of a one-dimensional sampling algorithm, then giving a pictorial description that includes the details that make the method valid.

The skeleton of slice sampling

Let us assume that we want to draw samples from $P(x) \propto P^*(x)$ where x is a real number. A one-dimensional slice sampling algorithm is a method for making transitions from a two-dimensional point (x, u) lying under the curve $P^*(x)$ to another point (x', u') lying under the same curve, such that the probability distribution of (x, u) tends to a uniform distribution over the area under the curve $P^*(x)$, whatever initial point we start from – like the uniform distribution under the curve $P^*(x)$ produced by rejection sampling (section 29.3).

A single transition $(x, u) \rightarrow (x', u')$ of a one-dimensional slice sampling algorithm has the following steps, of which steps 3 and 8 will require further elaboration.

1: evaluate $P^*(x)$
2: draw a vertical coordinate $u' \sim \text{Uniform}(0, P^*(x))$
3: create a horizontal interval (x_l, x_r) enclosing x
4: loop {
5: draw $x' \sim \text{Uniform}(x_l, x_r)$
6: evaluate $P^*(x')$
7: if $P^*(x') > u'$ break out of loop 4-9
8: else modify the interval (x_l, x_r)
9: }

There are several methods for creating the interval (x_l, x_r) in step 3, and several methods for modifying it at step 8. The important point is that the overall method must satisfy detailed balance, so that the uniform distribution for (x, u) under the curve $P^*(x)$ is invariant.

The 'stepping out' method for step 3

In the 'stepping out' method for creating an interval (x_l, x_r) enclosing x, we step out in steps of length w until we find endpoints x_l and x_r at which P^* is smaller than u. The algorithm is shown in figure 29.16.

3a: draw $r \sim \text{Uniform}(0, 1)$
3b: x_l := $x - rw$
3c: x_r := $x + (1 - r)w$
3d: while $(P^*(x_l) > u)$ { $x_l := x_l - w$ }
3e: while $(P^*(x_r) > u)$ { $x_r := x_r + w$ }

The 'shrinking' method for step 8

Whenever a point x' is drawn such that (x', u') lies above the curve $P^*(x)$, we shrink the interval so that one of the end points is x', and such that the original point x is still enclosed in the interval.

8a: if $(x' > x)$ { x_r := x' }
8b: else { x_l := x' }

Properties of slice sampling

Like a standard Metropolis method, slice sampling gets around by a random walk, but whereas in the Metropolis method, the choice of the step size is

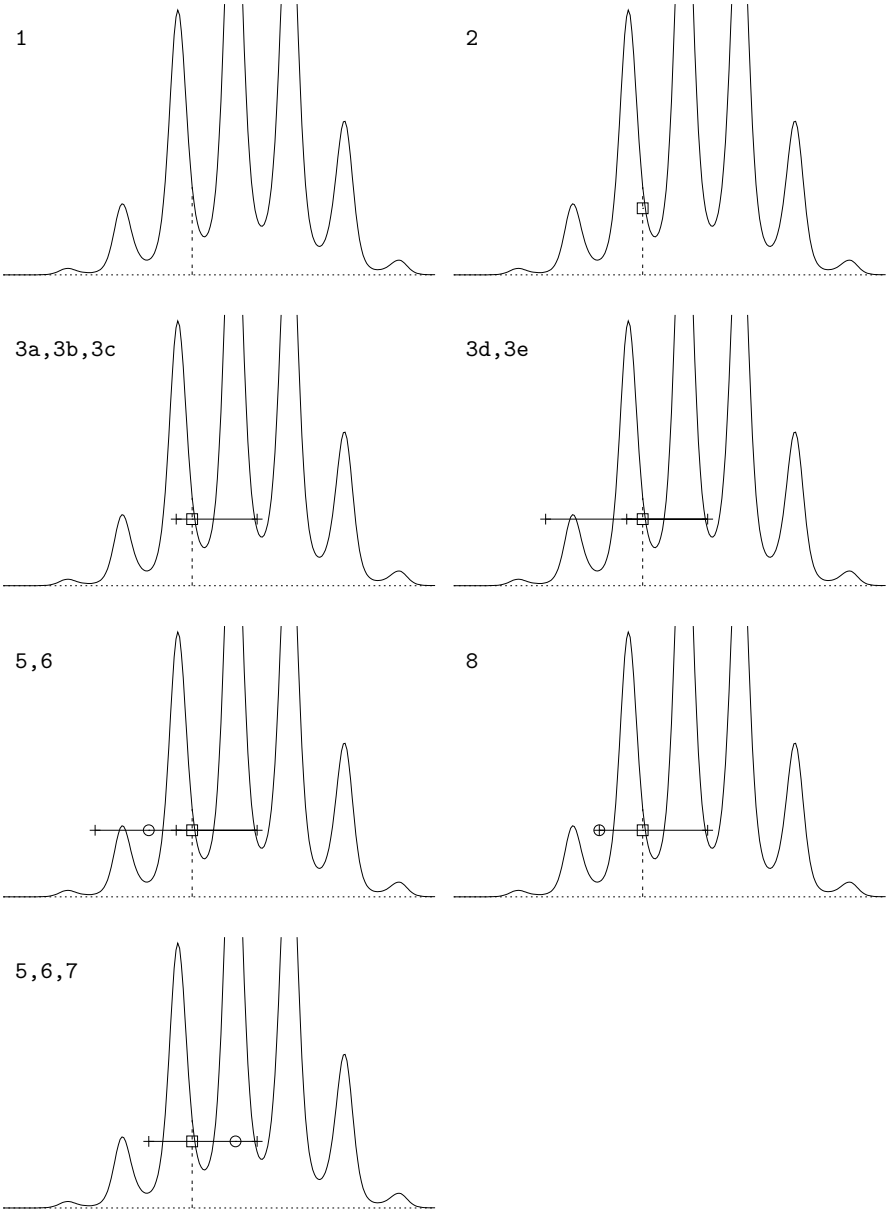

Figure 29.16. Slice sampling. Each panel is labelled by the steps of the algorithm that are executed in it. At step 1, $P^*(x)$ is evaluated at the current point x. At step 2, a vertical coordinate is selected giving the point (x, u') shown by the box; At steps 3a-c, an interval of size w containing (x, u') is created at random. At step 3d, P^* is evaluated at the left end of the interval and is found to be larger than u', so a step to the left of size w is made. At step 3e, P^* is evaluated at the right end of the interval and is found to be smaller than u', so no stepping out to the right is needed. When step 3d is repeated, P^* is found to be smaller than u', so the stepping out halts. At step 5 a point is drawn from the interval, shown by a $\circ$. Step 6 establishes that this point is above P^* and step 8 shrinks the interval to the rejected point in such a way that the original point x is still in the interval. When step 5 is repeated, the new coordinate x' (which is to the right-hand side of the interval) gives a value of P^* greater than u', so this point x' is the outcome at step 7.

critical to the rate of progress, in slice sampling the step size is self-tuning. If the initial interval size w is too small by a factor f compared with the width of the probable region then the stepping-out procedure expands the interval size. The cost of this stepping-out is only linear in f, whereas in the Metropolis method the computer-time scales as the square of f if the step size is too small.

If the chosen value of w is too large by a factor F then the algorithm spends a time proportional to the logarithm of F shrinking the interval down to the right size, since the interval typically shrinks by a factor in the ballpark of 0.6 each time a point is rejected. In contrast, the Metropolis algorithm responds to a too-large step size by rejecting almost all proposals, so the rate of progress is exponentially bad in F. There are no rejections in slice sampling. The probability of staying in exactly the same place is very small.

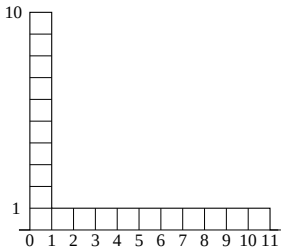

Figure 29.17. $P^*(x)$.

▷ Exercise 29.10.[2] Investigate the properties of slice sampling applied to the density shown in figure 29.17. x is a real variable between 0.0 and 11.0. How long does it take typically for slice sampling to get from an x in the peak region $x \in (0, 1)$ to an x in the tail region $x \in (1, 11)$, and *vice versa*? Confirm that the probabilities of these transitions do yield an asymptotic probability density that is correct.

How slice sampling is used in real problems

An N-dimensional density $P(\mathbf{x}) \propto P^*(\mathbf{x})$ may be sampled with the help of one-dimensional slice sampling method presented above by picking a sequence of directions $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots$ and defining $\mathbf{x} = \mathbf{x}^{(t)} + x\mathbf{y}^{(t)}$. The function $P^*(x)$ above is replaced by $P^*(\mathbf{x}) = P^*(\mathbf{x}^{(t)} + x\mathbf{y}^{(t)})$. The directions may be chosen in various ways; for example, as in Gibbs sampling, the directions could be the coordinate axes; alternatively, the directions $\mathbf{y}^{(t)}$ may be selected at random in any manner such that the overall procedure satisfies detailed balance.

Computer-friendly slice sampling

The real variables of a probabilistic model will always be represented in a computer using a finite number of bits. In the following implementation of slice sampling due to Skilling, the stepping-out, randomization, and shrinking operations, described above in terms of floating-point operations, are replaced by binary and integer operations.

We assume that the variable x that is being slice-sampled is represented by a b-bit integer X taking on one of $B = 2^b$ values, $0, 1, 2, \ldots, B-1$, many or all of which correspond to valid values of x. Using an integer grid eliminates any errors in detailed balance that might ensue from variable-precision rounding of floating-point numbers. The mapping from X to x need not be linear; if it is nonlinear, we assume that the function $P^*(x)$ is replaced by an appropriately transformed function – for example, $P^{**}(X) \propto P^*(x)|\mathrm{d}x/\mathrm{d}X|$.

We assume the following operators on b-bit integers are available:

$X + N$	arithmetic sum, modulo B, of X and N.
$X - N$	difference, modulo B, of X and N.
$X \oplus N$	bitwise exclusive-or of X and N.
$N :=$ `randbits`(l)	sets N to a random l-bit integer.

A slice-sampling procedure for integers is then as follows:

Given: a current point X and a height $Y = P^*(X) \times \text{Uniform}(0,1) \leq P^*(X)$

1:	$U := \texttt{randbits}(b)$	Define a random translation U of the binary coordinate system.
2:	set l to a value $l \leq b$	Set initial l-bit sampling range.
3:	do {	
4:	$N := \texttt{randbits}(l)$	Define a random move within the current interval of width 2^l.
5:	$X' := ((X - U) \oplus N) + U$	Randomize the lowest l bits of X (in the translated coordinate system).
6:	$l := l - 1$	If X' is not acceptable, decrease l and try again
7:	} until $(X' = X)$ or $(P^*(X) \geq Y)$	with a smaller perturbation of X; termination at or before $l = 0$ is assured.

The translation U is introduced to avoid permanent sharp edges, where for example the adjacent binary integers `0111111111` and `1000000000` would otherwise be permanently in different sectors, making it difficult for X to move from one to the other.

The sequence of intervals from which the new candidate points are drawn is illustrated in figure 29.18. First, a point is drawn from the entire interval, shown by the top horizontal line. At each subsequent draw, the interval is halved in such a way as to contain the previous point X.

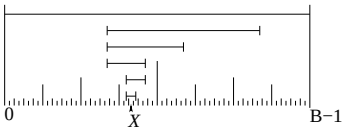

Figure 29.18. The sequence of intervals from which the new candidate points are drawn.

If preliminary stepping-out from the initial range is required, step 2 above can be replaced by the following similar procedure:

2a:	set l to a value $l < b$	l sets the initial width
2b:	do {	
2c:	$N := \texttt{randbits}(l)$	
2d:	$X' := ((X - U) \oplus N) + U$	
2e:	$l := l + 1$	
2f:	} until $(l = b)$ or $(P^*(X) < Y)$	

These shrinking and stepping out methods shrink and expand by a factor of two per evaluation. A variant is to shrink or expand by more than one bit each time, setting $l := l \pm \Delta l$ with $\Delta l > 1$. Taking Δl at each step from any pre-assigned distribution (which may include $\Delta l = 0$) allows extra flexibility.

Exercise 29.11.[4] In the shrinking phase, after an unacceptable X' has been produced, the choice of Δl is allowed to depend on the difference between the slice's height Y and the value of $P^*(X')$, without spoiling the algorithm's validity. (Prove this.) It might be a good idea to choose a larger value of Δl when $Y - P^*(X)$ is large. Investigate this idea theoretically or empirically.

A feature of using the integer representation is that, with a suitably extended number of bits, the single integer X can represent two or more real parameters – for example, by mapping X to (x_1, x_2, x_3) through a space-filling curve such as a Peano curve. Thus multi-dimensional slice sampling can be performed using the same software as for one dimension.

▶ 29.8 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 giving upper bounds on the convergence time are of little practical use. The exact sampling methods of Chapter 32 offer a solution to this problem for certain Markov chains.

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, as described in the next chapter, which describes the Hamiltonian Monte Carlo method, overrelaxation, and simulated annealing.

▶ 29.9 Further practical issues

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 (1993b)) and annealed importance sampling (Neal, 1998).

2. 'Thermodynamic integration' during simulated annealing, the 'acceptance ratio' method, and 'umbrella sampling' (reviewed by Neal (1993b)).

3. 'Reversible jump Markov chain Monte Carlo' (Green, 1995).

One way of dealing with Z, however, may be to find a solution to one's task that does not require that Z be evaluated. In Bayesian data modelling one might be able to 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). In noting the possibility of not computing Z, I am not endorsing this approach. The normalizing constant Z is often the single most important number in the problem, and I think every effort should be devoted to calculating it.

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.

▷ Exercise 29.12.[2, p.387] Explain why the rate of movement through the state space will be greater when B proposals $Q^{(1)}, \ldots, Q^{(B)}$ are considered *individually* in sequence, compared with the case of a single proposal Q^* defined by the concatenation of $Q^{(1)}, \ldots, Q^{(B)}$. Assume that each proposal distribution $Q^{(b)}(\mathbf{x}'; \mathbf{x})$ has an acceptance rate $f < 1/2$.

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 to aim 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.

Gibbs sampling in big models

Our description of Gibbs sampling involved sampling one parameter at a time, as described in equations (29.35–29.37). For big problems it may be more efficient to sample *groups* of variables jointly, that is to use several proposal distributions:

$$\begin{aligned} x_1^{(t+1)}, \ldots, x_a^{(t+1)} &\sim P(x_1, \ldots, x_a \,|\, x_{a+1}^{(t)}, \ldots, x_K^{(t)}) \qquad (29.47) \\ x_{a+1}^{(t+1)}, \ldots, x_b^{(t+1)} &\sim P(x_{a+1}, \ldots, x_b \,|\, x_1^{(t+1)}, \ldots, x_a^{(t+1)}, x_{b+1}^{(t)}, \ldots, x_K^{(t)}), \quad \text{etc.} \end{aligned}$$

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

$$\sigma^2 = \int \mathrm{d}^N \mathbf{x} \; P(\mathbf{x})(\phi(\mathbf{x}) - \Phi)^2. \qquad (29.48)$$

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 Cambridge, 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.

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.

Figure 29.19. Three possible Markov chain Monte Carlo strategies for obtaining twelve samples in a fixed amount of computer time. 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.

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 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 29.19):

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 is popular with Markov chain Monte Carlo experts (Gilks *et al.*, 1996) 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.

Finally, I should emphasize that there is no need to make the points nearly-independent. Averaging over dependent points is fine – it won't lead to any bias in the estimates. For example, when you use strategy 1 or 2, you may, if you wish, include all the points between the first and last sample in each run. Of course, estimating the accuracy of the estimate is harder when the points are dependent.

▶ 29.10 Summary

- Monte Carlo methods are a powerful tool that allow one to sample from 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

$$\int \phi(\mathbf{x})P(\mathbf{x}) \simeq \frac{1}{R}\sum_r \phi(\mathbf{x}^{(r)}). \tag{29.49}$$

- In high-dimensional problems the only satisfactory methods are those based on Markov chains, such as the Metropolis method, Gibbs sampling and slice sampling. Gibbs sampling is an attractive method because it has no adjustable parameters but its use is restricted to cases where samples can be generated from the conditional distributions. Slice sampling is attractive because, whilst it has step length parameters, its performance is not very sensitive to their values.

- Simple Metropolis algorithms and Gibbs sampling algorithms, although widely used, perform poorly because they explore the space by a slow random walk. The next chapter will discuss methods for speeding up Markov chain Monte Carlo simulations.

- Slice sampling does not avoid random walk behaviour, but it automatically chooses the largest appropriate step size, thus reducing the bad effects of the random walk compared with, say, a Metropolis method with a tiny step size.

▶ 29.11 Exercises

Exercise 29.13.[2C, p.388] A study of importance sampling. We already established in section 29.2 that importance sampling is likely to be useless in high-dimensional problems. This exercise explores a further cautionary tale, showing that importance sampling can fail even in one dimension, even with friendly Gaussian distributions.

Imagine that we want to know the expectation of a function $\phi(x)$ under a distribution $P(x)$,

$$\Phi = \int \mathrm{d}x\, P(x)\phi(x), \tag{29.50}$$

and that this expectation is estimated by importance sampling with a distribution $Q(x)$. Alternatively, perhaps we wish to estimate the normalizing constant Z in $P(x) = P^*(x)/Z$ using

$$Z = \int \mathrm{d}x\, P^*(x) = \int \mathrm{d}x\, Q(x)\frac{P^*(x)}{Q(x)} = \left\langle \frac{P^*(x)}{Q(x)} \right\rangle_{x\sim Q}. \tag{29.51}$$

Now, let $P(x)$ and $Q(x)$ be Gaussian distributions with mean zero and standard deviations σ_p and σ_q. Each point x drawn from Q will have an associated weight $P^*(x)/Q(x)$. What is the variance of the weights? [Assume that $P^* = P$, so P is actually normalized, and $Z = 1$, though we can pretend that we didn't know that.] What happens to the variance of the weights as $\sigma_q^2 \to \sigma_p^2/2$?

Check your theory by simulating this importance-sampling problem on a computer.

Exercise 29.14.[2] Consider the Metropolis algorithm for the one-dimensional toy problem of section 29.4, sampling from $\{0, 1, \ldots, 20\}$. Whenever the current state is one of the end states, the proposal density given in equation (29.34) will propose with probability 50% a state that will be rejected.

To reduce this 'waste', Fred modifies the software responsible for generating samples from Q so that when $x = 0$, the proposal density is 100% on $x' = 1$, and similarly when $x = 20$, $x' = 19$ is always proposed.

Fred sets the software that implements the acceptance rule so that the software accepts all proposed moves. What probability $P'(x)$ will Fred's modified software generate samples from?

What is the correct acceptance rule for Fred's proposal density, in order to obtain samples from $P(x)$?

▷ Exercise 29.15.[3C] Implement Gibbs sampling for the inference of a single one-dimensional Gaussian, which we studied using maximum likelihood in section 22.1. Assign a broad Gaussian prior to μ and a broad gamma prior (24.2) to the precision parameter $\beta = 1/\sigma^2$. Each update of μ will involve a sample from a Gaussian distribution, and each update of σ requires a sample from a gamma distribution.

Exercise 29.16.[3C] Gibbs sampling for clustering. Implement Gibbs sampling for the inference of a mixture of K one-dimensional Gaussians, which we studied using maximum likelihood in section 22.2. Allow the clusters to have different standard deviations σ_k. Assign priors to the means and standard deviations in the same way as the previous exercise. Either fix the prior probabilities of the classes $\{\pi_k\}$ to be equal or put a uniform prior over the parameters π and include them in the Gibbs sampling.

Notice the similarity of Gibbs sampling to the soft K-means clustering algorithm (algorithm 22.2). We can alternately *assign* the class labels $\{k_n\}$ given the parameters $\{\mu_k, \sigma_k\}$, then *update* the parameters given the class labels. The assignment step involves sampling from the probability distributions defined by the responsibilities (22.22), and the update step updates the means and variances using probability distributions centred on the K-means algorithm's values (22.23, 22.24).

Do your experiments confirm that Monte Carlo methods bypass the overfitting difficulties of maximum likelihood discussed in section 22.4?

A solution to this exercise and the previous one, written in `octave`, is available.[2]

▷ Exercise 29.17.[3C] Implement Gibbs sampling for the seven scientists inference problem, which we encountered in exercise 22.15 (p.311), and which you may have solved by exact marginalization (exercise 24.3 (p.325)) [it's not essential to have done the latter].

▷ Exercise 29.18.[2] A Metropolis method is used to explore a distribution $P(\mathbf{x})$ that is actually a 1000-dimensional spherical Gaussian distribution of standard deviation 1 in all dimensions. The proposal density Q is a 1000-dimensional spherical Gaussian distribution of standard deviation ϵ. Roughly what is the step size ϵ if the acceptance rate is 0.5? Assuming this value of ϵ,

(a) roughly how long would the method take to traverse the distribution and generate a sample independent of the initial condition?

(b) By how much does $\ln P(\mathbf{x})$ change in a typical step? By how much should $\ln P(\mathbf{x})$ vary when $\mathbf{x}$ is drawn from $P(\mathbf{x})$?

(c) What happens if, rather than using a Metropolis method that tries to change all components at once, one instead uses a concatenation of Metropolis updates changing one component at a time?

[2] `http://www.inference.phy.cam.ac.uk/mackay/itila/`

▷ Exercise 29.19.[2] When discussing the time taken by the Metropolis algorithm to generate independent samples we considered a distribution with longest spatial length scale L being explored using a proposal distribution with step size ϵ. Another dimension that a MCMC method must explore is the range of possible values of the log probability $\ln P^*(\mathbf{x})$. Assuming that the state $\mathbf{x}$ contains a number of independent random variables proportional to N, when samples are drawn from $P(\mathbf{x})$, the 'asymptotic equipartition' principle tell us that the value of $-\ln P(\mathbf{x})$ is likely to be close to the entropy of $\mathbf{x}$, varying either side with a standard deviation that scales as $\sqrt{N}$. Consider a Metropolis method with a symmetrical proposal density, that is, one that satisfies $Q(\mathbf{x};\mathbf{x}') = Q(\mathbf{x}';\mathbf{x})$. Assuming that accepted jumps either increase $\ln P^*(\mathbf{x})$ by some amount or decrease it by a *small* amount, e.g. $\ln e = 1$ (is this a reasonable assumption?), discuss how long it must take to generate roughly independent samples from $P(\mathbf{x})$. Discuss whether Gibbs sampling has similar properties.

Exercise 29.20.[3] Markov chain Monte Carlo methods do not compute partition functions Z, yet they allow ratios of quantities like Z to be estimated. For example, consider a random-walk Metropolis algorithm in a state space where the energy is zero in a connected accessible region, and infinitely large everywhere else; and imagine that the accessible space can be chopped into two regions connected by one or more corridor states. The fraction of times spent in each region at equilibrium is proportional to the volume of the region. How does the Monte Carlo method manage to do this without measuring the volumes?

Exercise 29.21.[5] Philosophy.

One curious defect of these Monte Carlo methods – which are widely used by Bayesian statisticians – is that they are all non-Bayesian (O'Hagan, 1987). They involve computer experiments from which *estimators* of quantities of interest are derived. These estimators depend on the proposal distributions that were used to generate the samples and on the random numbers that happened to come out of our random number generator. 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 that 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.

Can you make a Bayesian Monte Carlo method? (See Rasmussen and Ghahramani (2003) for a practical attempt.)

▶ 29.12 Solutions

Solution to exercise 29.1 (p.364). We wish to show that

$$\hat{\Phi} \equiv \frac{\sum_r w_r \phi(x^{(r)})}{\sum_r w_r} \tag{29.52}$$

converges to the expectation of Φ under P. We consider the numerator and the denominator separately. First, the denominator. Consider a single importance weight.

$$w_r \equiv \frac{P^*(x^{(r)})}{Q^*(x^{(r)})}. \tag{29.53}$$