Chapter 1

Sampling Methods

There are many probabilistic models of practical interest for which exact inference is intractable. One important class of inference algorithms is based on deterministic approximation schemes and includes methods such as variational methods. Here we consider an alternative very general and widely used framework for approximate inference based on numerical sampling, also known as the Monte Carlo technique.

Although for some applications of graphical models the posterior distribution over unobserved variables will be of direct interest in itself, for most situations the posterior distribution is required primarily for the purpose of evaluating expectations, for example in order to make predictions.

The fundamental problem which we therefore wish to address in this chapter involves finding the expectation of some function $f(x)$ with respect to a probability distribution $p(x)$. Here, the components of x might comprise discrete or continuous variables, or some combination of the two. Thus in the case of continuous variables we wish to evaluate the expectation

$$\langle f \rangle = \int f(x)p(x)\,dx. \tag{1.1}$$

This is illustrated schematically for a 1-dimensional distribution in Figure **??**. We shall suppose

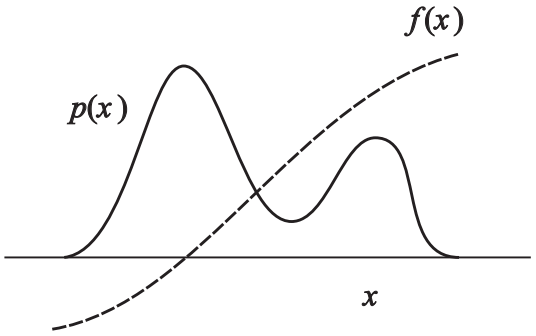

Figure 1.1: Schematic illustration of a function $f(x)$ whose expectation is to be evaluated with respect to a distribution $p(x)$.

that such expectations are too complex to be evaluated exactly using analytical techniques. In some cases an expectation can be evaluated by first finding an analytic approximation to the posterior distribution. One approach to finding an approximate posterior distribution is to use variational

methods, as discussed at length in Chapter ??. The Laplace approximation framework, described in Chapter ??, has also been applied to this problem.

The general idea behind sampling methods is to obtain a set of samples $x^{(l)}$ (where $m = 1, \ldots, M$) drawn from the distribution $p(x)$. This allows the expectation (**??**) to be approximated by a finite sum

$$\widehat{f} = \frac{1}{M} \sum_{m=1}^{M} f(x^{(m)}). \tag{1.2}$$

The accuracy of such an approximation will depend on a variety of factors, some of which will be discussed in greater detail later in this chapter. Here we note that as long as the samples $x^{(m)}$ are drawn from the distribution $p(x)$ then $\langle \widehat{f} \rangle = \langle f \rangle$ and so the estimator has the correct mean. The variance of the estimator is easily seen to be σ^2/M, where

$$\sigma^2 = \langle (f - \langle f \rangle)^2 \rangle \tag{1.3}$$

is the variance of the function $f(x)$ under the distribution $p(x)$. It is worth emphasizing that the accuracy of the estimator therefore does not depend on the dimensionality of x, and that, potentially, high accuracy may be achievable with a relatively small number of samples $x^{(m)}$. Furthermore, the variance of the estimator will decrease with increasing number M of samples.

One potential difficulty, however, is that the samples $\{x^{(m)}\}$ might not be independent, and so the effective sample size might be much smaller than the apparent sample size. Also, referring back to Figure **??**, we note that if $f(x)$ is small in regions where $p(x)$ is large, and vice versa, then the expectation may be dominated by regions of small probability, implying that relatively large sample sizes will be required to achieve sufficient accuracy.

While sampling methods have wide applicability, we shall of course be primarily interested in the case in which the distribution $p(x)$ is specified in terms of a graphical model. In the case of a directed graph with no observed variables it is straightforward to sample from the joint distribution (assuming that it is possible to sample from the conditional distributions at each node) using the following *ancestral sampling* approach. The joint distribution is specified by

$$p(x) = \prod_{i=1}^{d} p(x_i | x_{\pi(i)}) \tag{1.4}$$

where $x_{\pi(i)}$ denotes the set of variables associated with the parents of x_i. To obtain a sample from the joint distribution we make one pass through the set of variables in the order $x_1, \ldots, x_d$ sampling from the conditional distributions $p(x_i | x_{\pi(i)})$. This is always possible since at each step all of the parent values will have been instantiated. After one pass through the graph we will have obtained a sample from the joint distribution.

Now consider the case of a directed graph in which some of the nodes are instantiated with observed values. We can in principle extend the above procedure, at least in the case of nodes representing discrete variables, to give the following *logic sampling* approach, which can be seen as a special case of 'importance sampling' discussed in Section **??**. At each step, when a sampled value is obtained for a variable x_i whose value is observed, the sampled value is compared to the observed value and if they agree then again the sample value is retained and the algorithm proceeds to next variable in turn. However, if the sampled value and the observed value disagree, then the whole sample so far is discarded and the algorithm starts again with the first node in the graph. This algorithm samples correctly from the posterior distribution since it corresponds

simply to drawing samples from the joint distribution of hidden variables and data variables and then discarding those samples which disagree with the observed data (with the slight saving of not continuing with the sampling from the joint distribution as soon as one contradictory value is observed). However, the overall probability of accepting a sample from the posterior decreases rapidly as the number of observed variables, and the number of states which those variables can take, increases.

In the case of probability distributions defined by an undirected graph there is no one-pass sampling strategy which will sample even from the prior distribution with no observed variables. Instead, computationally more expensive techniques must be employed, such as Gibbs sampling which is discussed in Section **??**.

As well as sampling from conditional distributions we may also require samples from a marginal distribution. If we already have a strategy for sampling from a joint distribution $p(x, y)$ then it is straightforward to obtain samples from the marginal distribution $p(x)$ simply by ignoring the values for y in each sample.

1.0.1 Sampling and the EM Algorithm

In addition to providing a mechanism for direct implementation of the Bayesian framework, Monte Carlo methods can also play a role in the frequentist paradigm, for example to find maximum likelihood solutions. In particular, sampling methods can be used to approximate the E-step of the EM algorithm for models in which the E-step cannot be performed analytically. Consider a model with hidden variables x_H, visible (observed) variables x_V and parameters θ. The function which is optimized with respect to θ in the M-step is the expected complete-data log likelihood, given by

$$Q(\theta, \theta_{\text{old}}) = \int p(x_H|x_V, \theta_{\text{old}}) \ln p(x_H, x_V|\theta)\, dx_H. \tag{1.5}$$

We can use sampling methods to approximate this integral by a finite sum over samples $\{x_H^{(l)}\}$ drawn from the current estimate for the posterior distribution $p(x_H|x_V, \theta_{\text{old}})$, so that

$$Q(\theta, \theta_{\text{old}}) \simeq \frac{1}{M} \sum_{m=1}^{M} \ln p(x_H^{(m)}, x_V|\theta). \tag{1.6}$$

The Q function is then optimized in the usual way in the M-step. This procedure is called the *Monte Carlo EM algorithm*.

It is straightforward to extend this to the problem of finding the mode of the posterior distribution over θ (the MAP estimate) when a prior distribution $p(\theta)$ has been defined, simply by adding $\ln p(\theta)$ to the function $Q(\theta, \theta_{\text{old}})$ before performing the M-step.

A particular instance of the Monte Carlo EM algorithm, called *stochastic EM*, arises if we consider a finite mixture model, and draw just one sample at each E-step. Suppose the L-dimensional binary latent variable x_z characterizes which of the L components of the mixture is responsible for generating each data point. In particular there is one vector-valued element of x_z for each data point in the data set, and each such vector has all of its L elements equal to zero except for the particular element representing the corresponding component of the mixture. In the E-step a sample of x_z is taken from the posterior distribution $p(x_z|x_V, \theta_{\text{old}})$ where x_V is the data set. This effectively makes a hard assignment of each data point to one of the components in the mixture. In the M-step, this sampled approximation to the posterior distribution is used to update the model parameters in the usual way.

Now suppose we move from a maximum likelihood approach to a full Bayesian treatment in

which we wish to sample from the posterior distribution over θ. In principle we would like to draw samples from the joint posterior $p(\theta, x_H|x_V)$, but we shall suppose that this is computationally difficult. Suppose further that it is relatively straightforward to sample from the complete-data parameter posterior $p(\theta|x_H, x_V)$. This inspires the *data augmentation* algorithm, which alternates between two steps known as the I-step (imputation step, analogous to an E-step) and the P-step (posterior step, analogous to an M-step).

I-step. We wish to sample from $p(x_H|x_V)$ but we cannot do this directly. We therefore note the relation

$$p(x_H|x_V) = \int p(x_H|\theta, x_V)p(\theta|x_V)\, d\theta \tag{1.7}$$

and hence for $m = 1, \ldots, M$ we first draw a sample $\theta^{(m)}$ from the current estimate for $p(\theta|x_V)$, and then use this to draw a sample $x_H^{(m)}$ from $p(x_H|\theta^{(m)}, x_V)$.

P-step. Given the relation

$$p(\theta|x_V) = \int p(\theta|x_H, x_V)p(x_H|x_V)\, dx_H \tag{1.8}$$

we use the samples $\{x_H^{(m)}\}$ obtained from the I-step to compute a revised estimate of the posterior distribution over θ given by

$$p(\theta|x_V) \simeq \frac{1}{M}\sum_{m=1}^{M} p(\theta|x_H^{(m)}, x_V). \tag{1.9}$$

By assumption, it will be feasible to sample from this approximation in the I-step.

Note that we are making a (somewhat artificial) distinction between parameters θ and hidden variables x_H. From now on we blur this distinction and focus simply on the problem of drawing samples from a given joint posterior distribution.

1.1 Basic Sampling Algorithms

In this section we consider the problem of sampling from some standard distributions defined over a single, continuous variable $x \in \Re$. Since the samples will be generated by a computer algorithm they will in fact be *pseudo-random* numbers, that is, they will be deterministically calculated, but must nevertheless pass appropriate tests for randomness. We begin by looking at the problem of generating pseudo-random numbers with a uniform distribution over $(0, 1)$, as this forms the basis for generating numbers having other, non-uniform, distributions.

1.1.1 Standard Distributions

Pseudo-random number generators are typically based on successive applications of a transformation function $D(\cdot)$, so that a sequence $(x^{(1)}, \ldots, x^{(N)})$ is obtained using $x^{(n)} = D(x^{(n-1)})$. The particular sequence of numbers obtained is determined by the initial value $x^{(1)}$, known as the *seed*.

It might appear that the use of one of the standard chaotic functions for $D(\cdot)$ might be appropriate. A chaotic function would have the property that the resulting sequence is highly sensitive

to the initial value $x^{(1)}$, so that changes in subsequent values due to a small change in $x^{(1)}$ would grow exponentially as the sequence progresses. However, it turns out that a chaotic $D(\cdot)$ need not give rise to a practically acceptable random number generator, even when the limiting distribution is correct. For example, the finite precision used to represent continuous variables in a digital computer can result in some functions leading to sequences which converge to a fixed value.

Instead we seek choices for $D(\cdot)$ which take account of the finite representations of digital computers. It is often convenient to consider integer representations for random numbers, so that the output of the random number generator is an integer x from the set $\{0, \ldots, N\}$, where N is the largest integer which can be stored by the computer. Corresponding continuous numbers from $[0, 1)$ can subsequently be obtained using $x/(N + 1)$ which is interpreted as a floating point operation. First we define the *period* of a random number generator to be the smallest integer T such that $x^{(n+T)} = x^{(n)}$ for all n, in other words such that $D(\cdot)^T$ is the identity operator. Clearly the largest value which the period of a simple generator of the form $x^{(n)} = D(x^{(n-1)})$ can take is $M + 1$, but carelessness over the design of the random number generator can easily lead to significantly smaller values. Longer periods can be obtained using extensions of the transformation approach, such as using functions of $x^{(n)}$ and $x^{(n-1)}$.

Many common random number generators are based on the *linear congruential* method which defines

$$D(x) = (ax + b) \mod (M + 1). \tag{1.10}$$

Considerable care must be exercised in the choice of the constants a and b in order to ensure a generator with acceptable properties. One obviously desirable property is that the sequence have its maximum period of $M + 1$.

We next consider how to generate random numbers from non-uniform distributions, assuming that we already have available a source of uniformly distributed random numbers. Suppose that x is uniformly distributed over the interval $(0, 1)$, and suppose that we transform the values of x using some function $f(\cdot)$ so that $y = f(x)$. The distribution of y will be governed by

$$p(y) = p(x) \left| \frac{dx}{dy} \right| \tag{1.11}$$

where, in this case, $p(x) = 1$. Our goal is to choose the function $f(x)$ such that the resulting values of y have some specific desired distribution $p(y)$. Integrating **(??)** we obtain $x = h(y) \equiv \int_{-\infty}^{y} p(y')\, dy'$ which is the indefinite integral of $p(y)$. Thus, $y = h^{-1}(x)$, and so we have to transform the uniformly distributed random numbers using a function which is the inverse of the indefinite integral of the desired distribution. This is illustrated in Figure **??**.

Consider for example the exponential distribution

$$p(y) = \lambda \exp(-\lambda y). \tag{1.12}$$

In this case $h(y) = 1 - \exp(-\lambda y)$ and so if we transform our uniformly distributed variable x using $y = -\lambda^{-1} \ln(1 - x)$ then y will have an exponential distribution.

Another example of a distribution to which the transformation method can be applied is given by the Cauchy distribution

$$p(y) = \frac{1}{\pi} \frac{1}{1 + y^2}. \tag{1.13}$$

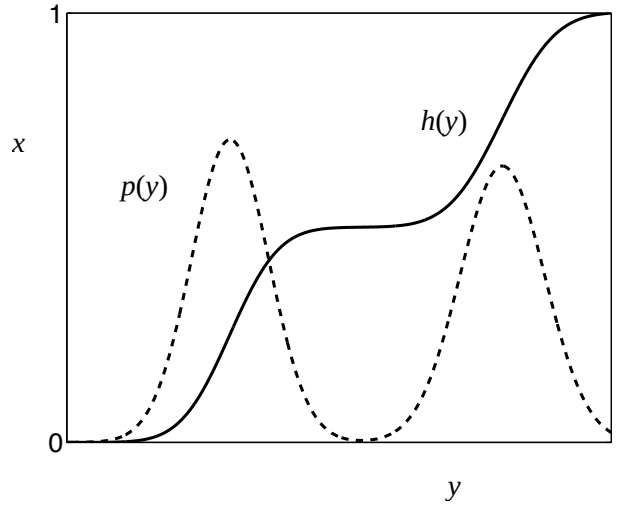

Figure 1.2: Geometrical interpretation of the transformation method for generating non-uniformly distributed random numbers. $h(y)$ is the indefinite integral of the desired distribution $p(y)$. If a uniformly distributed random variable x is transformed using $y = h^{-1}(x)$ then y will be distributed according to $p(y)$.

In this case the inverse of the indefinite integral is the 'tan' function.

The generalization to multiple variables is straightforward and involves the Jacobian of the change of variables, so that

$$p(y_1, \ldots, y_d) = p(x_1, \ldots, x_d) \left| \frac{\partial(y_1, \ldots, y_d)}{\partial(x_1, \ldots, Xx_d)} \right|. \tag{1.14}$$

As a final example of the transformation method we consider the Box-Muller method for generating samples from a Gaussian distribution. Suppose we generate pairs of uniformly distributed random numbers $x_1, x_2 \in (-1, 1)$ (which we can do by transforming a variable distributed uniformly over $(0, 1)$ using $x \to 2x - 1$). Next we discard each pair unless it satisfies $x_1^2 + x_2^2 \leq 1$. This leads to a uniform distribution of points inside the unit circle with $p(x_1, x_2) = 1/\pi$, as illustrated in Figure **??**. Then, for each pair x_1, x_2 we evaluate the quantities

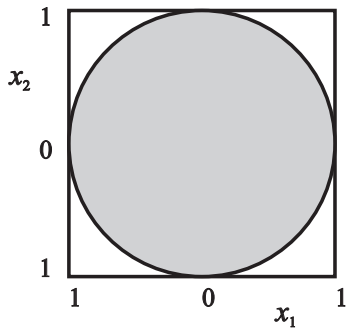

Figure 1.3: The Box-Muller method for generating normally distributed random numbers involves generating samples from a uniform distribution inside the unit circle.

$$y_1 = x_1 \left(\frac{-2 \ln x_1}{r^2} \right)^{1/2} \tag{1.15}$$

$$y_2 = x_2 \left(\frac{-2 \ln x_2}{r^2} \right)^{1/2} \tag{1.16}$$

where $r^2 = x_1^2 + x_2^2$. Then the joint distribution of y_1 and y_2 is given by

$$p(y_1, y_2) = p(x_1, X x_2) \left| \frac{\partial(y_1, y_2)}{\partial(x_1, x_2)} \right| \tag{1.17}$$

$$= \left[\frac{1}{\sqrt{2\pi}} \exp(-y_1^2/2) \right] \left[\frac{1}{\sqrt{2\pi}} \exp(-y_2^2/2) \right] \tag{1.18}$$

and so y_1 and y_2 are independent and each has a normal distribution with zero mean and unit variance.

Clearly the transformation $y \to \sigma y + \mu$ can be used to generate normally distributed random numbers with mean μ and variance σ^2.

To generate vector-valued variables having a multi-variate normal distribution with mean μ and covariance Σ we can employ the eigenvector / eigenvalue decomposition of the covariance matrix

$$\Sigma u_i = \lambda_i u_i. \tag{1.19}$$

It is then easily verified that, if x_i are univariate and normally distributed, then the variable

$$y = \mu + \sum_i \lambda_i^{1/2} x_i u_i \tag{1.20}$$

has the required multi-variate distribution. This is illustrated geometrically in Figure **??**. In practice it is computationally more efficient, and also more robust, to use a Cholesky decomposition of the form $\Sigma = LL^{\mathrm{T}}$. Then, if x is a vector valued random variable whose components are independent and normally distributed with zero mean and unit variance, then $y = \mu + Lx$ will have mean μ and covariance Σ.

Obviously the transformation technique depends for its success on the ability to calculate and then invert the indefinite integral of the required distribution. Such operations will only be feasible for a limited number of very simple distributions, and so we must turn to alternative approaches in search of a more general strategy. Here we consider two techniques called *rejection sampling* and *importance sampling*. Although mainly limited to univariate distributions and thus not directly applicable to complex problems in many dimensions, they do form important components in more general strategies.

1.1.2 Rejection Sampling

The rejection sampling framework allows us to sample from relatively complex distributions, subject to certain constraints. We begin by considering univariate distributions and discuss the extension to multiple dimensions subsequently.

Suppose we wish to samples from a distribution $p(x)$ which is not one of the simple, standard

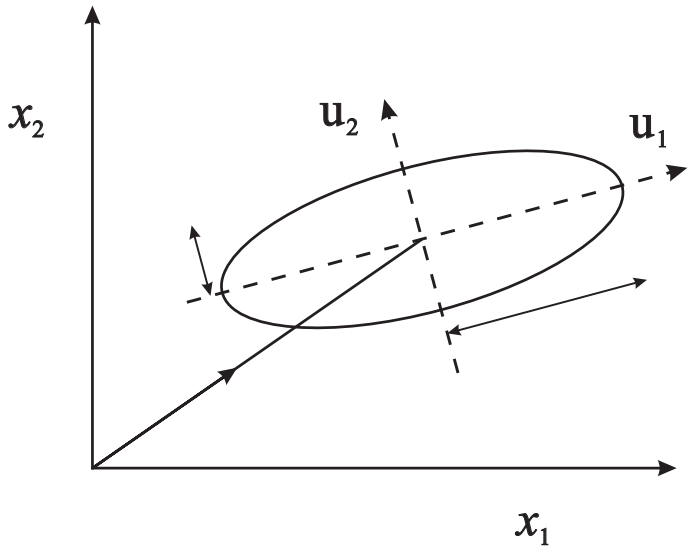

Figure 1.4: Geometrical view of a procedure for generating samples from a multi-variate normal distribution, given a source of univariate normally-distributed random numbers. The ellipse represents the one standard deviation contour of a Gaussian distribution with mean μ whose covariance matrix has eigenvectors u_i with corresponding eigenvalues λ_i. A sample from the multivariate Gaussian distribution can be obtained using a linear combination of the eigenvectors with coefficients given by suitably scaled univariate Gaussian distributed random variables.

distributions considered so far, and that sampling directly from $p(x)$ is difficult. Furthermore suppose, as is often the case, that we are easily able to evaluate $p(x)$ for any given value of x, up to some normalizing constant Z, so that

$$p(x) = \frac{1}{Z}\widetilde{p}(x) \tag{1.21}$$

where $\widetilde{p}(x)$ can readily be evaluated, but Z is unknown.

In order to apply rejection sampling we need some simpler distribution $q(x)$, sometimes called a proposal distribution, from which we can readily draw samples. We next introduce a constant k whose value is chosen such that $kq(x) \geq \widetilde{p}(x)$ for all values of x. The function $kq(x)$ is called the comparison function, and is illustrated in Figure **??**. Each step of the rejection sampler involves generating two random numbers. First, we generate a number x_0 from the distribution $q(x)$. Next, we generate a number u_0 from the uniform distribution over $[0, kq(x_0)]$. This pair of random numbers has uniform distribution under the curve of the function $kq(x)$. Finally, if $u_0 > \widetilde{p}(x_0)$ then the sample is rejected, otherwise u_0 is retained. Thus the pair is rejected if it lies in the grey shaded region in Figure **??**. The remaining pairs then have uniform distribution under the curve of $\widetilde{p}(X)$, and hence the corresponding X values are distributed according to $p(x)$, as desired.

We can see more formally that the rejection sampling procedure samples from the correct distribution as follows. The original values of x are generated with distribution $q(x)$ and these samples are then accepted with probability $\widetilde{p}(x)/kq(x)$ and so the resulting distribution of x is given by

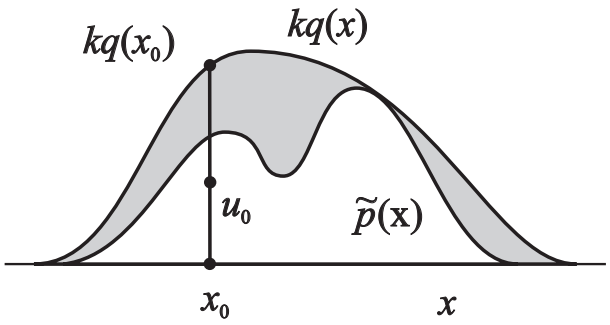

Figure 1.5: In the rejection sampling method, samples are drawn from a simple distribution $q(x)$ and rejected if they fall in the grey area shown. The resulting samples are distributed according to $p(x)$, which is the normalized version of $\widetilde{p}(x)$.

normalization as

$$\frac{[\widetilde{p}(x)/kq(x)]q(x)}{\int [\widetilde{p}(x)/kq(x)]q(x)\,dx} = \frac{\widetilde{p}(x)}{\int \widetilde{p}(x)\,dx} = p(x). \tag{1.22}$$

The probability that a sample will be accepted is given by

$$\begin{aligned} p(\text{accept}) &= \int [\widetilde{p}(x)/kx(x)]q(x)\,dx \\ &= \frac{1}{k}\int \widetilde{p}(x)\,dx. \end{aligned} \tag{1.23}$$

Thus the fraction of points which are rejected by this method depends on the ratio of the area under the unnormalized distribution $\widetilde{p}(x)$ to the area under the curve $kq(x)$. We therefore see that the constant k should be as small as possible subject to the limitation that $kq(x)$ must be nowhere less than $\widetilde{p}(x)$.

As an illustration of the use of rejection sampling, consider the task of sampling from the Gamma distribution

$$\text{Gam}(x|a) = \frac{x^{a-1}\exp(-ax)}{\Gamma(a)}. \tag{1.24}$$

where $a > 0$. This has, for $a > 1$, a bell-shaped form, as shown in Figure **??**. A suitable proposal distribution is therefore the Cauchy (**??**) since this too is bell-shaped and since we can use the transformation method, discussed earlier, to sample from it. We need to generalize the Cauchy slightly to ensure that it nowhere has a smaller value than the Gamma distribution. This can be achieved by transforming a uniform random variable z using $x = b\tan z + c$, which gives random numbers distributed according to

$$q(x) = \frac{k}{1 + (x-c)^2/b^2}. \tag{1.25}$$

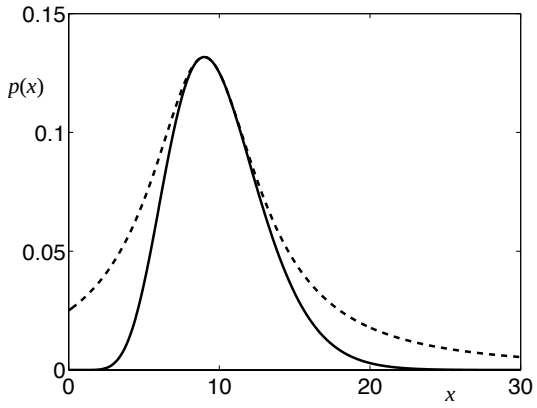

Figure 1.6: Plot showing the Gamma distribution given by (**??**) as the solid curve, with a scaled Cauchy proposal distribution given by the dashed curve. Samples from the Gamma distribution can be obtained by sampling from the Cauchy and then applying the rejection sampling criterion.

The minimum reject rate is obtained by setting $c = a - 1$, $b^2 = 2a - 1$ and choosing the constant k to be as small as possible while still satisfying the requirement $kq(x) \geq \widetilde{p}(x)$. The resulting comparison function is shown, together with the Gamma distribution, in Figure **??**.

1.1.3 Adaptive Rejection Sampling

In many instances where we might wish to apply rejection sampling it proves difficult to determine a suitable analytic form for the envelope distribution $q(x)$. An alternative approach is to construct the envelope function on the fly based on measured values of the distribution $p(x)$. We consider only the univariate case since it is only here that the algorithm is of significant practical value. Construction of an envelope function is particularly straightforward for cases in which $p(x)$ is log concave, in other words when $\ln p(x)$ has derivatives which are non-increasing functions of x. The construction of a suitable envelope function is illustrated graphically in Figure **??**.

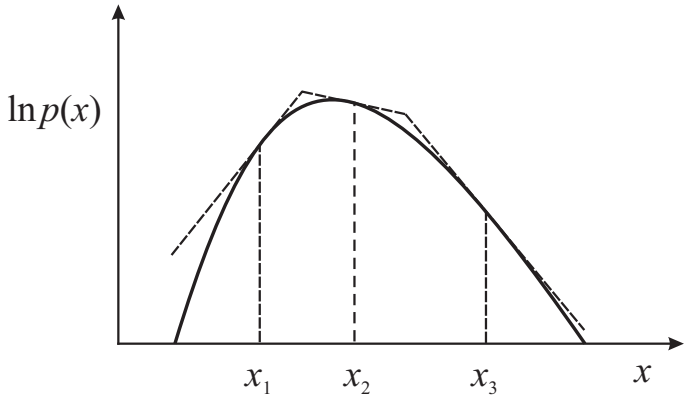

Figure 1.7: In the case of distributions which are log concave, an envelope function for use in rejection sampling can be constructed using the tangent lines computed at a set of grid points. If a sample point is rejected it is added to the set of grid points and used to refine the envelope distribution.

The function $\ln p(x)$ and its gradient are evaluated at some initial set of grid points, and the intersections of resulting tangent lines are used to construct the envelope function. Next a sample value is drawn from the envelope distribution. This is straightforward (Exercise **??**) since the log of the envelope distribution is a succession of linear functions, and hence the envelope distribution itself comprises a piecewise exponential distribution of the form

$$q(x) = k_i \lambda_i \exp\{-\lambda_i(x - x_{i-1})\} \qquad x \in (x_{i-1}, x_i]. \tag{1.26}$$

Once a sample has been drawn the usual rejection criterion can be applied. If the sample is accepted then it will be a draw from the desired distribution. If, however, the sample is rejected, then it is incorporated into the set of grid points, a new tangent line is computed and the envelope function is thereby refined. As the number of grid points increases, so the envelope function becomes a better approximation of the desired distribution $p(x)$ and the probability of rejection decreases.

A variant of the algorithm exists which avoids the evaluation of derivatives. The adaptive rejection sampling framework can also be extended to distributions which are not log concave, simply by following each rejection sampling step with a Metropolis-Hastings step, giving rise to *adaptive rejection Metropolis* sampling.

Clearly for rejection sampling to be of practical value we require that the comparison function be close to the required distribution so that the rate of rejection is kept to a minimum. Now let us examine what happens when we try to use rejection sampling in spaces of high dimensionality. Consider, for the sake of illustration, a somewhat artificial problem in which we wish to sample from a zero-mean multi-variate normal distribution with covariance $\sigma_P^2 I$, where I is the unit matrix, by rejection sampling from a proposal distribution which is itself a zero-mean normal distribution having covariance $\sigma_Q^2 I$. Obviously we must have $\sigma_Q^2 > \sigma_P^2$ in order that there exists a k such that $kq(x) \geq p(x)$. In d-dimensions the optimum value of k is given by $k = (\sigma_q/\sigma_p)^d$, as illustrated for $d = 1$ in Figure **??**. The acceptance rate will be the ratio of volumes under the

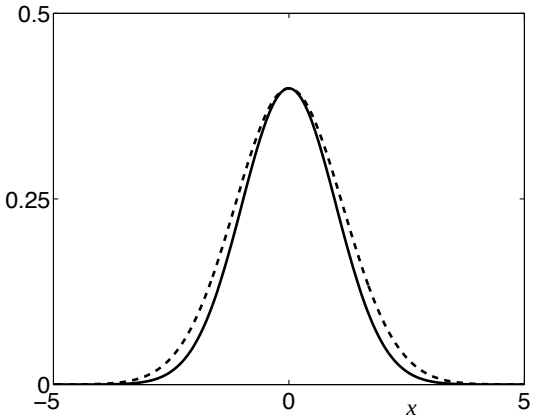

Figure 1.8: Illustrative example of rejection sampling involving sampling from a normal distribution $p(x)$ shown by the solid curve, by using rejection sampling from a proposal distribution $q(x)$, shown by the dashed curve, which is also normal.

$p(x)$ and $kq(x)$ which, since both distributions are normalized, is just $1/k$. Thus the acceptance rate diminishes exponentially with dimensionality. Even if σ_q exceeds σ_p by just one percent, for $d = 1000$ the acceptance ratio will be approximately $1/20,000$. In this illustrative example the comparison function is close to the required distribution. For more practical examples, where the desired distribution may be multi-modal and sharply peaked, it will be extremely difficult to find

a good proposal distribution and comparison function. Furthermore, the exponential decrease of acceptance rate with dimensionality is a generic feature of rejection sampling. Although rejection can be a useful technique in one or two dimensions it is unsuited to problems of high dimensionality. It can, however, play a role as a sub-routine in more sophisticated algorithms for sampling in high dimensional spaces.

1.1.4 Importance Sampling

One of the principal reasons for wishing to sample from complicated probability distributions is to be able to evaluate expectations of the form (**??**). Importance sampling provides a framework for approximating expectations, but does not itself provide a mechanism for drawing samples from distribution $p(x)$.

The finite sum approximation to the expectation, given by (**??**), depends on being able to draw samples from the distribution $p(x)$. Suppose, however, that it is impractical to sample directly from $p(x)$ by that we can evaluate $p(x)$ easily for any given value of x. As we discussed at the start of this chapter, one strategy for evaluating expectations would be to discretize the x-space into a uniform grid and to evaluate the integrand as a sum of the form

$$\langle f \rangle \simeq \sum_{m=1}^{M} p(x^{(m)}) f(x^{(m)}). \tag{1.27}$$

An obvious problem with this approach is that the number of terms in the summation grows exponentially with the dimensionality of x. Furthermore, as we have already noted, the kinds of probability distributions of interest will often have much of their mass confined to relatively small regions of x space and so uniform sampling will be very inefficient since in high dimensional problems only a very small proportion of the samples will make a significant contribution to the sum. We would really like to choose the sample points to fall in regions where $p(x)$ is large (or more precisely where the product $p(x)f(x)$ is large).

As in the case of rejection sampling, importance sampling is based on the use of a distribution $q(x)$ from which it is easy to draw samples. This is illustrated in Figure **??** We can then express the

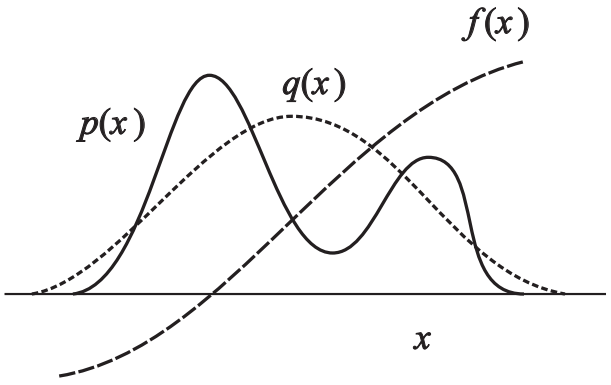

Figure 1.9: Importance sampling addresses the problem of evaluating the expectation of a function $f(x)$ with respect to a distribution $p(x)$ from which it is difficult to draw samples directly. Instead, samples $\{x^{(m)}\}$ are drawn from a simpler distribution $q(x)$ and the corresponding terms in the summation are weighted by the ratios $p(x^{(m)})/q(x^{(m)})$.

expectation in the form of a finite sum over samples $\{x^{(m)}\}$ drawn from $q(x)$

$$\begin{aligned}\langle f \rangle &= \int f(x)p(x)\,dx \\ &= \int f(x)\frac{p(x)}{q(x)}q(x)\,dx \\ &\simeq \frac{1}{M}\sum_{m=1}^{M}\frac{p(x^{(m)})}{q(x^{(m)})}f(x^{(m)}).\end{aligned} \tag{1.28}$$

The quantities $r_m = p(x^{(m)})/q(x^{(m)})$ are known as importance weights, and they correct the bias introduced by sampling from the wrong distribution. Note that, unlike rejection sampling, all of the samples generated are retained.

It will often be the case that the distribution $p(x)$ can only be evaluated up to a normalization constant, so that $p(x) = \widetilde{p}(x)/Z_p$ where $\widetilde{p}(x)$ can be evaluated easily, whereas Z_p is unknown. Similarly, we may wish to use an importance sampling distribution $q(x) = \widetilde{q}(x)/Z_q$ which has the same property. We then have

$$\begin{aligned}\langle f \rangle &= \int f(x)p(x)\,dx \\ &= \frac{Z_q}{Z_p}\int f(x)\frac{\widetilde{p}(x)}{\widetilde{q}(x)}q(x)\,dx \\ &\simeq \frac{Z_q}{Z_p}\frac{1}{M}\sum_{m=1}^{M} r_m f(x^{(m)}).\end{aligned} \tag{1.29}$$

where $r_m = \widetilde{p}(x^{(m)})/\widetilde{q}(x^{(m)})$. We can use the same sample set to evaluate the ratio Z_p/Z_q with the result

$$\begin{aligned}\frac{Z_p}{Z_q} &= \frac{1}{Z_q}\int \widetilde{p}(x)\,dx \\ &= \int p(x)\frac{q(x)}{\widetilde{q}(x)}\,dx \\ &\simeq \frac{1}{L}\sum_{m=1}^{L} r_m\end{aligned} \tag{1.30}$$

and hence

$$\langle f \rangle \simeq \frac{\sum_m r_m f(x^{(m)})}{\sum_m r_m}. \tag{1.31}$$

As with rejection sampling, the success of the importance sampling approach depends crucially

on how well the sampling distribution $q(x)$ matches the desired distribution $p(x)$. If, as is often the case, $p(x)f(x)$ is strongly varying and has a significant proportion of its mass concentrated over relatively small regions of x space, then the set of importance weights $\{r_m\}$ may be dominated by a few weights having large values, with the remaining weights being relatively insignificant. Thus the effective sample size can be much smaller than the apparent sample size M. The problem is even more severe if none of the samples fall in the regions where $p(x)f(x)$ is large. In that case the apparent variances of r_m and $r_m f(x_m)$ may be small even though the estimate of the expectation may be severely wrong. Hence a major drawback of the importance sampling method is the potential to produce results which are arbitrarily in error and with no diagnostic indication. This also highlights a key requirement for the sampling distribution $q(x)$, namely that it should not be small or zero in regions where $p(x)$ may be significant. In practice for continuous densities this implies the requirement to use distributions $q(x)$ which are heavy tailed.

We can apply the importance sampling technique to distributions defined by graphical models in various ways. For discrete variables a very simple approach is called *uniform sampling*. The joint distribution for a directed graph is defined by (**??**). Each sample from the joint distribution is obtained by first setting those variables x_i which are in the evidence set equal to their observed values. Each of the remaining variables is then sampled independently from a uniform distribution over the space of possible instantiations. To determine the corresponding weight associated with a sample $x^{(m)}$ we note that the sampling distribution $\widetilde{q}(x)$ is uniform over the possible choices for x, and that $\widetilde{p}(x|e) = \widetilde{p}(x)$, where e denotes the evidence associated with instantiated variables, and the equality follows from the fact that every sample x which is generated is necessarily consistent with the evidence. Thus the weights r_m are simply proportional to $p(x)$. Note that the variables can be sampled in any order. This approach can again yield poor results if the posterior distribution is far from uniform.

An improvement on this approach is called *likelihood weighted sampling* and is based on ancestral sampling of the variables. For each variable x_i in turn, if that variable is in the evidence set then it is just set to its instantiated value. If it is not in the evidence set then it is sampled from the conditional distribution $p(x_i|x_{\pi(i)})$ in which the conditioning variables are set to their currently sampled values. The weighting associated with the resulting sample x is given by

$$r(x) = \prod_{i \notin e} \frac{p(x_i|x_{\pi(i)})}{p(x_i|x_{\pi(i)})} \prod_{i \in e} \frac{p(x_i|x_{\pi(i)})}{1} = \prod_{i \in e} p(x_i|x_{\pi(i)}). \tag{1.32}$$

This method can be further extended using *self-importance sampling* in which the importance sampling distribution is continually updated to reflect the current estimated posterior distribution. A further refinement called *Markov blanket scoring* distributes fractions of samples to the states of a node in proportion to the probability of these values conditioned on the Markov blanket of a node.

1.1.5 Sampling-Importance-Resampling

The rejection sampling method discussed in Section **??** depends in part for its success on the determination of a suitable value for the constant k. For many examples of distributions $p(x)$ and $q(x)$ it will be impractical to determine a suitable value for k in that any value which is sufficiently large to guarantee a bound on the desired distribution will lead to impractically small acceptance rates.

As in the case of rejection sampling, the *sampling-importance-resampling* (SIR) approach also makes use of a sampling distribution $q(x)$, but avoids having to determine the constant k. There are two stages to the scheme. In the first stage N samples $x^{(1)}, \ldots, x^{(N)}$ are drawn from $q(x)$. Then

in the second stage weights $w^{(1)}, \ldots, w^{(N)}$ are constructed using

$$w^{(n)} = \frac{\widetilde{p}(x^{(n)})/q(x^{(n)})}{\sum_{n=1}^{N} \widetilde{p}(x^{(n)})/q(x^{(n)})}. \tag{1.33}$$

A second set of M samples is drawn from the discrete distribution $(x^{(1)}, \ldots, x^{(N)})$ with probabilities given by the weights $(w^{(1)}, \ldots, w^{(N)})$.

The resulting M samples are only approximately distributed according to $p(x)$, but the distribution becomes correct in the limit $N \to \infty$. To see this we note that the cumulative distribution of the resampled values is given by

$$\begin{aligned} \Pr(x \leq a) &= \sum_{n:x_n \leq a} w^{(n)} \\ &= \frac{\sum_{n=1}^{N} I(x_n \leq a)\widetilde{p}(x_n)/q(x_n)}{\sum_{n=1}^{N} \widetilde{p}(x_n)/q(x_n)} \end{aligned} \tag{1.34}$$

where $I(.)$ is the indicator function (which equals 1 if it's argument is true and 0 otherwise). Taking the limit $N \to \infty$, and assuming suitable regularity of the distributions, we can replace the sums by integrals weighted according to the original sampling distribution $q(x)$

$$\begin{aligned} \Pr(x \leq a) &= \frac{\int I(x \leq a)[\widetilde{p}(x)/q(x)]q(x)\,dx}{\int [\widetilde{p}(x)/q(x)]q(x)\,dx} \\ &= \frac{\int I(x \leq a)\widetilde{p}(x)\,dx}{\int \widetilde{p}(x)\,dx} \\ &= \int I(x \leq a)p(x)\,dx \end{aligned} \tag{1.35}$$

which is the cumulative distribution function of $p(x)$. Again we see that the normalization of $p(x)$ is not required.

For a finite value of N, and a given initial sample set, the resampled values will only approximately be drawn from the desired distribution. As with rejection sampling, the approximation improves as the sampling distribution $q(x)$ gets closer to the desired distribution $p(x)$. When $q(x) = p(x)$ the initial samples $(x^{(1)}, \ldots, x^{(N)})$ have the desired distribution, and the weights $w_n = 1/N$ so that the resampled values also have the desired distribution.

If moments with respect to the distribution $p(x)$ are required, then they can be evaluated di-

rectly using the original samples together with the weights, since

$$\begin{aligned}\langle F(x)\rangle &= \int F(x)p(x)\,dx \\ &= \frac{\int F(x)[\widetilde{p}(x)/q(x)]q(x)\,dx}{\int [\widetilde{p}(x)/q(x)]q(x)\,dx} \\ &\simeq \sum_{n=1}^{N} F(x_n)w_n. \end{aligned} \tag{1.36}$$

1.1.6 Particle Filters

We can apply the re-sampling formalism of Section **??** to obtain a sequential Monte Carlo algorithm known as the particle filter. Consider the class of distributions represented by the graphical model in Figure **??**. In many applications the observed data y_t arrive sequentially and we wish

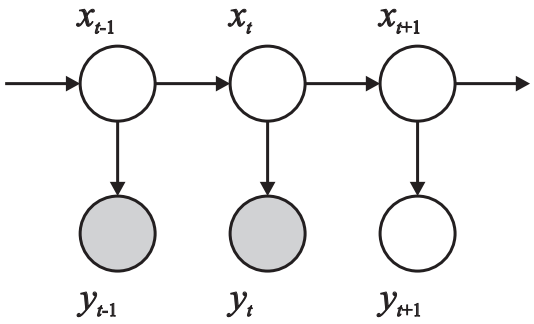

Figure 1.10: Directed graphical model in which a sequence of observations $y_1, \ldots, y_t, \ldots$ is explained in terms of a hidden Markov chain of latent variables $x_1, \ldots, x_t, \ldots$. Sampling from the posterior distribution of the hidden variables can be performed sequentially using the particle filter algorithm.

to update the posterior distribution of the hidden variables in the light of each new observation. The probabilistic model is specified in terms of the transition probability $p(x_t|x_{t-1})$ and the observation model $p(y_t|x_t)$. This class of models includes the hidden Markov model (Chapter ??) and the Kalman filter (Chapter ??) as special cases. The standard hidden Markov model uses discrete distributions while the simplest form of the Kalman filter is based on linear-Gaussian distributions, and so both models are amenable to exact solution, as discussed already.

There is, however, considerable interest in extending such models to more complex choices for the conditional probability distributions, particularly for the observation model $p(y_t|x_t)$. This can lead to highly complex posterior distributions over the hidden variables x_t which are no longer analytically tractable. We therefore consider the application of sampling methods. In particular, suppose we are given the observed values $y_{(t)} = (y_1, \ldots, y_t)$ and we wish to draw M samples from the posterior distribution $p(x_t|y_{(t)})$, in order to evaluate the expectation of some function $f(x)$ with

respect to this distribution. Using Bayes' theorem we have

$$\begin{aligned}
\langle f(x_t)\rangle &= \int f(x_t)p(x_t|y_{(t)})\,dx_t \\
&= \int f(x_t)p(x_t|y_t, y_{(t-1)})\,dx_t \\
&= \frac{\int f(x_t)p(y_t|x_t)p(x_t|y_{(t-1)})\,dx_t}{\int p(y_t|x_t)p(x_t|y_{(t-1)})\,dx_t} \\
&\simeq \sum_{m=1}^{M} w_t^{(m)} f(x_t^{(m)}) \qquad (1.37)
\end{aligned}$$

where $\{x_t^{(m)}\}$ is a set of samples drawn from $p(x_t|y_{(t-1)})$, and we have made use of the conditional independence property $p(y_t|x_t, y_{(t-1)}) = p(y_t|x_t)$ which follows from the graph in Figure **??**. The sampling weights $\{w_t^{(m)}\}$ are defined by

$$w_t^{(m)} = \frac{p(y_t|x_t^{(m)})}{\sum_{m=1}^{M} p(y_t|x_t^{(m)})}. \qquad (1.38)$$

Thus the posterior distribution $p(x_t|y_t)$ is represented by the set of samples $\{x_t^{(m)}\}$ together with the corresponding weights $\{w_t^{(m)}\}$. Note that these weights satisfy $0 \leqslant w_t^{(m)} 1$ and $\sum_m w_t^{(m)} = 1$.

Since we wish to find a sequential sampling scheme we suppose that a set of samples and weights have been obtained at time step t and that we have subsequently observed the value of y_{t+1} and we wish to find the weights and samples at time step $t+1$. We first sample from the distribution $p(x_{t+1}|y_{(t)})$. This is straightforward since, again using Bayes' theorem

$$\begin{aligned}
p(x_{t+1}|y_{(t)}) &= \int p(x_{t+1}|x_t, y_{(t)})p(x_t|y_{(t)})\,dx_t \\
&= \int p(x_{t+1}|x_t)p(x_t|y_{(t)})\,dx_t \\
&= \int p(x_{t+1}|x_t)p(x_t|y_t, y_{(t-1)})\,dx_t \\
&= \frac{\int p(x_{t+1}|x_t)p(y_t|x_t)p(x_t|y_{(t-1)})\,dx_t}{\int p(y_t|x_t)p(x_t|y_{(t-1)})\,dx_t} \\
&= \sum_m w_t^{(m)} p(x_{t+1}|x_t^{(m)}) \qquad (1.39)
\end{aligned}$$

where we have made use of the conditional independence properties $p(x_{t+1}|x_t, y_{(t)}) = p(x_{t+1}|x_t)$ and $p(y_t|x_t, y_{(t-1)}) = p(y_t|x_t)$ which follow from the application of the d-separation criterion to the graph in Figure **??**. The distribution given by (**??**) is a mixture distribution, and samples can be drawn by choosing a component m with probability given by the mixing coefficients $w^{(m)}$ and then drawing a sample from the corresponding component.

In summary, we can view each step of the particle filter algorithm as comprising two stages. At time step t we have a sample representation of the posterior distribution $p(x_t|y_{(t)})$ expressed as samples $\{x_t^{(m)}\}$ with corresponding weights $\{w_t^{(m)}\}$. This can be viewed as a mixture representation of the form (**??**). To obtain the corresponding representation for the next time step, we first draw M samples from the mixture distribution (**??**), and then for each sample we use the new observation y_{t+1} to evaluate the corresponding weights $w_{t+1}^m \propto p(y_{t+1}|x_{t+1}^{(m)})$. This is illustrated schematically in Figure **??**.

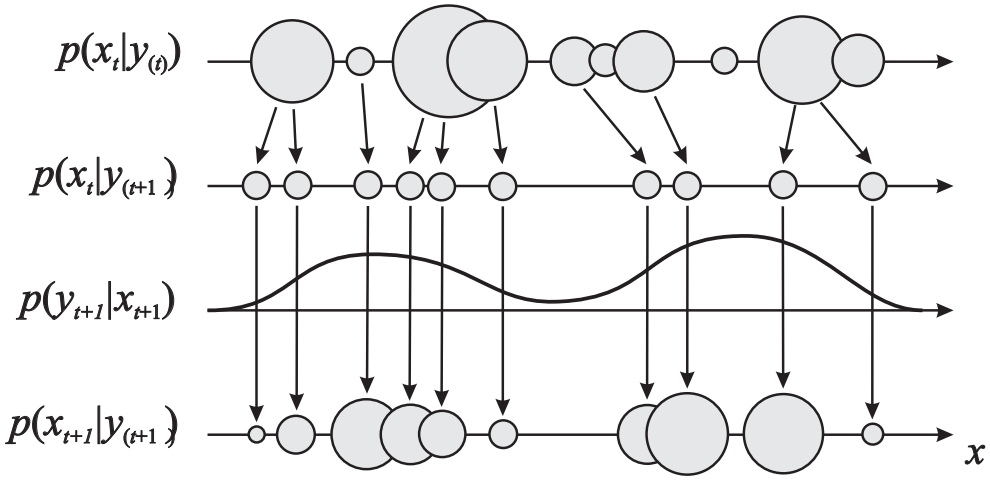

Figure 1.11: Schematic illustration of the operation of the particle filter. At time step t the posterior $p(x_t|y_t^{(m)})$ is represented as a mixture distribution, shown schematically as circles whose sizes are proportional to the weights w_t^m. A set of M samples is then drawn from this distribution, and the new weights $w_{t+1}^{(m)}$ evaluated using $p(y_{t+1}|x_{t+1}^{(m)})$.

1.2 Markov Chain Monte Carlo

In the previous section we discussed the rejection sampling and importance sampling strategies for evaluating expectations of functions, and we saw that they suffer from severe limitations particularly in spaces of high dimensionality. We turn in this section to a very general and powerful framework called Markov chain Monte Carlo (MCMC) which allows sampling from a large class of distributions, and which scales well with the dimensionality of the sample space.

As with rejection and importance sampling we again sample from a distribution $q(x)$, known as a *proposal distribution*, which differs from the desired distribution $p(x)$. This time, however, we maintain a record of the current state $x^{(t)}$, and the proposal distribution $q(x|x^{(t)})$ depends on this current state, and so the sequence of samples forms a Markov chain. Again, if we write $p(x) = \widetilde{p}(x)/Z$, we assume that $\widetilde{p}(x)$ can readily be evaluated for any given value of x, although the value of Z may be unknown. The proposal distribution itself is chosen to be sufficiently simple that it is straightforward to draw samples from it directly. At each cycle of the algorithm we generate a

candidate sample x^* from the proposal distribution and then accept the sample according to an appropriate criterion. In the basic *Metropolis* algorithm we assume that the proposal distribution is symmetric, that is $q(x_1|x_2) = q(x_2|x_1)$ for all values of x_1 and x_2. The candidate sample is then accepted with probability

$$A(x^*, x^{(t)}) = \min\left(1, \frac{\widetilde{p}(x^*)}{\widetilde{p}(x^{(t)})}\right). \tag{1.40}$$

This can be achieved by choosing a random number u with uniform distribution over the unit interval $(0, 1)$ and then accepting the sample if $A(x^*, x^{(t)}) > u$. Note that if the step from $x^{(t)}$ to x^* causes an increase in the value of $p(x)$ then the candidate point is certain to be kept.

If the candidate sample is accepted then $x^{(t+1)} = x^*$, otherwise the candidate point x^* is discarded, $x^{(t+1)}$ is set to $x^{(t)}$, and another candidate sample drawn from the distribution $Q(x; x^{(t)})$. This is in contrast to rejection sampling, where rejected samples are simple discarded. In the Metropolis algorithm when a candidate point is rejected, the previous sample is included instead in the final list of samples, leaded to multiple copies of samples[1]. As we shall see, as long as $q(x_1, x_2)$ is positive for any values of x_1 and x_2 (this is a sufficient but not necessary condition) the distribution of $x^{(t)}$ tends to $p(x)$ as $t \to \infty$. It should be emphasized, however, that the sequence $x_1, x_2, \ldots$ is not a set of independent samples from $p(x)$ since successive samples are highly correlated. If we wish to obtain independent samples then we can discard most of the sequence and just retain every N^{th} sample. For N sufficiently large the retained samples will for all practical purposes be independent.

Figure **??** shows a simple illustrative example of sampling from two-dimensional normal distribution using the Metropolis algorithm in which the proposal distribution is an isotropic Gaussian.

Further insight into the nature of Markov chain Monte Carlo algorithms can be gleaned by looking at the properties of a specific example namely a simple random walk. Consider a state space consisting of the integers, with probabilities

$$\begin{aligned} p(x_{n+1} = x_n) &= 0.5 \\ p(x_{n+1} = x_n + 1) &= 0.25 \\ p(x_{n+1} = x_n - 1) &= 0.25 \end{aligned} \tag{1.41}$$

where x_n denotes the state at step n. If the initial state is $x_1 = 0$ then by symmetry the expected state at time n will also be zero $\langle x_n \rangle = 0$, and similarly it is easily seen that $\langle x_n^2 \rangle = n/2$. Thus after n steps the random walk has only travelled a distance which on average is proportional to the square root of n. This square root dependence is typical of random walk behaviour and shows that random walks are very inefficient in exploring the state space. As we shall see, a central goal in designing Markov chain Monte Carlo methods is to avoid random walk behaviour.

1.2.1 Markov Chains

Before discussing Markov chain Monte Carlo methods in more detail, it is useful to study some general properties of Markov chains in more detail. In particular we ask under what circumstances will a Markov chain converge to the desired distribution. A first-order Markov chain is defined

[1] In a practical implementation only a single copy of each retained sample would be kept, along with an integer weighting factor recording how many times that state was retained.

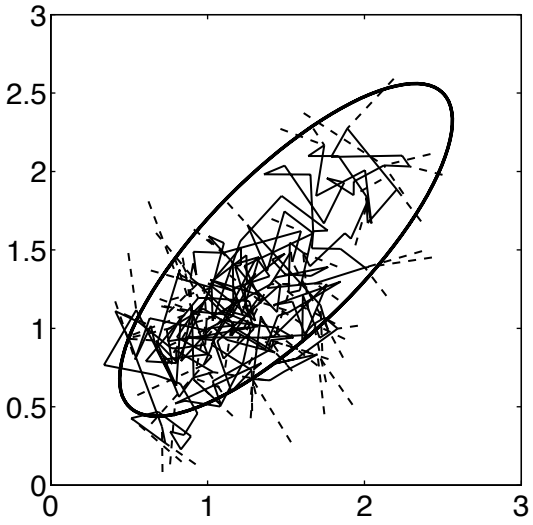

Figure 1.12: A simple illustration using Metropolis algorithm to sample from a normal distribution whose one standard deviation contour is shown by the ellipse. The proposal distribution is an isotropic normal distribution whose standard deviation is 0.2. Steps which are accepted are shown as solid lines, while rejected steps are shown dashed. A total of 300 candidate samples are generated, which include 85 rejections.

to be a series of random variables $x_1, \ldots, x_M$ such that the following conditional independence property holds for $m \in \{1, \ldots, M-1\}$

$$p(x_{m+1}|x_1, \ldots, x_m) = p(x_{m+1}|x_m). \tag{1.42}$$

This of course can be represented as a directed graph in the form of a chain. We can then specify the Markov chain by giving the probability distribution for the initial variable $p_0(x)$ together with the conditional probabilities for subsequent variables in the form of *transition probabilities* $T_m(x_m, x_{m+1}) \equiv p(x_{m+1}|x_m)$. A Markov chain is called *homogeneous* if the transition probabilities are the same for all m. Initially we will restrict attention to finite, discrete state spaces for the variables, and discuss the extension to countably infinite and continuous state spaces later.

The marginal probability for a particular variable can be expressed in terms of the marginal probability for the previous variable in the chain in the form

$$p(x_{m+1}) = \sum_{x_m} p(x_{m+1}|x_m)p(x_m). \tag{1.43}$$

A distribution is said to be invariant, or stationary, with respect to a Markov chain if each step in the chain leaves that distribution invariant. Thus, for a homogeneous Markov chain with transition probabilities $T(x, x')$, the distribution $p^*(x)$ is invariant if

$$p^*(x) = \sum_{x'} T(x', x)p^*(x'). \tag{1.44}$$

Note that a given Markov chain may have more than one invariant distribution. For instance, if

the transition probabilities are given by the identity transformation, then any distribution will be invariant.

A sufficient (but not necessary) condition for ensuring that the required distribution $p(x)$ is invariant, is to choose the transition probabilities to satisfy the property of *detailed balance*, defined by

$$p^*(x)T(x,x') = p^*(x')T(x',x) \tag{1.45}$$

for the particular distribution $p^*(x)$. It is easily seen that a transition probability which satisfies detailed balance with respect to a particular distribution will leave that distribution invariant, since

$$\sum_{x'} p^*(x')T(x',x) = \sum_{x'} p^*(x)T(x,x') = p^*(x)\sum_{x'} p(x'|x) = p^*(x). \tag{1.46}$$

A Markov chain which respects detailed balance is said to be *reversible*.

Our goal is to use Markov chains to sample from a given distribution. We can achieve this if we set up a Markov chain such that the desired distribution is invariant. However, we must also require that for, $m \to \infty$, the distribution $p_m(x)$ converges to the required invariant distribution $p^*(x)$, irrespective of the choice of initial distribution $p_0(x)$. This property is called *ergodicity*, and the invariant distribution is then called the *equilibrium* distribution. Clearly an ergodic Markov chain can have only one equilibrium distribution.

In Appendix **??** we show that a homogeneous Markov chain will be ergodic, subject only to weak restrictions on the invariant distribution and the transition probabilities.

In practice we often construct the transition probabilities from a set of 'base' transitions $B_1, \ldots, B_n$. This can be achieved through a mixture distribution of the form

$$T(x,x') = \sum_{k=1}^{n} \alpha_k B_k(x,x') \tag{1.47}$$

for some set of mixing coefficients $\alpha_1, \ldots, \alpha_n$ satisfying $\alpha_k \geq 0$ and $\sum_k \alpha_k = 1$. Alternatively, the base transitions may be combined through successive application, so that

$$T(x,x') = \sum_{x_1} \cdots \sum_{x_{n-1}} B_1(x,x_1)\ldots B_{n-1}(x_{n-2},x_{n-1})B_n(x_{n-1},x'). \tag{1.48}$$

If a distribution is invariant with respect to each of the base transitions, then obviously it will also be invariant with respect to either of the $T(x,x')$ given by (**??**) or (**??**). For the case of the mixture (**??**), if each of the base transitions satisfies detailed balance, then the mixture transition T will also satisfy detailed balance. This does not hold for the transition probability constructed using (**??**), although by symmetrizing the order of application of the base transitions, in the form $B_1, B_2, \ldots, B_n, B_n, \ldots, B_2, B_1$, detailed balance can be restored.

A common example of the use of composite transition probabilities is where each base transition changes only a subset of the variables. Clearly each base transition itself will not be ergodic. However, the composite probability will be ergodic, as long as condition (**??**) is satisfied.

1.2.2 The Metropolis-Hastings Algorithm

Earlier we introduced the basic Metropolis algorithm, without actually demonstrating that it samples from the required distribution. Before giving a proof we first discuss a generalization, known as the *Metropolis-Hastings* algorithm, to the case where the proposal distribution is no longer a symmetric function of its arguments. In particular at step n of the algorithm, in which the current state is x_n, we draw a sample x^* from the distribution $q_k(x, x_n)$ and then accept it with probability $A_k(x^*, x_n)$ where

$$A_k(x', x) = \min\left(1, \frac{p(x')q_k(x', x)}{p(x)q_k(x, x')}\right). \tag{1.49}$$

Here k labels the members of the set of possible transitions being considered. This is generally referred to as the Metropolis-Hastings algorithm. Obviously for a symmetric proposal distribution this reduces to the standard Metropolis criterion given by (**??**).

We can show that $p(x)$ is an invariant distribution of the Markov chain defined by the Metropolis-Hastings algorithm by showing that detailed balance, defined by (**??**), is satisfied. Using (**??**) we have

$$\begin{aligned} p(x)q_k(x, x')A_k(x', x) &= \min\left(p(x)q_k(x, x'), p(x')q_k(x', x)\right) \\ &= \min\left(p(x')q_k(x', x), p(x)q_k(x, x')\right) \\ &= p(x')q_k(x', x)A_k(x, x') \end{aligned} \tag{1.50}$$

as required.

It is worth noting that the evaluation of the acceptance criterion (**??**) does not require knowledge of the normalizing constant Z in the probability distribution $p(x) = \widetilde{p}(x)/Z$.

The specific choice of proposal distribution can have a marked effect on the performance of the algorithm. For continuous state spaces a common choice is a Gaussian centered on the current state, leading to an important trade-off in determining the width parameter of this distribution. If the width is small then the proportion of accepted transitions will be high, but progress through the state space takes the form of a slow random walk leading to very long correlation times. However, if the width parameter is large then the rejection rate will be high since, in the kind of complex problems we are considering, many of the proposed steps will be to states for which the probability $p(x)$ is low. Consider a multi-variate distribution $p(x)$ having strong correlations between the components of x, as illustrated in Figure **??**. The scale ρ of the proposal distribution should be as large as possible without incurring high rejection rates. This suggests that ρ should be of the same order as the smallest length scale $\sigma_{\min}$. The system then explores the distribution along the more extended direction by means of a random walk, and so the number of steps to arrive at a state which is more or less independent of the original state is of order $(\sigma_{\max}/\sigma_{\min})^2$. In fact in two dimensions the increase in rejection rate as ρ increases is offset by the larger steps sizes of those transitions which are accepted, and more generally for a multivariate Gaussian the number of steps required to obtain independent samples scales like $(\sigma_{\max}/\sigma_2)^2$ where σ_2 is the second smallest standard deviation. These details aside, it remains the case that if the length scales over which the distributions varies are very different in different directions then the Metropolis Hastings algorithm can lead to very slow convergence.

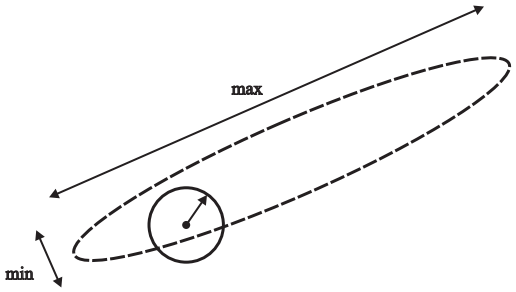

Figure 1.13: Schematic illustration of the use of an isotropic Gaussian proposal distribution (circle) to sample from a correlated multivariate Gaussian distribution (dashed ellipse) having very different standard deviations in different directions, using the Metropolis-Hastings algorithm. In order to keep the rejection rate low the scale ρ of the proposal distribution should be of the order of the smallest standard deviation $\sigma_{\min}$, which leads to random walk behaviour in which independent states are separated by roughly $(\sigma_{\max}/\sigma_{\min})^2$ steps where $\sigma_{\max}$ is the largest standard deviation.

1.3 Gibbs Sampling

Gibbs sampling is a simple and widely applicable Markov chain Monte Carlo algorithm, which has been used extensively in the context of probabilistic graphical models. As we shall see, Gibbs sampling can be seen as a special case of the Metropolis-Hastings algorithm.

Consider the distribution $p(x) = p(x_1, \ldots, x_d)$ from which we wish to sample, and suppose that we have chosen some initial state for the Markov chain. Each step of the Gibbs sampling procedure involves replacing the value of one of the variables by a value drawn from the distribution of that variable conditioned on the values of the remaining variables. Thus we replace x_i by a value drawn from the distribution $p(x_i|\{x_{j\neq i}\})$, where x_i denotes the ith component of x. This procedure is repeated either by cycling through the variables in some particular order, or by choosing the variable to be updated at each step at random from some distribution.

For example, suppose we have a distribution $p(x_1, x_2, x_3)$ over three variables, and at step τ of the algorithm we have selected values $x_1^{(\tau)}, x_2^{(\tau)}$ and $x_3^{(\tau)}$. We first replace $x_1^{(\tau)}$ by a new value $x_1^{(\tau+1)}$ obtained by sampling from the conditional distribution

$$p(x_1|x_2^{(\tau)}, x_3^{(\tau)}). \tag{1.51}$$

Next we replace $x_2^{(\tau)}$ by a value $x_2^{(\tau+1)}$ obtained by sampling from the conditional distribution

$$p(x_2|x_1^{(\tau+1)}, x_3^{(\tau)}) \tag{1.52}$$

so that the new value for x_1 is used straight away in subsequent sampling steps. Then we update x_3 with a sample $x_3^{(\tau+1)}$ drawn from

$$p(x_3|x_1^{(\tau+1)}, x_2^{(\tau+1)}) \tag{1.53}$$

and so on, cycling through the three variables in turn. The basic Gibbs sampling algorithm is summarized in Figure **??**.

To show that this procedure samples from the required distribution we first of all note that

1. Initialize $\{x_i : i = 1, \ldots, n\}$
2. For $\tau = 1, \ldots, T$.
   - Sample $x_1^{(\tau+1)} \sim p(x_1|x_2^{(\tau)}, x_3^{(\tau)}, \ldots, x_n^{(\tau)})$.
   - Sample $x_2^{(\tau+1)} \sim p(x_2|x_1^{(\tau+1)}, x_3^{(\tau)}, \ldots, x_n^{(\tau)})$.
   - $\vdots$
   - Sample $x_j^{(\tau+1)} \sim p(x_j|x_1^{(\tau+1)}, \ldots, x_{j-1}^{(\tau+1)}, x_{j+1}^{(\tau)}, \ldots, x_n^{(\tau)})$.
   - $\vdots$
   - Sample $x_n^{(\tau+1)} \sim p(x_n|x_1^{(\tau+1)}, x_2^{(\tau+1)}, \ldots, x_{n-1}^{(\tau+1)})$.

Figure 1.14: Psuedo-code for the Gibbs sampling algorithm.

the distribution $p(x)$ is an invariant of each the Gibbs sampling steps individually, and hence of the whole Markov chain. This follows from the fact that when we sample from $p(x_i|\{x_{j\neq i}\})$ the marginal distribution $p(\{x_{j\neq i}\})$ is clearly invariant since the values of the $\{x_{j\neq i}\}$ are unchanged. Also, each step by definition samples from the correct conditional distribution $p(x_i|\{x_{j\neq i}\})$. Since these conditional and marginal distributions together specify the joint distribution, we see that the joint distribution is itself invariant.

The second requirement to be satisfied in order that the Gibbs sampling procedure samples from the correct distribution is that it be ergodic. A sufficient condition for ergodicity is that none of the conditional distributions be anywhere zero. If this is the case then any point in x space can be reached from any other point in a finite number of steps involving one update of each of the component variables. If this requirement is not satisfied, so that some of the conditional distributions have zeros, then ergodicity, if it applies, must be proven explicitly.

The distribution of initial states must also be specified in order to complete the algorithm, although samples drawn after many iterations should become independent of this distribution. Of course successive samples from the Markov chain will be highly correlated, and so to obtain samples which are nearly independent it will be necessary to sub-sample the sequence.

Where appropriate we can generalize the Gibbs procedure slightly to sample from sets of variables, conditioned on the remaining variables, at each step. Note that these sets do not need to be disjoint.

We can obtain the Gibbs sampling procedure as a particular instance of the Metropolis-Hastings algorithm as follows. Consider a Metropolis-Hastings sampling step involving the group of variable x_k in which the remaining variables x_{-k} remain fixed, and for which the transition probability $q_k(x, x')$ is given by $p(x'_k|x_{-k})$. We note that $x'_{-k} = x_{-k}$ since these components are unchanged by the sampling step. Also, $p(x) = p(x_k|x_{-k})p(x_{-k})$. Thus the factor which determines the acceptance probability in the Metropolis-Hastings (**??**) is given by

$$\frac{p(x')q_k(x', x)}{p(x)q_k(x, x')} = \frac{p(x'_k|x'_{-k})p(x'_{-k})p(x_k|x'_{-k})}{p(x_k|x_{-k})p(x_{-k})p(x'_k|x_{-k})} = 1. \tag{1.54}$$

Thus the Metropolis-Hastings steps are always accepted, and therefore this choice of proposal distribution corresponds to the Gibbs sampling algorithm.

The practical applicability of Gibbs sampling depends on the ease with which samples can be

drawn from conditional distributions $p(x_k|x_{-k})$. In the case of graphical models, the conditional distributions for individual nodes depend only on the variables in the corresponding Markov blankets, as illustrated in Figure **??**. For directed graphs, a very broad choice of conditional distribu-

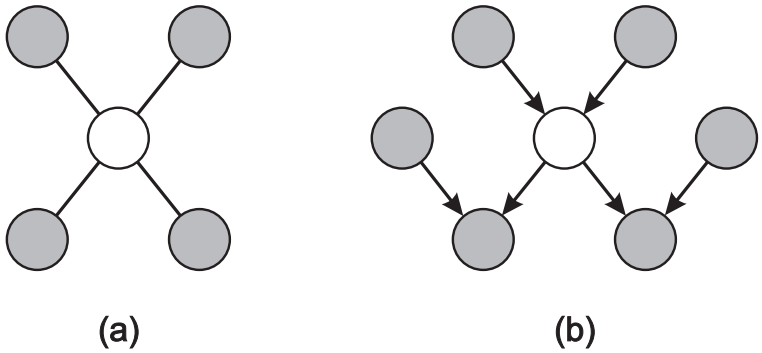

Figure 1.15: The Gibbs sampling requires samples to be drawn from the conditional distribution of a variable conditioned on the remaining variables. For graphical models, this conditional distribution is a function only of the states of the nodes in the Markov blanket. In the case of an undirected graph (a) this comprises the set of neighbours while for a directed graph (b) the Markov blanket comprises the parents, the children and the co-parents.

tions for the individual nodes conditioned on their parents will lead to conditional distributions for Gibbs sampling which are log concave. The adaptive rejection sampling methods discussed in Section **??** therefore provide a framework for Monte Carlo sampling from directed graphs with broad applicability.

As with the Metropolis algorithm, we can gain some insight into the behaviour of Gibbs sampling by investigating its application to a Gaussian distribution. Consider a correlated Gaussian in two variables, as illustrated in Figure **??**, having a conditional distribution of width l and a marginal distribution of width L. The typical step size is governed by the conditional distributions and will be of order l. Since the state evolves according to a random walk, the number of steps needed to obtain independent samples from the distribution will be of order $(L/l)^2$. Of course if the Gaussian distribution were uncorrelated then the Gibbs sampling procedure would be optimally efficient. For this simple problem we could rotate the coordinate system in order to decorrelate the variable, however, in practical applications it will generally be infeasible to find such transformations.

One approach to reducing random walk behaviour in Gibbs sampling is called *over-relaxation*. In its original form this applies to problems for which the conditional distributions are Gaussian. This represents a more general class of distributions than the multi-variate Gaussian, since for example the non-Gaussian distribution $p(x, y) \propto \exp(-x^2y^2)$ has Gaussian conditional distributions. At each step of the Gibbs sampling algorithm the conditional distribution for a particular component x_i has some mean μ_i and some variance σ_i^2. In the over-relaxation framework the value of x_i is replaced with

$$x_i' = \mu_i + \alpha(x_i - \mu_i) + \sigma_i(1 - \alpha_i^2)^{1/2}\nu \tag{1.55}$$

where ν is a Gaussian random variate with zero mean and unit variance, and α is a parameter such that $-1 < \alpha < 1$. For $\alpha = 0$ the method is equivalent to standard Gibbs sampling. When α is negative the step is biased to the opposite side of the mean. It is easily seen that this step leaves the desired distribution invariant since if x_i has mean μ_i and variance σ_i^2, then so too does x_i'. Also it is clear that if the original Gibbs sampling is ergodic, then the over-relaxed version will be ergodic

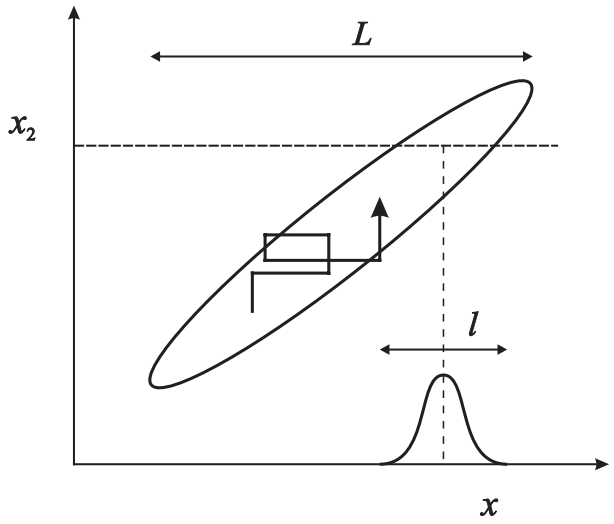

Figure 1.16: Illustration of Gibbs sampling by alternate updates of two variables whose distribution is a correlated Gaussian. The step size is governed by standard deviation of the conditional distribution, and is $O(l)$, leading to slow progress in the direction of elongation of the distribution. The number of steps needed to obtain an independent sample from the distribution is $O((L/l)^2)$.

also. Thus the effect of over-relaxation is to tend to produce directed motion through state space when the variables are highly correlated. The framework of *ordered over-relaxation* generalizes this approach to non-Gaussian distributions.

The simplicity of Gibbs sampling allows it to be applied to a broad range of models. In fact it has been used as the basis for a general purpose software package BUGS ('Bayesian inference Using Gibbs Sampling') for sampling from models specified as directed acyclic graphs. The conditional distribution for each node is dependent only on the states of nodes in the corresponding Markov blanket. If the graph is constructed using distributions from the exponential family, and if the parent-child relationships preserve conjugacy, then the full conditional distributions arising in Gibbs sampling will take the same form as the original conditional distributions (conditioned on the parents) defining each node, and so standard sampling techniques can be employed. In general the full conditional distributions will be of a complex form that does not permit the use of standard sampling algorithms. However, these conditionals will be log concave and sampling can be done efficiently using adaptive rejection sampling (assuming the corresponding variable is a scalar).

Because the basic Gibbs sampling technique considers one variable at a time, there are strong dependencies between successive samples. At the opposite extreme, if we could draw samples directly from the joint distribution (an operation that we are supposing is intractable) then successive samples would be independent. We can hope to improve on the simple Gibbs sampler by sampling successively from groups of variables rather than individual variables. This is achieved in the *blocking Gibbs* sampling algorithm by choosing blocks of variables, not necessarily disjoint, and then sampling jointly from the variables in each block in turn, conditioned on the remaining variables. This can be done in a way that preserves tractability while leading to a large block by starting with the junction tree for the full graph in which the block comprises the complete set of variables, and then removing variables from the block until a tractable structure is obtained.

1.4 Slice Sampling

We have seen that one of the difficulties with the Metropolis algorithm is the sensitivity to step size. If this is too small the result is slow decorrelation due to random walk behaviour while if it is too large the result is inefficiency due to a high rejection rate. The technique of *slice sampling* provides an adaptive step size which is automatically adjusted to match the characteristics of the distribution. Again it requires that we be able to evaluate the unnormalized distribution $\widetilde{p}(x)$.

Slice sampling involves augmenting x with an additional variable u and then drawing samples from the joint (x, u) space. We shall see another example of this approach when we discuss hybrid Monte Carlo in Section **??**. As with rejection sampling, the goal is to sample uniformly from the volume under the surface defined by $\widehat{p}(x, u)$, in other words to sample from the distribution given by

$$\widehat{p}(x,u) = \begin{cases} 1/Z & \text{if } 0 \leqslant u \leqslant p(x) \\ 0 & \text{otherwise} \end{cases}$$

where $Z = \int \widetilde{p}(x)\, dx$. The marginal distribution over x is given by

$$\int \widehat{p}(x,u)\, du = \int_0^{\widetilde{p}(x)} \frac{1}{Z}\, du = \frac{\widetilde{p}(x)}{Z} = p(x)$$

and so we can sample from $p(x)$ by sampling from $\widehat{p}(x, u)$ and then ignoring the u values. This can be achieved by alternately sampling x and u. Given the value of x we evaluate $\widetilde{p}(x)$ and then sample u uniformly in the range $0 \leqslant u \leqslant \widetilde{p}(x)$, which is straightforward. Then we fix u and sample x uniformly from the 'slice' through the distribution defined by $\{x : \widetilde{p}(x) > u\}$. This is illustrated in Figure **??**(a).

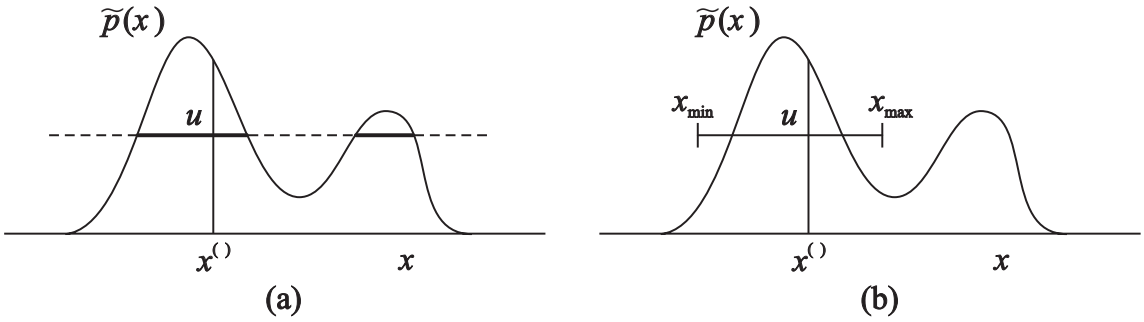

Figure 1.17: Illustration of slice sampling. (a) For a given value $x^{(\tau)}$, a value of u is chosen uniformly in the region $0 \leqslant u \leqslant \widetilde{p}(x^{(\tau)})$, which defines a 'slice' through the distribution, shown by the solid horizontal lines. (b) Since sampling directly from a slice is infeasible, a new sample of x is drawn from a region $x_{\min} \leqslant x \leqslant x_{\max}$ which contains the previous value $x^{(\tau)}$.

In practice it can be difficult to sample directly from a slice through the distribution and so instead we define a sampling scheme which leaves the uniform distribution under $\widehat{p}(x, u)$ invariant, which can be achieved by ensuring that detailed balance is satisfied. Here we consider the case of a univariate x.

Suppose the current value of x is denoted $x^{(\tau)}$ and that we have obtained a corresponding sample u. The next value of x is sampled uniformly from a region $x_{\min} \leqslant x \leqslant x_{\max}$ which contains

$x^{(\tau)}$. If the sample chosen lies within the slice, in other words if $\widetilde{p}(x) > u$, then the sample is retained and forms $x^{(\tau+1)}$.

It is in the choice of the region $x_{\min} \leqslant x \leqslant x_{\max}$ that the adaptation to the characteristic length scales of the distribution takes place. We want the region to encompass as much of the slice as possible so as to allow large moves in x space, while having as little as possible of this region lying outside the slice, since this makes the sampling less efficient.

One approach to the choice of region involves starting with a region containing $x^{(\tau)}$ having some width w and then testing each of the end points to see if they lie within the slice. If either end point does not then the region is extended in that direction by increments of value w until the endpoint lies outside the region. A candidate value x' is then chosen uniformly from this region and if it lies within the slice then it forms $x^{(\tau+1)}$. If it lies outside the slice then the region is shrunk such that x' forms an end point and such that the region still contains $x^{(\tau)}$. Then another candidate point is drawn uniformly from this reduced region and so on, until a value of x is found which lies within the slice.

Slice sampling can be applied to multivariate distributions by repeatedly sampling each variable in turn, in the manner of Gibbs sampling. This requires that we are able to compute, for each component x_i, a function which is proportional to $p(x_i|\{x_j\}_{j\neq i})$.

1.5 The Hybrid Monte Carlo Algorithm

As we have already noted, one of the major limitations of the Metropolis algorithm is that it can exhibit random walk behaviour whereby the distance traversed through the state space grows only as the square root of the number of steps. The problem cannot be resolved simply by taking bigger steps as this leads to a high rejection rate.

In this section we introduce a more sophisticated class of transitions based on an analogy with physical systems and which has the property of being able to make large changes to the system state while keeping the rejection probability small. It is applicable to distributions over continuous variables for which we can readily evaluate the gradient of the log probability with respect to the state variables. We discuss the dynamical systems framework in Section **??**, and then in Section **??** we explain how this may be combined with the Metropolis algorithm to yield the powerful hybrid Monte Carlo algorithm.

1.5.1 Dynamical Systems

The dynamical approach to stochastic sampling has its origins in algorithms for simulating the behaviour of physical systems evolving under Hamiltonian dynamics. In a Markov chain Monte Carlo simulation the goal is to sample from a given probability distribution $p(x)$. The framework of Hamiltonian dynamics is exploited by casting the probabilistic simulation in the form of a Hamiltonian system. In order to remain in keeping with the literature in this area we make use of the relevant dynamical systems terminology where appropriate. This will be defined as we go along, and no background knowledge of the physical basis for this formalism is required.

The dynamics which we consider corresponds to the evolution of the state variable $x = \{x_i\}$ under continuous time, which we denote by τ. Classical dynamics is described by second order differential equations over time (acceleration is proportional to the applied force). We can decompose a second order equation into two coupled first order equations by introducing intermediate *momentum* variables r corresponding to the rate of change of the state variables x. Thus

$$r_i = \frac{dx_i}{d\tau} \tag{1.56}$$

where the x_i can be regarded as *position* variables in this dynamics perspective. Thus for each position variable there is a corresponding momentum variable, and the joint space of position and momentum variables is called *phase space*.

Without loss of generality we can write the probability distribution $p(x)$ in the form

$$p(x) = \frac{1}{Z} \exp\left(-E(x)\right) \tag{1.57}$$

where $E(x)$ is interpreted as the *potential energy* of the system when in state x. The system acceleration is the rate of change of momentum and is given by the applied *force* which itself is the negative gradient of the potential energy

$$\frac{dR_i}{d\tau} = -\frac{\partial E(x)}{\partial x_i}. \tag{1.58}$$

It is convenient to reformulate this dynamical system using the Hamiltonian framework. To do this, we first define the *kinetic energy* by

$$K(r) = \frac{1}{2}\|r\|^2 = \frac{1}{2}\sum_i r_i^2. \tag{1.59}$$

The total energy of the system is then the sum of its potential and kinetic energies

$$H(x, r) = E(x) + K(r) \tag{1.60}$$

which is called the *Hamiltonian* function. Using (**??**), (**??**), (**??**) and (**??**) we can now express the dynamics of the system in terms of the Hamiltonian equations given by

$$\frac{dx_i}{d\tau} = \frac{\partial H}{\partial r_i} \tag{1.61}$$

$$\frac{dr_i}{d\tau} = -\frac{\partial H}{\partial x_i}. \tag{1.62}$$

$$\tag{1.63}$$

During the evolution of this dynamical system, the value of the Hamiltonian H is constant, as is easily seen by differentiation

$$\begin{aligned} \frac{dH}{d\tau} &= \sum_i \left\{ \frac{\partial H}{\partial x_i}\frac{dx_i}{d\tau} + \frac{\partial H}{\partial r_i}\frac{dr_i}{d\tau} \right\} \\ &= \sum_i \left\{ \frac{\partial H}{\partial x_i}\frac{\partial H}{\partial r_i} - \frac{\partial H}{\partial r_i}\frac{\partial H}{\partial x_i} \right\} = 0. \end{aligned} \tag{1.64}$$

A second important property of Hamiltonian dynamical systems, known as *Liouville's Theorem*, is that they preserve volume in phase space. This can be seen by noting that the flow field (rate of

change of location in phase space) is given by

$$V = \left(\frac{dx}{d\tau}, \frac{dr}{d\tau} \right) \tag{1.65}$$

and that the divergence of this field vanishes

$$\text{div } V = \sum_i \left\{ \frac{\partial}{\partial x_i} \frac{\partial H}{\partial x_i} + \frac{\partial}{\partial r_i} \frac{\partial H}{\partial r_i} \right\} = \sum_i \left\{ \frac{\partial^2 H}{\partial x_i \partial r_i} - \frac{\partial^2 H}{\partial r_i \partial x_i} \right\} = 0. \tag{1.66}$$

Now consider the joint distribution over phase space whose total energy is the Hamiltonian, i.e. the distribution given by

$$p(x, r) = \frac{1}{Z_H} \exp(-H(x, r)). \tag{1.67}$$

Using the two results of conservation of volume and conservation of H it follows that the Hamiltonian dynamics will leave $p(x, r)$ invariant. This can be seen by considering a small region of phase space over which H is approximately constant. If we follow the evolution of the Hamiltonian equations for a finite time, then the volume of this region will remain unchanged as will the value of H in this region, and hence the probability density, which is a function only of H, will also be unchanged.

Although H is invariant, the values of x and r will vary, and so by integrating the Hamiltonian dynamics over a finite time duration it becomes possible to make large changes to x in a systematic way which avoids random walk behaviour.

Evolution under the Hamiltonian dynamics will not, however, sample ergodically from $p(x, r)$ since the value of H is constant. In order to make arrive at an ergodic sampling scheme we can introduce additional moves in phase space which change the value of H while also leaving the distribution $p(x, r)$ invariant. The simplest way to achieve this is to replace the value of r with one drawn from its distribution conditioned on x. This can be regarded as a Gibbs sampling step, and hence from Section **??** we see that this leaves the desired distribution invariant. Noting that x and r are independent in the distribution $p(x, r)$ we see that the conditional distribution of r is a Gaussian from which it is straightforward to sample.

In a practical application of this approach we have to address the problem of performing a numerical integration of the Hamiltonian equations. This will necessarily introduce numerical errors and so we should devise a scheme which minimizes the impact of such errors. In fact it turns out that integration schemes can be devised for which Liouville's theorem still holds exactly. This property will be important in the hybrid Monte Carlo algorithm which is discussed in Section **??**. One scheme for achieving this is called the *leapfrog* discretization and involves alternately updating discrete-time approximations $\widehat{x}$ and $\widehat{r}$ to the position and momentum variables using

$$\widehat{r}_i(\tau + \epsilon/2) = \widehat{r}_i(\tau) - \frac{\epsilon}{2} \frac{\partial E}{\partial x_i}(\widehat{x}(\tau)) \tag{1.68}$$

$$\widehat{x}_i(\tau + \epsilon) = \widehat{x}_i(\tau) + \epsilon \widehat{r}_i(\tau + \epsilon/2) \tag{1.69}$$

$$\widehat{r}_i(\tau + \epsilon) = \widehat{r}_i(\tau + \epsilon/2) - \frac{\epsilon}{2} \frac{\partial E}{\partial x_i}(\widehat{x}(\tau + \epsilon)). \tag{1.70}$$

We see that this takes the form of a half-step update of the momentum variables with step size $\epsilon/2$, followed by a full-step update of the position variables with step size ϵ, followed by a second half-step update of the momentum variables. If several leapfrog steps are applied in succession, it can be seen that half-step updates to the momentum variables can be combined into full-step updates with step size ϵ. The successive updates to position and momentum variables then leapfrog over each other. In order to advance the dynamics by a time interval τ we need to take τ/ϵ steps. The error involved in the discretized approximation to the continuous time dynamics will go to zero, assuming a smooth function $E(x)$, in the limit $\epsilon \to 0$. However, for a non-zero ϵ as used in practice, some residual error will remain. We shall see in Section **??** how the effects of such errors can be eliminated in the hybrid Monte Carlo algorithm.

In summary then, the Hamiltonian dynamical approach involves alternating between a series of leapfrog updates and a re-sampling of the momentum variables from their marginal distribution.

Note that the Hamiltonian dynamics method, unlike the basic Metropolis algorithm, is able to make use of information about the gradient of the log probability distribution as well as about the distribution itself. An analogous situation is familiar from the domain of function optimization. In most cases where gradient information is available it is highly advantageous to make use of it. Informally, this follows from the fact that in a space of dimension d the additional computational cost of evaluating a gradient compared to evaluating the function itself will typically be a fixed factor independent of d, whereas the gradient vector conveys d pieces of information compared to the one piece of information given by the function itself.

1.5.2 Hybrid Monte Carlo

As we discussed in the previous section, for a non-zero step size ϵ the discretization of the leapfrog algorithm will introduce errors into the integration of the Hamiltonian dynamical equations. Hybrid Monte Carlo combines Hamiltonian dynamics with the Metropolis algorithm and thereby removes any bias associated with the discretization.

Specifically, the algorithm uses a Markov chain consisting of alternate stochastic updates of the momentum variables r and Hamiltonian dynamical updates using the leapfrog algorithm. After each application of the leapfrog algorithm the resulting candidate state is accepted or rejected according to the Metropolis criterion based on the value of the Hamiltonian H. Thus if (x, r) is the initial state and (x^*, r^*) is the state after the leapfrog integration, then this candidate state is accepted with probability

$$\min\left(1, \exp\{H(x, r) - H(x^*, r^*)\}\right). \tag{1.71}$$

If the leapfrog integration were to simulate the Hamiltonian dynamics perfectly, then every such candidate step would automatically be accepted since the value of H would be unchanged. Due to numerical errors, the value of H may sometimes decrease, and we would like the Metropolis criterion to remove any bias due to this effect and ensure that the resulting samples are indeed drawn from the required distribution. In order for this to be the case, we need to ensure that the update equations corresponding to the leapfrog integration satisfy detailed balance (**??**). This is easily achieved by modifying the leapfrog scheme as follows. Before the start of each leapfrog integration sequence we choose at random, with equal probability, whether to integrate forwards in time (using step size ϵ) or backwards in time (using step size $-\epsilon$). We first note that the leapfrog integration scheme (**??**)–(**??**) is time-reversible, so that integration for L steps using stepsize $-\epsilon$ will exactly undo the effect of integration for L steps using stepsize ϵ.

Next we show that the leapfrog integration preserves phase space volume exactly. This follows from the fact that each step in the leapfrog scheme updates either the x_i variable or the r_i variable by an amount which is a function only of the other variable. As shown in Figure **??** this has the

effect of shearing a region of phase space while not altering its volume.

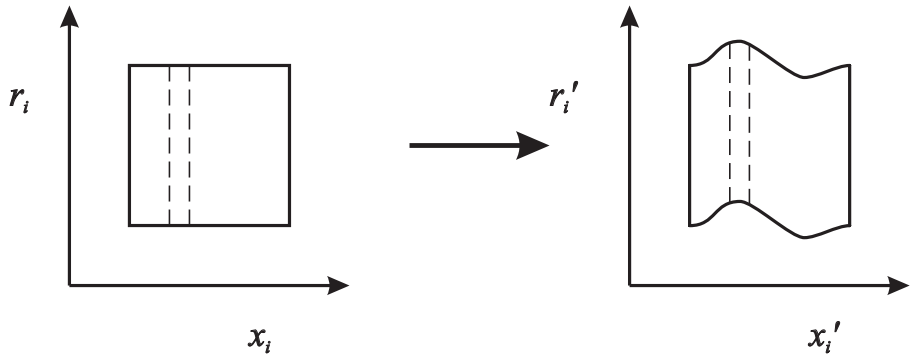

Figure 1.18: Each step of the leapfrog algorithm (**??**)–(**??**) modifies either a position variable x_i or a momentum variable r_i. Since the change to one variable is a function only of the other, any region in phase space will be sheared without change of volume.

Finally, we use these results to show that detailed balance holds. Consider a small region $\mathcal{R}$ of phase space which, under a sequence of L leapfrog iterations of step size ϵ, maps to a region $\mathcal{R}'$. Using conservation of volume under the leapfrog iteration we see that if $\mathcal{R}$ has volume δV then so too will $\mathcal{R}'$. If we choose an initial point from the distribution (**??**) and then update using L leapfrog interactions, the probability of the transition going from $\mathcal{R}$ to $\mathcal{R}'$ is given by

$$\frac{1}{Z_H}\exp(-H(\mathcal{R}))\delta V\frac{1}{2}\min\left\{1,\exp(-H(\mathcal{R})+H(\mathcal{R}'))\right\}. \tag{1.72}$$

where the factor of $1/2$ arises from the probability of choosing to integrate with a positive step size rather than a negative one. Similarly, the probability of starting in region $\mathcal{R}'$ and integrating backwards in time to end up in region $\mathcal{R}$ is given by

$$\frac{1}{Z_H}\exp(-H(\mathcal{R}'))\delta V\frac{1}{2}\min\left\{1,\exp(-H(\mathcal{R}')+H(\mathcal{R}))\right\}. \tag{1.73}$$

It is easily seen that the two probabilities (**??**) and (**??**) are equal, and hence detailed balance holds[2].

It is not difficult to construct examples for which the leapfrog algorithm returns to its starting position after a particular number of iterations. In such cases the random replacement of the momentum values before each leapfrog integration will not be sufficient to ensure ergodicity since the position variables will never be updated. Such phenomena are easily avoided by choosing the magnitude of the step size at random from some small interval, before each leapfrog integration.

We can gain some insight into the behaviour of the hybrid Monte Carlo algorithm by considering its application to a multivariate Gaussian. For convenience consider a Gaussian distribution $p(x)$ with independent components, for which the Hamiltonian is given by

$$H(x,r)=\frac{1}{2}\sum_i\frac{1}{\sigma_i^2}x_i^2+\frac{1}{2}\sum_i r_i^2. \tag{1.74}$$

Our conclusions will be equally valid for a Gaussian distribution having correlated components

[2]This proof ignores any overlap between the regions $\mathcal{R}$ and $\mathcal{R}'$, but is easily generalized to allow for such overlap.

since the hybrid Monte Carlo algorithm exhibits rotational isotropy. During the leapfrog integration each pair of phase space variables x_i, r_i evolves independently. However, the acceptance or rejection of the candidate point is based on the value of H which depends on the values of all of the variables. Thus, a significant integration error in any one of the variables could lead to a high probability of rejection. In order that the discrete leapfrog integration be a reasonably good approximation to the true continuous-time dynamics it is necessary for the leapfrog integration scale ϵ to be smaller than the shortest length-scale over which the potential is varying significantly. This is governed by the smallest value of σ_i which we denote by $\sigma_{\min}$. Recall that the goal of the leapfrog integration in hybrid Monte Carlo is move to a substantial distance through phase space to a new state which is relatively independent of the initial state and still achieve a high probability of acceptance. In order to achieve this the leapfrog integration must be continued for a number of iterations of order $\sigma_{\max}/\sigma_{\min}$.

By contrast, consider the behaviour of a simple Metropolis algorithm with an isotropic Gaussian proposal distribution of variance s^2. In order to avoid high rejection rates the value of s must be of order $\sigma_{\min}$. The exploration of state space then proceeds by a random walk, and takes of order $(\sigma_{\max}/\sigma_{\min})^2$ steps to arrive at a roughly independent state.

1.6 Estimating the Partition Function

As we have seen, most of the sampling algorithms considered in this chapter require only the functional form of the probability distribution up to a multiplicative constant. Thus if we write

$$p_E(x) = \frac{1}{Z_E} \exp(-E(x)) \tag{1.75}$$

then the value of the partition function Z is not needed in order to draw samples from $p(x)$. However, knowledge of the value of Z can be useful for Bayesian model comparison, and so it is of interest to consider how its value might be obtained. We assume that direct evaluation by summing, or integrating, the function $\exp(-E(x))$ over the state space of x is intractable.

For model comparison it is actually the ratio of the partition functions for two models which is required. Multiplication of this ratio by the ratio of prior probabilities gives the ratio of posterior probabilities, which can then be used for model selection or model averaging.

One way to estimate a ratio of partition functions is to use importance sampling from a distribution with energy function $G(x)$

$$\begin{aligned}
\frac{Z_E}{Z_G} &= \frac{\sum_x \exp(-E(x))}{\sum_x \exp(-G(x))} \\
&= \frac{\sum_x \exp(-E(x) + G(x)) \exp(-G(x))}{\sum_x \exp(-G(x))} \\
&= \langle \exp(-E + G) \rangle_G \\
&\simeq \sum_i \exp(-E(x_i) + G(x_i))
\end{aligned} \tag{1.76}$$

where $\{x_i\}$ are samples drawn from the distribution defined by $p_G(x)$. If the distribution p_G is one for which the partition function can be evaluated analytically, for example a Gaussian, then the

absolute value of Z_E can be obtained.

This approach will only yield accurate results if the importance sampling distribution p_G is closely matched to the distribution p_E, so that the ratio $E(x)/G(x)$ does not have wide variations. In practice suitable analytically specified importance sampling distributions cannot readily be found for the kinds of complex models considered in this book.

An alternative approach is therefore to use the samples obtained from a Markov chain to define the importance sampling distribution. If the transition probability for the Markov chain is given by $T(x, x')$, and the sample set is given by $x_1, \ldots, x_N$, then the sampling distribution can be written as

$$\frac{1}{Z_G} \exp(-G) = \sum_{n=1}^{N} T(x_n, x) \tag{1.77}$$

which can be used directly in (**??**).

Methods for estimating the ratio of two partition functions require for their success that the two corresponding distributions be reasonably closely matched. This is especially problematic if we wish to find the absolute value of the partition function for a complex distribution since it is only for relatively simple distributions that the partition function can be evaluated directly, and so attempting to estimate the ratio of partition functions directly is unlikely to be successful. This problem can be tackled using a technique known as *chaining* which involved introducing a succession of intermediate distributions $P_2, \ldots, P_{M-1}$ which interpolate between a simple distribution $P_1(x)$ for which we can evaluate the normalization Z_1, and the desired complex distribution $P_M(x)$. We then have

$$\frac{Z_M}{Z_1} = \frac{Z_2}{Z_1}\frac{Z_3}{Z_2} \cdots \frac{Z_M}{Z_{M-1}} \tag{1.78}$$

in which the intermediate ratios can be determined using Monte Carlo methods as discussed above. One way to construct such a sequence of intermediate systems is to use an energy function containing a continuous parameter $0 \leq \alpha \leq 1$ which interpolates between the two distributions

$$E_\alpha(x) = (1 - \alpha)E_1(x) + \alpha E_M(x). \tag{1.79}$$

If the intermediate ratios in (**??**) are to be found using Monte Carlo, it may be more efficient to use a single Markov chain run than to restart the Markov chain for each ratio. In this case the Markov chain is run initially for the system P_1 and then after some suitable number of steps moves on to the next distribution in the sequence. Note, however, that the system must remain close to the equilibrium distribution at each stage.

1.7 Simulated Annealing

In order to apply sampling methods to inference tasks involving graphical models, a variety of matters must be addressed. Our goal here is not to provide an extensive practitioners manual, but rather to highlight some of the more significant issues.

Since a Markov chain takes some time to reach its equilibrium distribution it is common to discard the values obtained from the early part of the chain (called the 'burn in' period). This highlights an important trade-off in running Markov chain samplers with limited computational resources. For the same computational expense it will be possible to run one long Markov chain,

or alternatively several short ones. With a single chain there is only one burn in period. However, such a chain may become stuck exploring a relatively isolated region of the distribution while failing to discover other regions of significant probability.

An important concern in practice is the speed of convergence of a Markov chain to its equilibrium distribution, as well as the decorrelation time between independent samples. This can become particularly problematic in situations where the distribution has relatively isolated regions of high probability such that transitions from one region to another are infrequent. One technique for addressing this problem is called *simulated annealing*. Consider the problem of sampling from a distribution of the form

$$p(x) = \frac{1}{Z} \exp(-E(x)). \tag{1.80}$$

We first modify the distribution by introduction of a *temperature* parameter T to give

$$p(x) = \frac{1}{Z(T)} \exp(-E(x)/T) \tag{1.81}$$

so that the original distribution corresponds to $T = 1$. The effect of using values of $T > 1$ is to deemphasize the dynamic range between regions of high and low probability and hence increase the probability of transition from one high probability region to another through an intervening region of low probability. During the course of the simulation the value of T is gradually reduced from a high value down to $T = 1$. Note that this procedure no longer samples from the correct distribution, but it may be used to locate a region of high probability, after which the simulation is continued using $T = 1$ to allow samples from the correct distribution to be obtained. The method can also be used as an optimization technique in which T is gradually reduced to zero, so that the samples converge onto a (local) minimum of the function $E(x)$.

A variant of the simulated annealing approach involves decomposing the energy function $E(x)$ into the sum of a term $E_0(x)$ which has nice properties (for example it may be separable or it may be a convex function) and a term $E_1(x)$ which represents the difference between $E_0(x)$ and the true energy $E(x)$. Samples are then drawn from the distribution

$$p(x) = \frac{1}{Z(T)} \exp(-E_0(x) + E_1(x)/T) \tag{1.82}$$

with initially $T \gg 1$, and again T is gradually reduced to 1. For example $E_0(x)$ may correspond to the prior, and $E_1(x)$ may represent the contribution from the likelihood function.

An important consideration in the use of simulated annealing is the choice of annealing schedule for the reduction in T. In the simplest case this is pre-defined, for example a geometric reduction in T obtained by multiplying T by a fixed constant $0 < \alpha < 1$ after each iteration, although in more sophisticated approaches the reduction in T may be governed by the observed sequence of sampled values.

Although simulated annealing does not quite sample from the correct distribution, the related technique of *simulated tempering* avoids this problem. It involves the introduction of a fixed set of temperature values of which $T = 1$ is the lowest. The state of the temperature becomes a stochastic variable and is itself sampled as part of the Markov chain, and a modification to the energy function encourages the system to spend roughly equal times at the different temperatures. At higher temperature values the system can move more freely from one region of state space to another, whereas when the system has temperature $T = 1$ the state space samples are drawn from the required distribution.

1.8 Historical Remarks and Bibliography

Markov chain Monte Carlo methods have their origins in physics, and it was only towards the end of the 1980s that they started to have significant impact in the field of statistics. The first published paper to describe the Monte Carlo methods was **?**), and the Metropolis algorithm was introduced by **?**). An important contribution by **?**) showed that the Metropolis algorithm, and its generalization the Metropolis-Hastings algorithm, are members of a large family of possible algorithms.

The logic sampling method for directed graphs with evidence was introduced by **?**). Likelihood weighted sampling was proposed by **?**), and also by **?**) who also proposed the 'self-importance' sampling and 'Markov blanket scoring' extensions.

Adaptive rejection sampling was developed by **?**) and extended to the derivative-free case by **?**). The adaptive rejection Metropolis sampling technique for non log-concave distributions is described in **?**).

The particle filtering, or sequential Monte Carlo, approach has appeared in the literature under various names including the *bootstrap filter* (**?**), *survival of the fittest* (**?**) and *condensation* (**?**). It has been applied to Markov decision processes by **?**).

Gibbs sampling was popularized by the work of **?**), and the BUGS software, which builds heavily on the Gibbs sampling framework, is described in **?**). The technique of blocking Gibbs was proposed by **?**).

Slice sampling was introduced by **?**), who also provides a proof that it samples from the desired distribution. The hybrid Monte Carlo algorithm was introduced originally in the physics literature by **?**), and later applied to Bayesian inference for neural networks by **?**).

Diagnostic tests for convergence of Markov chain Monte Carlo algorithms are summarized in Chapter 8 of **?**).

There are numerous texts dealing with Monte Carlo methods. Those of particular interest from the statistical inference perspective include **?**), **?**), **?**), **?**), **?**), **?**) and **?**). Also there are review articles by **?**), **?**), **?**), **?**), **?**), **?**) and **?**) which provide additional information on sampling methods for statistical inference.

Exercises

1.1 (⋆) Show that the finite sample estimator (**??**) has mean $\langle f \rangle$ and variance σ^2/L where σ^2 is defined by (**??**).

1.2 (⋆) Let x be a d-dimensional random variable having a Gaussian distribution with zero mean and unit covariance matrix, and suppose that the positive definite symmetric matrix Σ has the Cholesky decomposition $\Sigma = LL^{\mathrm{T}}$. Show that the variable $y = \mu + Lx$ has a Gaussian distribution with mean μ and covariance Σ. This provides a technique for generating samples from a general multivariate Gaussian using samples from a univariate Gaussian having zero mean and unit variance.

1.3 (⋆) Suppose that z has a uniform distribution over the interval $[0, 1]$. Show that the variable $x = b \tan z + c$ has a Cauchy distribution given by (**??**).

1.4 (⋆⋆) Determine expressions for the coefficients k_i in the envelope distribution (**??**) for adaptive rejection sampling using the properties of continuity and normalization. Hence determine a scheme for sampling from this distribution.

1.5 (⋆) Use the sum and product rules of probability to verify directly the conditional independence properties $p(x_{t+1}|x_t, y_{(t)}) = p(x_{t+1}|x_t)$ and $p(y_t|x_t, y_{(t-1)}) = p(y_t|x_t)$ for the graph in Figure **??**.

1.6 (⋆) Show that the simple random walk over the integers defined by (**??**) has the property that $\langle x_n^2 \rangle = \langle x_{n-1}^2 \rangle + 1/2$ and hence by induction that $\langle x_n^2 \rangle = n/2$.

1.7 (⋆⋆) Show that the Gibbs sampling algorithm, discussed in Section **??**, satisfies detailed balance as defined by (**??**).

1.8 (⋆⋆) Consider the simple 3-node graph shown in Figure **??** in which the observed node x is given by a Gaussian distribution $\mathcal{N}(x|\mu, \tau)$ with mean μ and precision τ. Suppose that the

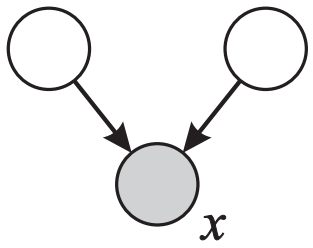

Figure 1.19: A graph involving an observed Gaussian variable x with prior distributions over its mean μ and precision τ.

marginal distributions over the mean and precision are given by $\mathcal{N}(\mu|\mu_0, s_0)$ and $\mathrm{Gam}(\tau|a, b)$, where $\mathrm{Gam}(\cdot|a, b)$ denotes a Gamma distribution. Write down expressions for the conditional distributions $p(\mu|x, \tau)$ and $p(\tau|x, \mu)$ which would be required in order to apply Gibbs sampling from the posterior distribution $p(\mu, \tau|x)$.

1.9 (⋆) Verify that the over relaxation update (**??**), in which x_i has mean μ_i and variance σ_i, and where ν has zero mean and unit variance, gives a value x_i' with mean μ_i and variance σ_i^2.

.1 Ergodicity of Markov Chains

In this appendix we show that a homogeneous Markov chain will be ergodic, subject only to a weak restriction on the invariant distribution and the transition probabilities.

Theorem .1 *Consider a transition probability $T(x', x)$ defining the relation between the marginal probabilities of x at subsequent steps of a Markov chain*

$$p_{m+1}(x) = \sum_{x'} T(x', x) p_m(x'). \tag{83}$$

together with a particular distribution $p^*(x)$ which is an invariant distribution of $T(x', x)$ and which is such that

$$\nu = \min_x \min_{\{x': p^*(x')>0)\}} \frac{T(x, x')}{p^*(x)} > 0. \tag{84}$$

The corresponding Markov chain will be ergodic, so that for any choice of initial distribution $p_0(x)$

$$\lim_{m\to\infty} p_m(x) = p^*(x). \tag{85}$$

Proof: The proof uses induction to show that the distribution at step m can be written as a mixture of the invariant distribution and some other distribution. At each step the proportion of the invariant component cannot decrease, while the condition (**??**) ensures that the remaining component will generate some contribution to the invariant component.

We suppose that at step m the following holds

$$p_m(x) = [1 - (1 - \nu)^m]p^*(x) + (1 - \nu)^m r_m(x) \tag{86}$$

where $r_m(x)$ is an arbitrary non-negative and normalized function, which can represents a valid probability distribution. Since we cannot have $p^*(x^z prime) < T(x', x)$ for all x', we must have $\nu < 1$, and so $p_m(x)$ is a convex combination of two distributions and hence is also a valid probability distribution. The result (**??**) clearly holds for $m = 0$, with $r_0(x) = p_0(x)$. We now show that the equivalent result must hold at step $m + 1$. Using (**??**) we have

$$\begin{aligned}
p_{m+1}(x) &= \sum_{x'} p_m(x')T(x', x) \\
&= [1 - (1 - \nu)^m] \sum_{x'} p^*(x')T(x', x) + (1 - \nu)^m \sum_{x'} r_m(x')T(x', x) \\
&= [1 - (1 - \nu)^m]p^*(x) + (1 - \nu)^m \sum_{x'} r_m(x') \left\{T(x', x) - \nu p^*(x) + \nu p^*(x)\right\} \\
&= [1 - (1 - \nu)^{m+1}]p^*(x) + (1 - \nu)^{m+1} \sum_{x'} r_m(x') \frac{T(x', x) - \nu p^*(x)}{1 - \nu} \\
&= [1 - (1 - \nu)^{m+1}]p^*(x) + (1 - \nu)^{m+1} r_{m+1}(x)
\end{aligned} \tag{87}$$

where we have defined

$$r_{m+1}(x) = \sum_{x'} r_m(x') \frac{T(x', x) - \nu p^*(x)}{1 - \nu}. \tag{88}$$

Clearly $\sum_x r_{m+1}(x) = 1$, and from (**??**) we have $r_{m+1}(x) \geq 0$, so that $r_{m+1}(x)$ therefore represents a probability distribution. □

This theorem easily generalizes to Markov chains over continuous state spaces.