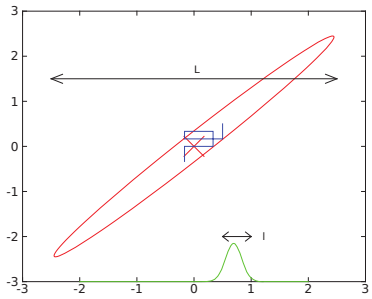


Figure 24.6 Illustration of potentially slow sampling when using Gibbs sampling for a skewed 2D Gaussian. Based on Figure 11.11 of (Bishop 2006b). Figure generated by `gibbsGaussDemo`.

will move very slowly through the state space. In particular, the size of the moves is controlled by the variance of the conditional distributions. If this is ℓ in the x_1 direction, and the support of the distribution is L along this dimension, then we need $O((L/\ell)^2)$ steps to obtain an independent sample.

In some cases we can efficiently sample groups of variables at a time. This is called **blocking Gibbs sampling** or **blocked Gibbs sampling** (Jensen et al. 1995; Wilkinson and Yeung 2002), and can make much bigger moves through the state space.

24.3 Metropolis Hastings algorithm

Although Gibbs sampling is simple, it is somewhat restricted in the set of models to which it can be applied. For example, it is not much help in computing $p(\mathbf{w}|\mathcal{D})$ for a logistic regression model, since the corresponding graphical model has no useful Markov structure. In addition, Gibbs sampling can be quite slow, as we mentioned above.

Fortunately, there is a more general algorithm that can be used, known as the **Metropolis Hastings** or **MH** algorithm, which we describe below.

24.3.1 Basic idea

The basic idea in MH is that at each step, we propose to move from the current state $\mathbf{x}$ to a new state $\mathbf{x}'$ with probability $q(\mathbf{x}'|\mathbf{x})$, where q is called the **proposal distribution** (also called the **kernel**). The user is free to use any kind of proposal they want, subject to some conditions which we explain below. This makes MH quite a flexible method. A commonly used proposal is a symmetric Gaussian distribution centered on the current state, $q(\mathbf{x}'|\mathbf{x}) = \mathcal{N}(\mathbf{x}'|\mathbf{x}, \boldsymbol{\Sigma})$; this is called a **random walk Metropolis algorithm**. We discuss how to choose $\boldsymbol{\Sigma}$ in Section 24.3.3. If we use a proposal of the form $q(\mathbf{x}'|\mathbf{x}) = q(\mathbf{x}')$, where the new state is independent of the old state, we get a method known as the **independence sampler**, which is similar to importance sampling (Section 23.4).

Having proposed a move to $\mathbf{x}'$, we then decide whether to **accept** this proposal or not according to some formula, which ensures that the fraction of time spent in each state is proportional to $p^*(\mathbf{x})$. If the proposal is accepted, the new state is $\mathbf{x}'$, otherwise the new state

is the same as the current state, $\mathbf{x}$ (i.e., we repeat the sample).

If the proposal is symmetric, so $q(\mathbf{x}'|\mathbf{x}) = q(\mathbf{x}|\mathbf{x}')$, the acceptance probability is given by the following formula:

$$r = \min(1, \frac{p^*(\mathbf{x}')}{p^*(\mathbf{x})}) \tag{24.45}$$

We see that if $\mathbf{x}'$ is more probable than $\mathbf{x}$, we definitely move there (since $\frac{p^*(\mathbf{x}')}{p^*(\mathbf{x})} > 1$), but if $\mathbf{x}'$ is less probable, we may still move there anyway, depending on the relative probabilities. So instead of greedily moving to only more probable states, we occasionally allow "downhill" moves to less probable states. In Section 24.3.6, we prove that this procedure ensures that the fraction of time we spend in each state $\mathbf{x}$ is proportional to $p^*(\mathbf{x})$.

If the proposal is asymmetric, so $q(\mathbf{x}'|\mathbf{x}) \neq q(\mathbf{x}|\mathbf{x}')$, we need the **Hastings correction**, given by the following:

$$r = \min(1, \alpha) \tag{24.46}$$

$$\alpha = \frac{p^*(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}{p^*(\mathbf{x})q(\mathbf{x}'|\mathbf{x})} = \frac{p^*(\mathbf{x}')/q(\mathbf{x}'|\mathbf{x})}{p^*(\mathbf{x})/q(\mathbf{x}|\mathbf{x}')} \tag{24.47}$$

This correction is needed to compensate for the fact that the proposal distribution itself (rather than just the target distribution) might favor certain states.

An important reason why MH is a useful algorithm is that, when evaluating α, we only need to know the target density up to a normalization constant. In particular, suppose $p^*(\mathbf{x}) = \frac{1}{Z}\tilde{p}(\mathbf{x})$, where $\tilde{p}(\mathbf{x})$ is an unnormalized distribution and Z is the normalization constant. Then

$$\alpha = \frac{(\tilde{p}(\mathbf{x}')/Z)\ q(\mathbf{x}|\mathbf{x}')}{(\tilde{p}(\mathbf{x})/Z)\ q(\mathbf{x}'|\mathbf{x})} \tag{24.48}$$

so the Z's cancel. Hence we can sample from p^* even if Z is unknown. In particular, all we have to do is evaluate $\tilde{p}$ pointwise, where $\tilde{p}(\mathbf{x}) = p^*(\mathbf{x})Z$.

The overall algorithm is summarized in Algorithm 2.

24.3.2 Gibbs sampling is a special case of MH

It turns out that Gibbs sampling, which we discussed in Section 24.2, is a special case of MH. In particular, it is equivalent to using MH with a sequence of proposals of the form

$$q(\mathbf{x}'|\mathbf{x}) = p(x_i'|\mathbf{x}_{-i})\mathbb{I}(\mathbf{x}'_{-i} = \mathbf{x}_{-i}) \tag{24.49}$$

That is, we move to a new state where x_i is sampled from its full conditional, but $\mathbf{x}_{-i}$ is left unchanged.

We now prove that the acceptance rate of each such proposal is 1, so the overall algorithm also has an acceptance rate of 100%. We have

$$\alpha = \frac{p(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}{p(\mathbf{x})q(\mathbf{x}'|\mathbf{x})} = \frac{p(x_i'|\mathbf{x}'_{-i})p(\mathbf{x}'_{-i})p(x_i|\mathbf{x}'_{-i})}{p(x_i|\mathbf{x}_{-i})p(\mathbf{x}_{-i})p(x_i'|\mathbf{x}_{-i})} \tag{24.50}$$

$$= \frac{p(x_i'|\mathbf{x}_{-i})p(\mathbf{x}_{-i})p(x_i|\mathbf{x}_{-i})}{p(x_i|\mathbf{x}_{-i})p(\mathbf{x}_{-i})p(x_i'|\mathbf{x}_{-i})} = 1 \tag{24.51}$$

Algorithm 24.2: Metropolis Hastings algorithm

Initialize x^0 ;
for $s = 0, 1, 2, \ldots$ **do**
  Define $x = x^s$;
  Sample $x' \sim q(x'|x)$;
  Compute acceptance probability

$$\alpha = \frac{\tilde{p}(x')q(x|x')}{\tilde{p}(x)q(x'|x)}$$

  Compute $r = \min(1, \alpha)$;
  Sample $u \sim U(0, 1)$;
  Set new sample to

$$x^{s+1} = \begin{cases} x' & \text{if } u < r \\ x^s & \text{if } u \geq r \end{cases}$$

where we exploited the fact that $\mathbf{x}'_{-i} = \mathbf{x}_{-i}$, and that $q(\mathbf{x}'|\mathbf{x}) = p(x'_i|\mathbf{x}_{-i})$.

The fact that the acceptance rate is 100% does not necessarily mean that Gibbs will converge rapidly, since it only updates one coordinate at a time (see Section 24.2.8). Fortunately, there are many other kinds of proposals we can use, as we discuss below.

24.3.3 Proposal distributions

For a given target distribution p^*, a proposal distribution q is valid or admissible if it gives a non-zero probability of moving to the states that have non-zero probability in the target. Formally, we can write this as

$$\text{supp}(p^*) \subseteq \cup_x \text{supp}(q(\cdot|x)) \tag{24.52}$$

For example, a Gaussian random walk proposal has non-zero probability density on the entire state space, and hence is a valid proposal for any continuous state space.

Of course, in practice, it is important that the proposal spread its probability mass in just the right way. Figure 24.7 shows an example where we use MH to sample from a mixture of two 1D Gaussians using a random walk proposal, $q(x'|x) = \mathcal{N}(x'|x, v)$. This is a somewhat tricky target distribution, since it consists of two well separated modes. It is very important to set the variance of the proposal v correctly: If the variance is too low, the chain will only explore one of the modes, as shown in Figure 24.7(a), but if the variance is too large, most of the moves will be rejected, and the chain will be very **sticky**, i.e., it will stay in the same state for a long time. This is evident from the long stretches of repeated values in Figure 24.7(b). If we set the proposal's variance just right, we get the trace in Figure 24.7(c), where the samples clearly explore the support of the target distribution. We discuss how to tune the proposal below.

One big advantage of Gibbs sampling is that one does not need to choose the proposal

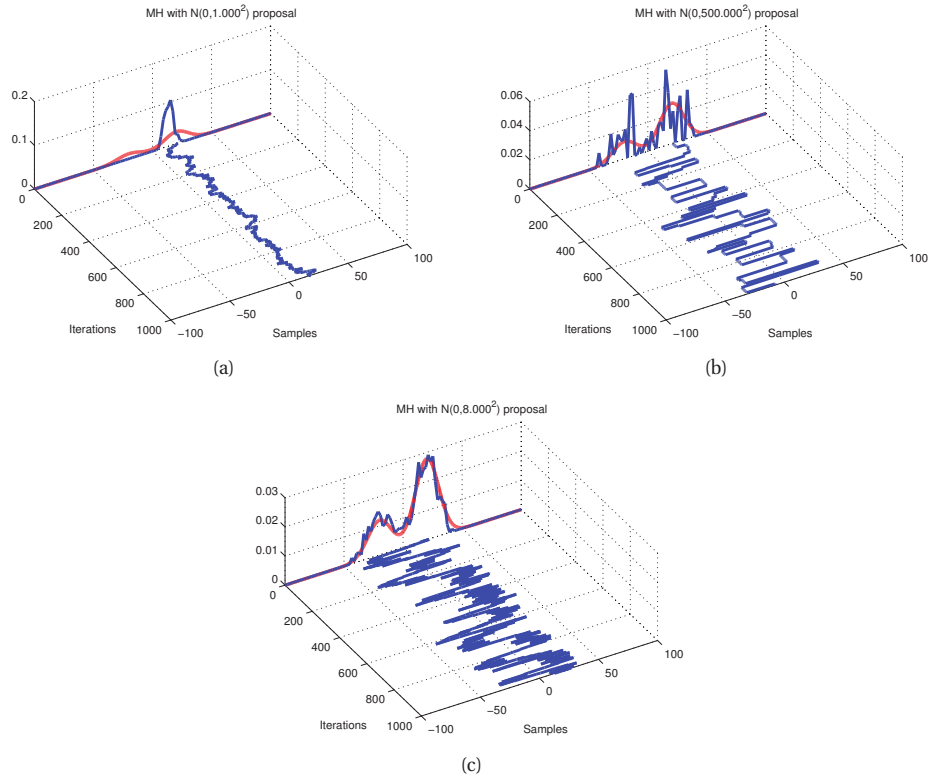
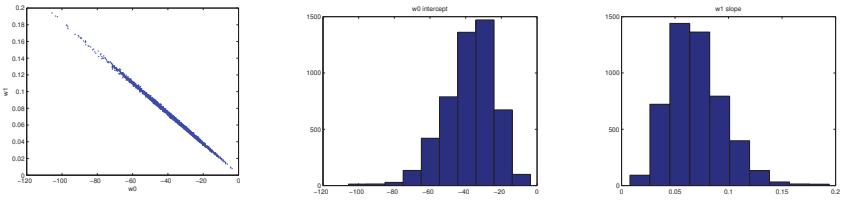


Figure 24.7 An example of the Metropolis Hastings algorithm for sampling from a mixture of two 1D Gaussians ($\boldsymbol{\mu} = (-20, 20)$, $\boldsymbol{\pi} = (0.3, 0.7)$, $\boldsymbol{\sigma} = (100, 100)$), using a Gaussian proposal with variances of $v \in \{1, 500, 8\}$. (a) When $v = 1$, the chain gets trapped near the starting state and fails to sample from the mode at $\mu = -20$. (b) When $v = 500$, the chain is very "sticky", so its effective sample size is low (as reflected by the rough histogram approximation at the end). (c) Using a variance of $v = 8$ is just right and leads to a good approximation of the true distribution (shown in red). Figure generated by `mcmcGmmDemo`. Based on code by Christophe Andrieu and Nando de Freitas.

distribution, and furthermore, the acceptance rate is 100%. Of course, a 100% acceptance can trivially be achieved by using a proposal with variance 0 (assuming we start at a mode), but this is obviously not exploring the posterior. So having a high acceptance is not the ultimate goal. We can increase the amount of exploration by increasing the variance of the Gaussian kernel. Often one experiments with different parameters until the acceptance rate is between 25% and 40%, which theory suggests is optimal, at least for Gaussian target distributions. These short initial runs, used to tune the proposal, are called **pilot runs**.



(a) (b) (c)

Figure 24.8 (a) Joint posterior of the parameters for 1d logistic regression when applied to some SAT data. (b) Marginal for the offset w_0. (c) Marginal for the slope w_1. We see that the marginals do not capture the fact that the parameters are highly correlated. Figure generated by `logregSatMhDemo`.

24.3.3.1 Gaussian proposals

If we have a continuous state space, the Hessian $\mathbf{H}$ at a local mode $\hat{\mathbf{w}}$ can be used to define the covariance of a Gaussian proposal distribution. This approach has the advantage that the Hessian models the local curvature and length scales of each dimension; this approach therefore avoids some of the slow mixing behavior of Gibbs sampling shown in Figure 24.6.

There are two obvious approaches: (1) an independence proposal, $q(\mathbf{w}'|\mathbf{w}) = \mathcal{N}(\mathbf{w}'|\hat{\mathbf{w}}, \mathbf{H}^{-1})$ or (2), a random walk proposal, $q(\mathbf{w}'|\mathbf{w}) = \mathcal{N}(\mathbf{w}'|\mathbf{w}, s^2\mathbf{H}^{-1})$, where s^2 is a scale factor chosen to facilitate rapid mixing. (Roberts and Rosenthal 2001) prove that, if the posterior is Gaussian, the asymptotically optimal value is to use $s^2 = 2.38^2/D$, where D is the dimensionality of $\mathbf{w}$; this results in an acceptance rate of 0.234.

For example, consider MH for binary logistic regression. From Equation 8.7, we have that the Hessian of the log-likelihood is $\mathbf{H}_l = \mathbf{X}^T\mathbf{D}\mathbf{X}$, where $\mathbf{D} = \text{diag}(\mu_i(1-\mu_i))$ and $\mu_i = \text{sigm}(\hat{\mathbf{w}}^T\mathbf{x}_i)$. If we assume a Gaussian prior, $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{V}_0)$, we have $\mathbf{H} = \mathbf{V}_0^{-1} + \mathbf{H}_l$, so the asymptotically optimal Gaussian proposal has the form

$$q(\mathbf{w}'|\mathbf{w}) = \mathcal{N}\left(\mathbf{w}, \frac{2.38^2}{D}\left(\mathbf{V}_0^{-1} + \mathbf{X}^T\mathbf{D}\mathbf{X}\right)^{-1}\right) \tag{24.53}$$

See (Gamerman 1997; Rossi et al. 2006; Fruhwirth-Schnatter and Fruhwirth 2010) for further details. The approach is illustrated in Figure 24.8, where we sample parameters from a 1d logistic regression model fit to some SAT data. We initialize the chain at the mode, computed using IRLS, and then use the above random walk Metropolis sampler.

If you cannot afford to compute the mode or its Hessian $\mathbf{X}\mathbf{D}\mathbf{X}$, an alternative approach, suggested in (Scott 2009), is to approximate the above proposal as follows:

$$q(\mathbf{w}'|\mathbf{w}) = \mathcal{N}\left(\mathbf{w}, \left(\mathbf{V}_0^{-1} + \frac{6}{\pi^2}\mathbf{X}^T\mathbf{X}\right)^{-1}\right) \tag{24.54}$$

24.3.3.2 Mixture proposals

If one doesn't know what kind of proposal to use, one can try a **mixture proposal**, which is a convex combination of base proposals:

$$q(\mathbf{x}'|\mathbf{x}) = \sum_{k=1}^{K} w_k q_k(\mathbf{x}'|\mathbf{x}) \tag{24.55}$$

where w_k are the mixing weights. As long as each q_k is individually valid, the overall proposal will also be valid.

24.3.3.3 Data-driven MCMC

The most efficient proposals depend not just on the previous hidden state, but also the visible data, i.e., they have the form $q(\mathbf{x}'|\mathbf{x}, \mathcal{D})$. This is called **data-driven MCMC** (see e.g., (Tu and Zhu 2002)). To create such proposals, one can sample $(\mathbf{x}, \mathcal{D})$ pairs from the forwards model and then train a discriminative classifier to predict $p(\mathbf{x}|f(\mathcal{D}))$, where $f(\mathcal{D})$ are some features extracted from the visible data.

Typically $\mathbf{x}$ is a high-dimensional vector (e.g., position and orientation of all the limbs of a person in a visual object detector), so it is hard to predict the entire state vector, $p(\mathbf{x}|f(\mathcal{D}))$. Instead we might train a discriminative detector to predict parts of the state-space, $p(x_k|f_k(\mathcal{D}))$, such as the location of just the face of a person. We can then use a proposal of the form

$$q(\mathbf{x}'|\mathbf{x}, \mathcal{D}) = \pi_0 q_0(\mathbf{x}'|\mathbf{x}) + \sum_k \pi_k q_k(x'_k|f_k(\mathcal{D})) \tag{24.56}$$

where q_0 is a standard data-independent proposal (e.g., random walk), and q_k updates the k'th component of the state space. For added efficiency, the discriminative proposals should suggest joint changes to multiple variables, but this is often hard to do.

The overall procedure is a form of **generate and test**: the discriminative proposals $q(\mathbf{x}'|\mathbf{x})$ generate new hypotheses, which are then "tested" by computing the posterior ratio $\frac{p(\mathbf{x}'|\mathcal{D})}{p(\mathbf{x}|\mathcal{D})}$, to see if the new hypothesis is better or worse. By adding an annealing step, one can modify the algorithm to find posterior modes; this is called **simulated annealing**, and is described in Section 24.6.1. One advantage of using the mode-seeking version of the algorithm is that we do not need to ensure the proposal distribution is reversible.

24.3.4 Adaptive MCMC

One can change the parameters of the proposal as the algorithm is running to increase efficiency. This is called **adaptive MCMC**. This allows one to start with a broad covariance (say), allowing large moves through the space until a mode is found, followed by a narrowing of the covariance to ensure careful exploration of the region around the mode.

However, one must be careful not to violate the Markov property; thus the parameters of the proposal should not depend on the entire history of the chain. It turns out that a sufficient condition to ensure this is that the adaption is "faded out" gradually over time. See e.g., (Andrieu and Thoms 2008) for details.

24.3.5 Initialization and mode hopping

It is necessary to start MCMC in an initial state that has non-zero probability. If the model has deterministic constraints, finding such a legal configuration may be a hard problem in itself. It is therefore common to initialize MCMC methods at a local mode, found using an optimizer.

In some domains (especially with discrete state spaces), it is a more effective use of computation time to perform multiple restarts of an optimizer, and to average over these modes, rather than exploring similar points around a local mode. However, in continuous state spaces, the mode contains negligible volume (Section 5.2.1.3), so it is necessary to locally explore around each mode, in order to visit enough posterior probability mass.

24.3.6 Why MH works *

To prove that the MH procedure generates samples from p^*, we have to use a bit of Markov chain theory, so be sure to read Section 17.2.3 first.

The MH algorithm defines a Markov chain with the following transition matrix:

$$p(\mathbf{x}'|\mathbf{x}) = \begin{cases} q(\mathbf{x}'|\mathbf{x})r(\mathbf{x}'|\mathbf{x}) & \text{if } \mathbf{x}' \neq \mathbf{x} \\ q(\mathbf{x}|\mathbf{x}) + \sum_{\mathbf{x}' \neq \mathbf{x}} q(\mathbf{x}'|\mathbf{x})(1 - r(\mathbf{x}'|\mathbf{x})) & \text{otherwise} \end{cases} \tag{24.57}$$

This follows from a case analysis: if you move to $\mathbf{x}'$ from $\mathbf{x}$, you must have proposed it (with probability $q(\mathbf{x}'|\mathbf{x})$) and it must have been accepted (with probability $r(\mathbf{x}'|\mathbf{x})$); otherwise you stay in state $\mathbf{x}$, either because that is what you proposed (with probability $q(\mathbf{x}|\mathbf{x})$), or because you proposed something else (with probability $q(\mathbf{x}'|\mathbf{x})$) but it was rejected (with probability $1 - r(\mathbf{x}'|\mathbf{x})$).

Let us analyse this Markov chain. Recall from Section 17.2.3.4 that a chain satisfies **detailed balance** if

$$p(\mathbf{x}'|\mathbf{x})p^*(\mathbf{x}) = p(\mathbf{x}|\mathbf{x}')p^*(\mathbf{x}') \tag{24.58}$$

We also showed that if a chain satisfies detailed balance, then p^* is its stationary distribution. Our goal is to show that the MH algorithm defines a transition function that satisfies detailed balance and hence that p^* is its stationary distribution. (If Equation 24.58 holds, we say that p^* is an **invariant** distribution wrt the Markov transition kernel q.)

Theorem 24.3.1. *If the transition matrix defined by the MH algorithm (given by Equation 24.57) is ergodic and irreducible, then p^* is its unique limiting distribution.*

Proof. Consider two states $\mathbf{x}$ and $\mathbf{x}'$. Either

$$p^*(\mathbf{x})q(\mathbf{x}'|\mathbf{x}) < p^*(\mathbf{x}')q(\mathbf{x}|\mathbf{x}') \tag{24.59}$$

or

$$p^*(\mathbf{x})q(\mathbf{x}'|\mathbf{x}) > p^*(\mathbf{x}')q(\mathbf{x}|\mathbf{x}') \tag{24.60}$$

We will ignore ties (which occur with probability zero for continuous distributions). Without loss of generality, assume that $p^*(\mathbf{x})q(\mathbf{x}'|\mathbf{x}) > p^*(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')$. Hence

$$\alpha(\mathbf{x}'|\mathbf{x}) = \frac{p^*(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}{p^*(\mathbf{x})q(\mathbf{x}'|\mathbf{x})} < 1 \tag{24.61}$$

Hence we have $r(\mathbf{x}'|\mathbf{x}) = \alpha(\mathbf{x}'|\mathbf{x})$ and $r(\mathbf{x}|\mathbf{x}') = 1$.

Now to move from $\mathbf{x}$ to $\mathbf{x}'$ we must first propose $\mathbf{x}'$ and then accept it. Hence

$$p(\mathbf{x}'|\mathbf{x}) = q(\mathbf{x}'|\mathbf{x})r(\mathbf{x}'|\mathbf{x}) = q(\mathbf{x}'|\mathbf{x})\frac{p^*(\mathbf{x}')q(\mathbf{x}|\mathbf{x}')}{p^*(\mathbf{x})q(\mathbf{x}'|\mathbf{x})} = \frac{p^*(\mathbf{x}')}{p^*(\mathbf{x})}q(\mathbf{x}|\mathbf{x}') \tag{24.62}$$

Hence

$$p^*(\mathbf{x})p(\mathbf{x}'|\mathbf{x}) = p^*(\mathbf{x}')q(\mathbf{x}|\mathbf{x}') \tag{24.63}$$

The backwards probability is

$$p(\mathbf{x}|\mathbf{x}') = q(\mathbf{x}|\mathbf{x}')r(\mathbf{x}|\mathbf{x}') = q(\mathbf{x}|\mathbf{x}') \tag{24.64}$$

since $r(\mathbf{x}|\mathbf{x}') = 1$. Inserting this into Equation 24.63 we get

$$p^*(\mathbf{x})p(\mathbf{x}'|\mathbf{x}) = p^*(\mathbf{x}')p(\mathbf{x}|\mathbf{x}') \tag{24.65}$$

so detailed balance holds wrt p^*. Hence, from Theorem 17.2.3, p^* is a stationary distribution. Furthermore, from Theorem 17.2.2, this distribution is unique, since the chain is ergodic and irreducible. □

24.3.7 Reversible jump (trans-dimensional) MCMC *

Suppose we have a set of models with different numbers of parameters, e.g., mixture models in which the number of mixture components is unknown. Let the model be denoted by m, and let its unknowns (e.g., parameters) be denoted by $\mathbf{x}_m \in \mathcal{X}_m$ (e.g., $\mathcal{X}_m = \mathbb{R}^{n_m}$, where n_m is the dimensionality of model m). Sampling in spaces of differing dimensionality is called **trans-dimensional MCMC** (Green 2003). We could sample the model indicator $m \in \{1, \ldots, M\}$ and sample all the parameters from the product space $\prod_{m=1}^{M} \mathcal{X}_m$, but this is very inefficient. It is more parsimonious to sample in the union space $\mathcal{X} = \cup_{m=1}^{M}\{m\} \times \mathcal{X}_m$, where we only worry about parameters for the currently active model.

The difficulty with this approach arises when we move between models of different dimensionality. The trouble is that when we compute the MH acceptance ratio, we are comparing densities defined in different dimensionality spaces, which is meaningless. It is like trying to compare a sphere with a circle. The solution, proposed by (Green 1998) and known as **reversible jump MCMC** or **RJMCMC**, is to augment the low dimensional space with extra random variables so that the two spaces have a common measure.

Unfortunately, we do not have space to go into details here. Suffice it to say that the method can be made to work in theory, although it is a bit tricky in practice. If, however, the continuous parameters can be integrated out (resulting in a method called collapsed RJMCMC), much of the difficulty goes away, since we are just left with a discrete state space, where there is no need to worry about change of measure. For example, (Denison et al. 2002) includes many examples of applications of collapsed RJMCMC applied to Bayesian inference fro adaptive basis-function models. They sample basis functions from a fixed set of candidates (e.g., centered on the data points), and integrate out the other parameters analytically. This provides a Bayesian alternative to using RVMs or SVMs.

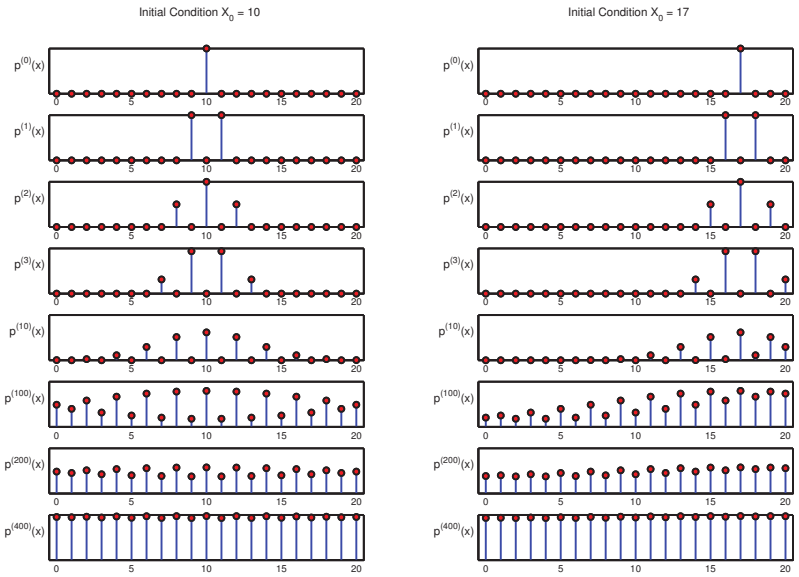


Figure 24.9 Illustration of convergence to the uniform distribution over $\{0, 1, \ldots, 20\}$ using a symmetric random walk starting from (left) state 10, and (right) state 17. Based on Figures 29.14 and 29.15 of (MacKay 2003). Figure generated by `randomWalk0to20Demo`.

24.4 Speed and accuracy of MCMC

In this section, we discuss a number of important theoretical and practical issues to do with MCMC.

24.4.1 The burn-in phase

We start MCMC from an arbitrary initial state. As we explained in Section 17.2.3, only when the chain has "forgotten" where it started from will the samples be coming from the chain's stationary distribution. Samples collected before the chain has reached its stationary distribution do not come from p^*, and are usually thrown away. The initial period, whose samples will be ignored, is called the **burn-in phase**.

For example, consider a uniform distribution on the integers $\{0, 1, \ldots, 20\}$. Suppose we sample from this using a symmetric random walk. In Figure 24.9, we show two runs of the algorithm. On the left, we start in state 10; on the right, we start in state 17. Even in this small problem it takes over 100 steps until the chain has "forgotten" where it started from.

It is difficult to diagnose when the chain has burned in, an issue we discuss in more detail below. (This is one of the fundamental weaknesses of MCMC.) As an interesting example of what can happen if you start collecting samples too early, consider the Potts model. Figure 24.10(a), shows a sample after 500 iterations of Gibbs sampling. This suggests that the model likes