do it. In other words, they need to hit the double 20 – what is the probability that they will succeed?

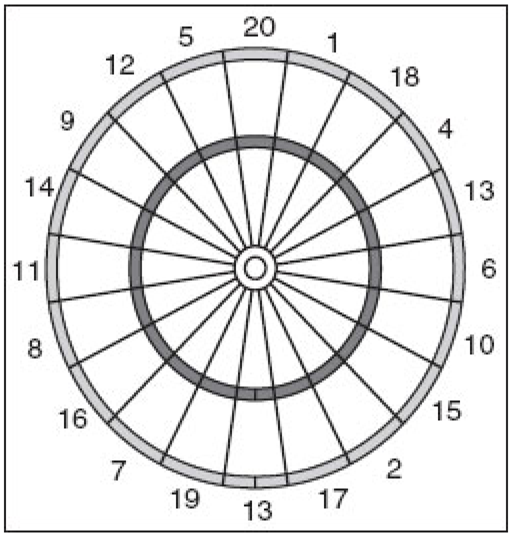


FIGURE 4.8 A dartboard.

Assume that there exists some probability density function defined over the position that the dart will land. In other words, when a player is aiming for, say, double twenty, the position at which the dart will land could be considered as an instance of some random variable. We will use the vector **y** to describe this position and therefore the density will look something like $p(\mathbf{y}|\Delta)$. Δ ought to depend (at least to some extent) on where the player is aiming. The extent to which this dependence exists depends on the player's skill. For a professional trying to hit double twenty, we might expect the density to be tightly concentrated around the double twenty area. For a poor player, aiming might make very little difference to where the dart ends up. So, Δ depends on the skill of the player and the strength of their technique – making $p(\mathbf{y}|\Delta)$ very hard to define.

At this point it would be easy to give up. But, taking a step backwards, we are not directly interested in $p(\mathbf{y}|\Delta)$, just the probability that the player throws a double 20. Do we need to be able to write down an analytic expression for $p(\mathbf{y}|\Delta)$ to work this out? Before we answer this, let us satisfy ourselves that we could work it out if we could write down $p(\mathbf{y}|\Delta)$. Define the random variable $T = f(\mathbf{y})$ where $f(\mathbf{y})$ is 1 if **y** is inside the double twenty region and zero otherwise. T depends on **y** and hence depends on Δ. So, we're interested in the following probability: $P(T = 1|\Delta)$. This is nothing more than an expectation. In particular, it looks rather like the expectations we had to compute for the binary response model in the previous section:

$$P(T = 1|\Delta) = \mathbf{E}_{p(\mathbf{y}|\Delta)}\{f(\mathbf{y})\} = \int f(\mathbf{y})p(\mathbf{y}|\Delta)\, d\mathbf{y}. \tag{4.16}$$

In theory, if we could write down $p(\mathbf{y}|\Delta)$, we could work this out. However, we have also seen that we can compute quantities like this with a sample-based approximation. In particular, if $\mathbf{y}_s$ is the sth of N_s samples from $p(\mathbf{y}|\Delta)$, our approximation would look like:

$$P(T = 1|\Delta) \simeq \frac{1}{N_s}\sum_{s=1}^{N_s} f(\mathbf{y}_s).$$

So, we do not need to be able to write down $p(\mathbf{y}|\Delta)$ to be able to compute $P(T = 1|\Delta)$ as long as we can sample from it. Fortunately, sampling from $p(\mathbf{y}|\Delta)$ is pretty easy – we get our player, some darts and a board and we ask the player to aim for double twenty. The position of each dart thrown is a sample from $p(\mathbf{y}|\Delta)$. If we record $\mathbf{y}_s$ for each of N_s throws, we can compute the sample-based approximation given in Equation 4.16. In fact, in this case it works out as just the proportion of times the player throws a double twenty).

We can explicitly relate this procedure to our binary response model. Firstly, the quantity of interest in the darts case, $P(T = 1|\Delta)$ is analogous to the predictive probability in the binary response model: $P(T_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}, \sigma^2)$. In both cases to compute this quantity, we must take an expectation with respect to some density: our darts distribution $p(\mathbf{y}|\Delta)$ is analogous to $p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \sigma^2)$ – the posterior density over our parameters. In the darts case, we approximated this expectation by drawing samples directly from the posterior (despite the fact that we couldn't write it down). In the binary response case, we approximated the posterior with something we could sample from and then sampled. We will now see how we can sample directly from $p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \sigma^2)$ (see Exercise 4.4).

4.5.2 The Metropolis–Hastings algorithm

In this section, we will introduce the Metropolis–Hastings[2] (MH) algorithm. Rather than go into too much detail we will introduce it as a recipe, describing the steps involved without proving why they work. References to further reading are provided at the end of the chapter.

Recall that we are attempting to sample from $p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \sigma^2)$ so that we can approximate the following expectation:

$$\begin{aligned} P(T_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}, \sigma^2) &= \mathbf{E}_{p(\mathbf{w}|\mathbf{X},\mathbf{t},\sigma^2)}\left\{P(T_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \mathbf{w})\right\} \\ &= \int P(T_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \mathbf{w})p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \sigma^2)\, d\mathbf{w}, \end{aligned}$$

with:

$$P(T = 1|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}, \sigma^2) \simeq \frac{1}{N_s}\sum_{s=1}^{N_s} P(T_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \mathbf{w}_s).$$

Metropolis–Hastings generates a sequence of samples $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{s-1}, \mathbf{w}_s, \ldots, \mathbf{w}_{N_s}$. Generating a sample (say $\mathbf{w}_s$) consists of two steps. In the first step, we need to *propose* a new sample – a candidate for $\mathbf{w}_s$. This is performed by proposing a movement from the previous sample ($\mathbf{w}_{s-1}$). Secondly, the proposed sample is tested to see whether or not it should be accepted. If accepted, it becomes our new sample $\mathbf{w}_s$. If it is not accepted, our new sample is set to be equal to the previous one, $\mathbf{w}_s = \mathbf{w}_{s-1}$. This is continued until we have collected what we believe to be enough samples.

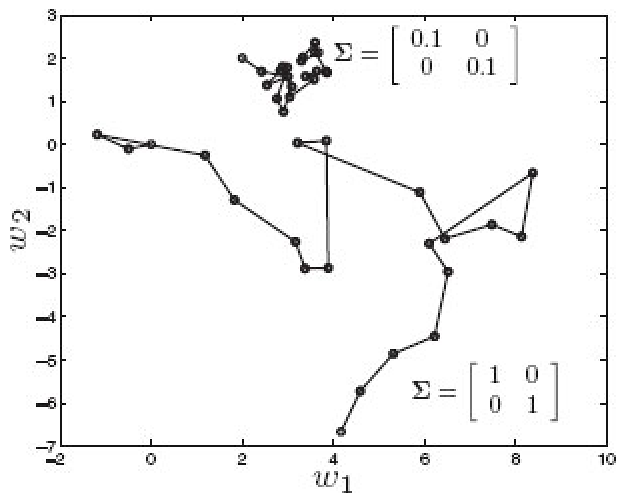


FIGURE 4.9 Two examples of random walks where the distribution over the next location is a Gaussian centred at the current location. The two walks have different covariance matrices, shown in the plot.

Now, if our proposal is based on a movement from the previous sample, what do we do for our first sample $\mathbf{w}_1$? It turns out that it doesn't matter where we start – $\mathbf{w}_1$ can be anything. As long as we sample for long enough, our sampler is guaranteed to converge to the distribution of interest. So, we can pluck a $\mathbf{w}_1$ from anywhere (sampling it from the prior would probably be a sensible choice), set the Metropolis–Hastings algorithm off, wait for it to converge to the correct distribution and then harvest as many samples as we need. A word of caution: the sampler is guaranteed to converge *in theory*. In practice, it is important to use one (or ideally more) of the methods available to test convergence before we start harvesting samples. We will now look at the proposal and acceptance steps in more detail.

Proposing a new sample

Assume that we have already sampled $s - 1$ values using the MH scheme. We will propose a sample based on a movement from $\mathbf{w}_{s-1}$. Calling our proposed sample $\widetilde{\mathbf{w}}_s$ (we can only call it $\mathbf{w}_s$ once it has been accepted), we need to define a density:

$$p(\widetilde{\mathbf{w}}_s|\mathbf{w}_{s-1}).$$

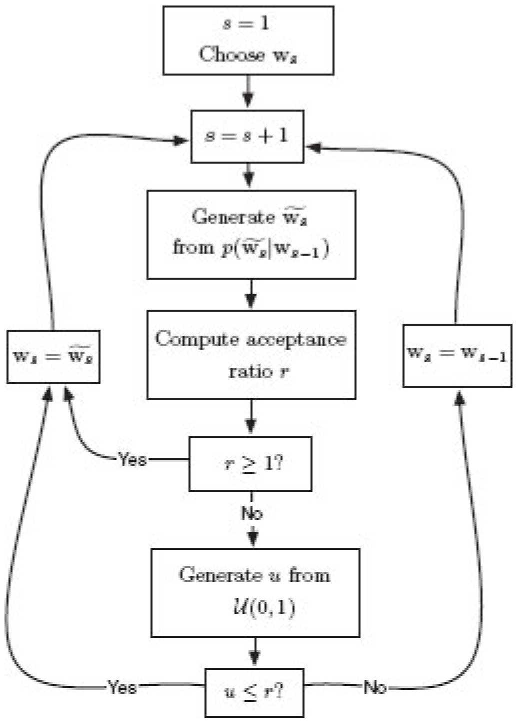
This density does not have to have any connection with the posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \sigma^2)$ from which we're trying to sample. We are free to define it as we please. In practice, the choice will have an impact on how long it will take the MH sampler to converge. A common choice is to use a Gaussian centred on the current sample, $\mathbf{w}_{s-1}$:

$$p(\widetilde{\mathbf{w}}_s|\mathbf{w}_{s-1}, \mathbf{\Sigma}) = \mathcal{N}(\mathbf{w}_{s-1}, \mathbf{\Sigma}).$$

Sampling a sequence of values like this creates what is known as a **random walk**. In Figure 4.9 we show two such walks (MATLAB script: **randwalks.m**). One starts from $\mathbf{w}_1 = [0, 0]^\mathsf{T}$ and has covariance $\mathbf{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ whilst the other starts from $\mathbf{w}_1 = [2, 2]^\mathsf{T}$ and has covariance $\mathbf{\Sigma} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$. The latter walk moves less distance in each step due to the smaller diagonal (variance) elements in the covariance matrix. As we have already mentioned, the Gaussian is a popular choice for the proposal density. One reason is the ease with which we can sample from it – choosing a proposal distribution that was hard to sample from would make things unnecessarily complicated. Another reason is that it is symmetric: moving to $\widetilde{\mathbf{w}}_s$ from $\mathbf{w}_{s-1}$ is just as likely as moving from $\widetilde{\mathbf{w}}_s$ to $\mathbf{w}_{s-1}$:

$$p(\widetilde{\mathbf{w}_s}|\mathbf{w}_{s-1},\Sigma) = p(\mathbf{w}_{s-1}|\widetilde{w_s},\Sigma).$$

We will see the advantage of this as we move on to the acceptance step.



$s = 1$
Choose $\mathbf{w}_s$
$s = s + 1$
Generate $\widetilde{\mathbf{w}_s}$ from $p(\widetilde{\mathbf{w}_s}|\mathbf{w}_{s-1})$
Compute acceptance ratio r
$\mathbf{w}_s = \widetilde{\mathbf{w}_s}$
$\mathbf{w}_s = \mathbf{w}_{s-1}$
$r \geq 1$?
Yes
No
Generate u from $\mathcal{U}(0,1)$
$u \leq r$?
Yes
No

FIGURE 4.10 The Metropolis–Hastings algorithm.

Accepting or rejecting We now have $\widetilde{\mathbf{w}_s}$, a candidate for $\mathbf{w}_S$. We must now decide whether we should accept it or reject it. To do this, we compute the following ratio:

$$r = \frac{p(\widetilde{w_s}|\mathbf{X},\mathbf{t},\sigma^2)}{p(w_{s-1}|\mathbf{X},\mathbf{t},\sigma^2)} \frac{p(w_{s-1}|\widetilde{w_s},\Sigma)}{p(\widetilde{w_s}|w_{s-1},\Sigma)}. \tag{4.17}$$

This is the ratio of the posterior density at the *proposed* sample to that at the old sample multiplied by the ratio of the proposal densities. The symmetry of the Gaussian proposal distribution discussed above allows us to ignore this last term, as it is always equal to 1. The first term is the ratio of posterior densities evaluated at the two different parameter values. We cannot compute the densities exactly because we cannot normalise them. However, because we are interested in a ratio, the normalisation constants cancel. So, we can substitute the ratio of posteriors with the ratio of the priors multiplied by the ratio of likelihoods. This leads us to the following expression:

$$r = \frac{g(\widetilde{w_s};\mathbf{X},\mathbf{t},\sigma^2)}{g(w_{s-1};\mathbf{X},\mathbf{t},\sigma^2)} = \frac{p(\widetilde{w_s}|\sigma^2)}{p(\mathbf{w}_{s-1}|\sigma^2)} \frac{p(\mathbf{t}|\widetilde{w_s},\mathbf{X})}{p(\mathbf{t}|w_{s-1},\mathbf{X})}.$$

This ratio will always be positive as the density functions are always positive. If it is one or greater, we accept the sample ($\mathbf{w}_S = \widetilde{\mathbf{w}_s}$). If r is less than 1, we accept the sample with probability equal to r. In other words, if we propose a set of parameters that corresponds to a higher value of the posterior density than $\mathbf{w}_{S-1}$, we always accept it ($r > 1$). If we propose a set that corresponds to a lower value of posterior density, we accept it sometimes, but not always. The algorithm is depicted in Figure 4.10. Notice that we have described the accept/reject step in more detail. If $r < 1$, we should accept with probability r. This is achieved by drawing a value (u) from a uniform distribution between 0 and 1. Because it is uniform, the probability that u will be less than or equal to r is equal to r. Hence, we accept the proposal if $u \leq r$ and reject otherwise. The whole process is best illustrated with an example.

Figure 4.11 shows the Metropolis–Hastings algorithm in action, sampling from an arbitrary density (indicated by contours) (MATLAB script: **mhexample.m**). The starting point, $\mathbf{w}_1$ is shown in Figure 4.11(a). Our proposal density is Gaussian with $\mathbf{\Sigma} = \mathbf{I}$. From the starting point, the first proposal made is $\widetilde{\mathbf{w}_2}$, shown in Figure 4.11(b). The proposal causes an increase in posterior density and is therefore accepted: $\mathbf{w}_2 = \widetilde{\mathbf{w}_2}$. This acceptance is indicated by the solid line in Figure 4.11(b). The next proposal $\widetilde{\mathbf{w}_3}$ causes a slight decrease in posterior density but is accepted nonetheless (remember that, if the proposal causes a decrease, there is still a probability of acceptance). This is shown by the new solid line in Figure 4.11(c). The next proposal $\widetilde{\mathbf{w}_4}$ causes a large decrease in the posterior density value. Such a proposal is highly unlikely to be accepted (the ratio is much less than 1) and in this instance it isn't. This is represented by the dashed line in Figure 4.11(c). Hence, $\mathbf{w}_4 \neq \widetilde{\mathbf{w}_4}$ and is instead set to $\mathbf{w}_4 = \mathbf{w}_3$. This process continues in Figures 4.11(d) and 4.11(e) by which time we have ten samples. Along the way, three proposals were rejected and in each of those instances the sample is set to be equal to the value of the previous (accepted) sample. Continuing this process, we can see the first 300 accepted samples in Figure 4.11(f). These samples look reasonably consistent with the density contours – samples seem to be more concentrated towards the centre of the density and very sparse around the edges.

The density we are sampling from in this example happens to be a Gaussian. So, we can go some way towards convincing ourselves that we are indeed sampling from the correct density by computing the mean and covariance of the samples and seeing if they correspond to the mean and covariance of the actual density. The actual mean and covariance are given by

$$\boldsymbol{\mu} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} 3 & 0.4 \\ 0.4 & 3 \end{bmatrix}.$$

After N_s = 10, 000 samples, we can compute the sample-based approximations to the mean and covariance ($\boldsymbol{\mu}'$, $\mathbf{S}'$) as follows:

$$\boldsymbol{\mu}' = \frac{1}{N_s}\sum_{s=1}^{N_s} w_s, \quad \mathbf{S}' = \frac{1}{N_s}\sum_{s=1}^{N_s}(w_s - \boldsymbol{\mu}')(w_s - \boldsymbol{\mu}')^{\mathrm{T}}.$$

These work out as

$$\boldsymbol{\mu}' = \begin{bmatrix} 0.9770 \\ 1.0928 \end{bmatrix}, \quad \mathbf{S}' = \begin{bmatrix} 3.0777 & 0.4405 \\ 0.4405 & 2.8983 \end{bmatrix},$$

which are both very similar to the true values.

Before we move on to applying MH to our binary response model, we need to discuss two related concepts – **burn-in** and **convergence**. As we can start our sampler from anywhere (there is no restriction on $\mathbf{w}_1$), we don't necessarily know if we are starting the sampler in an area that we should be generating samples from (it might be an area of very low posterior density). Therefore, the first few samples may not be representative and should be discarded. This period between the starting point and convergence of the sampler is known as the burn-in period. Sadly, it is not possible to conclusively determine how long this period should be. In the example described above, it is no more than a couple of samples but in some applications it could easily be hundreds or thousands. To overcome this problem we need a method for determining convergence. This is not convergence to a particular *value*, but convergence to a particular *distribution*. In other words, are the samples we are seeing coming from the correct distribution?

A popular method is to start several samplers simultaneously from different starting points. When all of the samplers are generating samples with similar characteristics (mean, variance, etc.), it suggests that they have all converged to the same distribution – the one we are trying to sample from.

We will now return to our binary response model. Using the MH scheme described above, we generate 10,000 samples from $p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \sigma^2)$ (MATLAB script: **logmh.m**). Our proposal density is a Gaussian with $\mathbf{\Sigma} = \gamma^2\mathbf{I}$ where $\gamma^2 = 0.5$. In Figure 4.12(a) we show every tenth sample (plotting all 10,000 samples makes for a very crowded plot) along with the posterior contours. The samples and the contours look reasonably coherent. If we like, we can use the samples to create marginal posterior densities for the two individual parameters. Recall from Section 2.2.6 that to marginalise w_2 from the posterior we would need to integrate (or sum, if the random variable is discrete) over all values w_2 could take:

$$p(w_1|\mathbf{X}, \mathbf{t}, \sigma^2) = \int p(w_1, w_2|\mathbf{X}, \mathbf{t}, \sigma^2)\, dw_2,$$

where $p(w_1, w_2|\mathbf{X}, \mathbf{t}, \sigma^2)$ is another way of writing $p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \sigma^2)$. To get a sample-based approximation, we take each of our samples, $\mathbf{w}_s$, and ignore w_2. In other words, if we throw away the value of w_2 from each sample, we are left with a set of samples from $p(w_1|\mathbf{X}, \mathbf{t}, \sigma^2)$. In Figures 4.12(b)–4.12(d), we show three popular ways of visualising these samples. In the first, (Figure 4.12(b)) we have split the range of possible values into 20 sections and counted the number of samples that fall in each section. The black bars show the numbers for w_1 and the grey bars for w_2. If we were to take the number of samples falling into a particular section and divide it by the total number of samples, the numbers obtained could be thought of as the posterior probabilities that the w_1 (or w_2) falls into each of these sections. In the second example (Figure 4.12(c)), we have just plotted all 10,000 samples for w_1 (a similar plot for w_2 looks almost identical). This plot gives us confidence that the sampler has converged very quickly. If it hadn't, we might see an overall increasing or decreasing trend. In Figure 4.12(d) we show two continuous density functions that have been fitted to the samples. This is, in itself, a machine learning task for which there are various possible solutions. If the samples looked like they had come from a Gaussian, we could fit Gaussian densities to the two sets of samples (see Exercise 2.8). In this example, we have used a more general technique known as kernel density estimation. This can be performed in MATLAB using the **ksdensity** function. We won't go into any more detail here – the important point is that there are many ways to visualise the samples and it is possible to turn them into (approximate) continuous density functions.

Finally we turn our attention back to the predictive probability, $P(T_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}, \sigma^2)$. When using the Laplace approximation, we approximated this quantity by drawing samples $\mathbf{w}_1, \ldots, \mathbf{w}_{N_s}$ from the approximate posterior and then computing

$$P(T_{\text{new}} = 1|\mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}, \sigma^2) = \frac{1}{N_s}\sum_{s=1}^{N_s} \frac{1}{1+\exp(-\mathbf{w}_s^{\mathsf{T}}\mathbf{x}_{\text{new}})}.$$

We now have a set of samples from the true posterior, $p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \sigma^2)$ and we can use them in exactly the same way. Figure 4.12(e) shows the predictive probability contours computed using these true posterior samples. Remember that these contours give the probability of classifying an object at any particular location as a square. The shape of the contours looks rather like the shape in Figure 4.7(b), which is not very surprising, as we saw in Figure 4.6(b) that the Laplace approximation didn't look too different from the true posterior. The only noticeable difference is that the contours in Figure 4.12(e) are slightly less tightly curved around the areas in which the data lie. This suggests that the probability reduces rather more slowly as we move away from the squares. The MH sampler is sampling from the true posterior and so the contours in Figure 4.12(e) should be considered closer to the truth than those for the Laplace approximation in Figure 4.6(a). This comparison is really just giving us an indication of how good the Laplace approximation is *at making predictions*. Figure 4.12(f) shows the decision boundaries corresponding to 20 of the MH samples picked at random (c.f. Figure 4.7(a)) (see Exercises 4.6, 4.7 and 4.8).

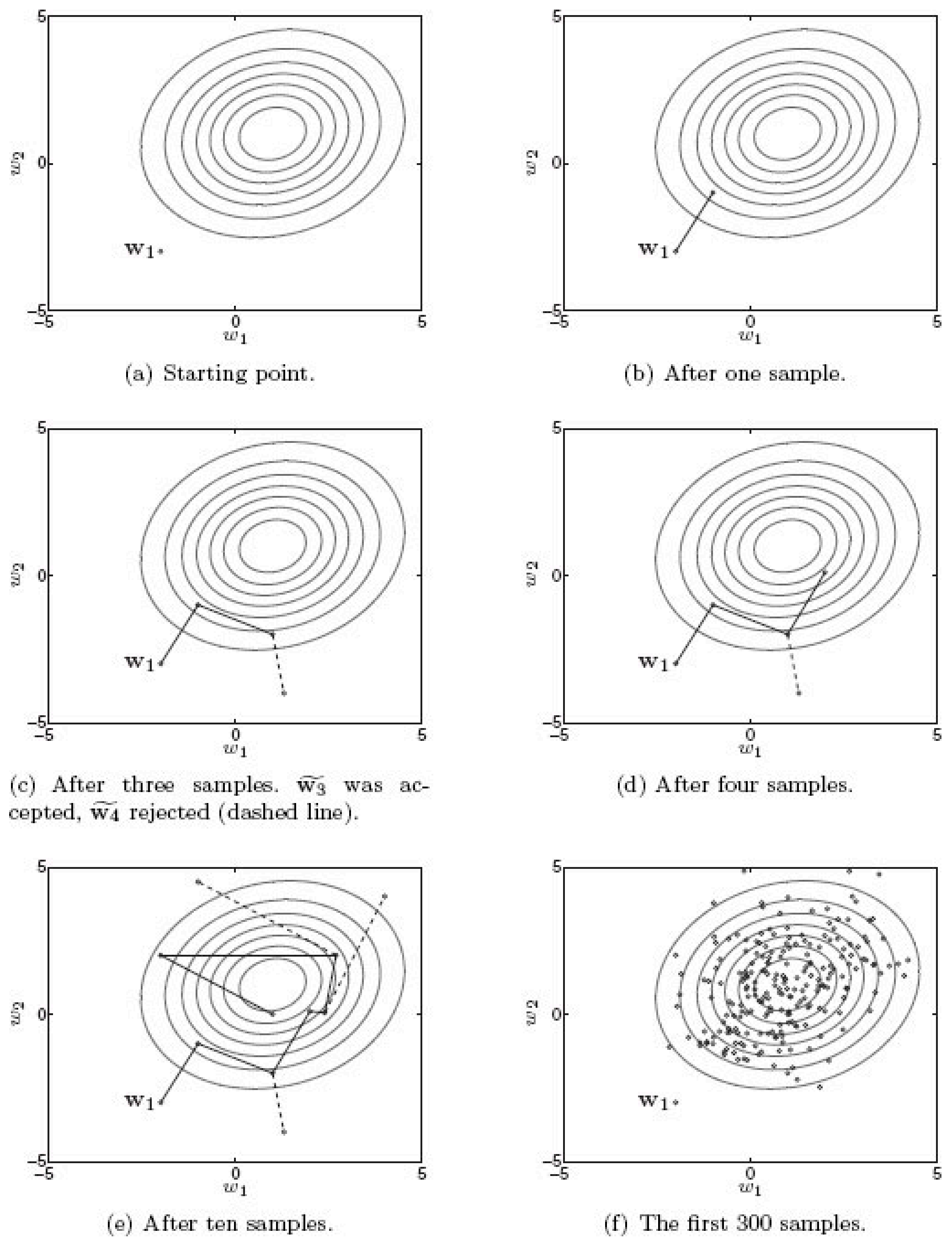
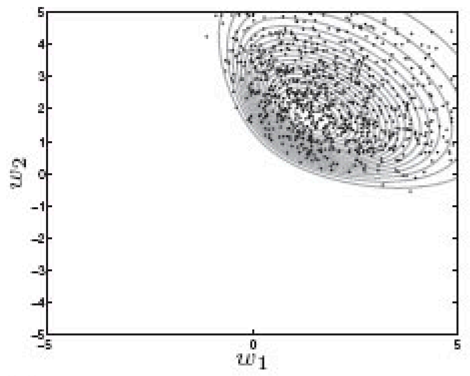
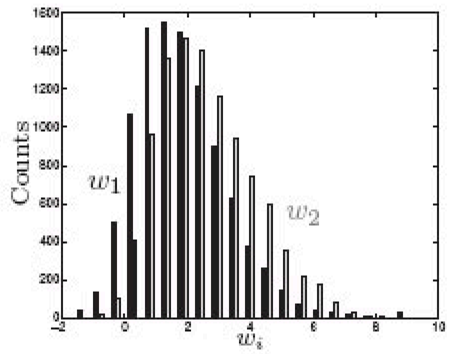


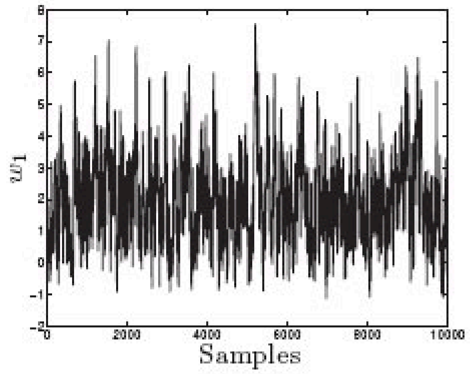
FIGURE 4.11 Example of the Metropolis–Hastings agorithm in operation.



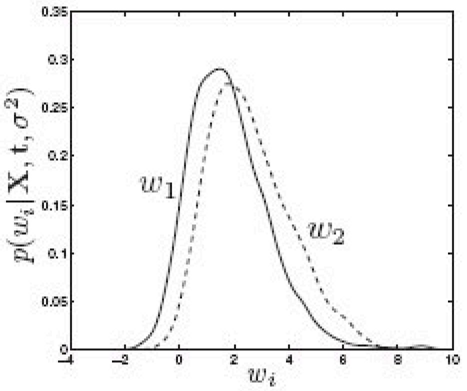
(a) One thousand of the MH samples along with the posterior contours.



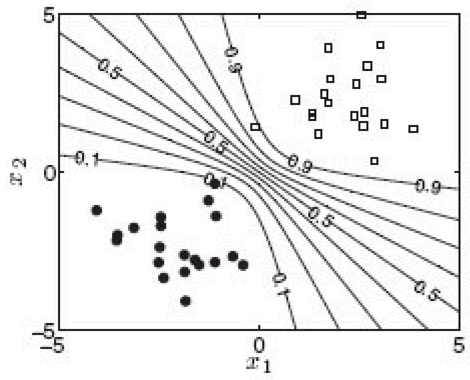
(b) Histograms of the samples for both w_1 (black) and w_2 (grey).



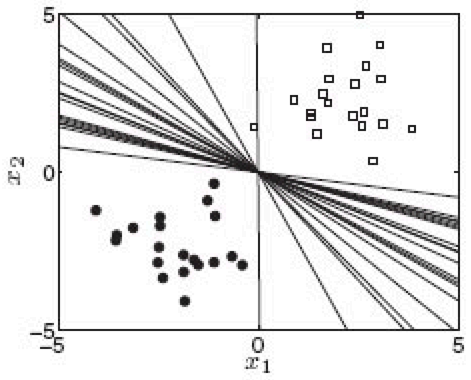
(c) All of the w_1 samples plotted against iteration, s.



(d) Continuous densities fitted to the w_1 and w_2 samples.

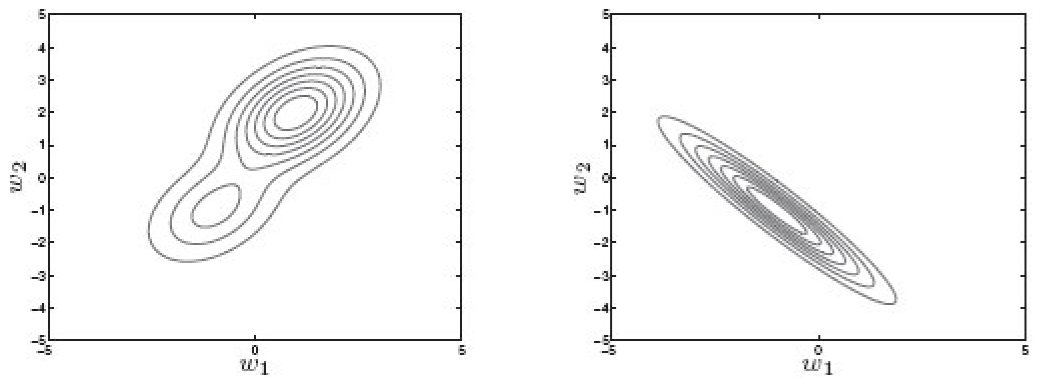


(e) Predictive probability contours. The contours show the probability of classifying an object at any location as a square. The probability of classifying an object as a circle at any point is 1 minus this value.



(f) Decision boundaries created from 20 randomly selected MH samples.

FIGURE 4.12 Results of applying the MH sampling algorithm to the binary response model.



(a) A bi-modal density.

(b) A density with high parameter correlation.

FIGURE 4.13 Two densities that would be tricky to sample from with MH.

4.5.3