Bayesian Inference

Chapter 6. Implementation of Bayesian inference

M. Concepción Ausín
Department of Statistics
Universidad Carlos III de Madrid

Master in Business Administration and Quantitative Methods
Master in Mathematical Engineering
Chapter 6. Implementation of Bayesian inference

Objective

To introduce the main numerical methods that can be used to evaluate the integrals necessary in many Bayesian problems. In particular, we concentrate on MCMC and Gibbs sampling approaches.

Recommended reading

Introduction

We have seen that numerical procedures are often needed in Bayesian inference for the computation of the posterior distribution:

$$p(\theta | x) = \frac{f(x | \theta) p(\theta)}{\int f(x | \theta) p(\theta) d\theta},$$

and for the computation of posterior moments, predictive distributions etc.

The different techniques which might be applied are as follows:

- Numerical integration
- Gaussian approximations (considered in chapter 8)
- Monte Carlo approaches:
  - direct methods
  - via Markov chains
Contents

1. Numerical integration

2. Monte Carlo approaches
   2.1. Importance sampling
   2.2. Rejection algorithm

3. MCMC methods
   3.1. Metropolis Hastings algorithm
   3.2. Slice sampling
   3.3. Gibbs sampling
   3.4. MCMC convergence assessment

4. Other algorithms
Numerical integration

Many numerical integration techniques have been developed. For fuller reviews, see for example Ausín (2007) or the Wikipedia:


Numerical integration consists of approximating an integral using a weighted sum of some values of the integrand,

\[ \int_{a}^{b} f(x) \, dx \approx \sum_{i=1}^{n} \omega_i f_i \]

where \( f_i = f(x_i) \) is the value of the function in a number of selected points, \( x_1, \ldots, x_n \), and \( \omega_1, \ldots, \omega_n \), are the weights associated to these points.
Numerical integration

Assume that we divide the interval \([a, b]\) into \(n\) equally spaced points,

\[ x_i = x_1 + ih, \quad \text{for } i = 1, \ldots, n - 1, \]

where \(x_1 = a\), \(x_n = b\) and \(h = (b - a)/n\).

One of the simplest approaches is the trapezoidal rule which approximates \(f(x)\) on \([x_1, x_2]\) using a straight line joining \((x_1, f_1)\) and \((x_2, f_2)\) such that,

\[
\int_{x_1}^{x_2} f(x) \, dx \approx \frac{(f_1 + f_2)}{2} h.
\]

This rule can be used repeatedly to approximate the integral over the whole interval, \([x_1, x_n]\), with,

\[
\int_{x_1}^{x_2} f(x) \, dx = h \left( \frac{f_1}{2} + f_2 + \ldots + f_{n-1} + \frac{f_n}{2} \right).
\]
Numerical integration

If \( f(x) \) is smooth enough (with finite fourth derivative), a better approximation is the **Simpson’s rule**, which uses a quadratic polynomial joining the points \((x_1, f_1)\), \((x_2, f_2)\) and \((x_3, f_3)\) to approximate \( f(x) \) on the interval \([x_1, x_3]\) leading to,

\[
\int_{x_1}^{x_3} f(x) \, dx \approx \frac{h}{3} (f_1 + 4f_2 + f_3).
\]

As before, we can use Simpson’s rule repeatedly to approximate the integral over the whole interval, \([x_1, x_n]\), where \( n \) must be odd,

\[
\int_{x_1}^{x_n} f(x) \, dx = \frac{h}{3} (f_1 + 4f_2 + 2f_3 + 4f_4 + 2f_5 + \ldots + 4f_{n-1} + f_n).
\]
Numerical integration

Example 6.1.

Suppose that we wish to estimate the constant of a beta density, \( X \sim B(7, 10) \), with density function,

\[
\pi(x) \propto x^6 (1 - x)^9, \quad \text{for } 0 < x < 1.
\]

We shall try to estimate the beta function, \( B(7, 10) = \int_0^1 x^6 (1 - x)^9 \, dx \), using Simpson’s rule.

- Setting \( h = 0.1 \), we obtain \( B(7, 10) \approx 1.2500 \times 10^{-5} \).
- Setting \( h = 0.05 \), we obtain \( B(7, 10) \approx 1.24876 \times 10^{-5} \).
- The true value of the integral is \( B(7, 10) = 1.24875 \times 10^{-5} \).
Numerical integration

Remarks

- An improvement on Simpson’s basic rule is the adaptive Simpson’s rule, which does not fix the number of subintervals a priori, but instead, continues to subdivide the intervals until the estimated error reaches a given tolerance.

- Other rules have been designed to provide careful choices of the points, \( x_i \), in order to obtain much more accuracy in approximating the required integral. For example, Gaussian quadrature approaches use an approximation where these are determined as the roots of a class of orthogonal polynomials.

- The main problem with numerical integration approaches is the curse of dimensionality. As the dimension of the integral increases, the number of function evaluations necessary to achieve a given tolerance increases very rapidly.
Monte Carlo approaches

Suppose that we have \( X \sim \pi \) and that we wish to estimate the mean of some functional \( E[g(X)] \). Then given a sample, \( x \), of size \( n \) from \( \pi \), we can estimate:

\[
E[g(X)] = \int g(x) \pi(x) \, dx \approx \frac{1}{N} \sum_{i=1}^{N} g(x_i)
\]

When \( E[g(X)^2] \) exists, then we can estimate the sample variance using:

\[
V[g(X)] = E[g(X)^2] - E[g(X)]^2 \approx \frac{1}{N} \sum_{i=1}^{N} g(x_i)^2 - \left( \frac{1}{N} \sum_{i=1}^{N} g(x_i) \right)^2
\]

In many cases however, there is no straightforward way of generating a sample directly from \( \pi \). In such cases, two main alternatives can be considered: importance sampling and rejection sampling.
**Importance sampling**

Suppose that sampling from $\pi$ is complicated. Suppose instead that we can easily sample from another density, say $f$. Now we can write the expected value of $g(X)$ (under $\pi$) as:

$$E_{\pi} [g(X)] = \int g(x) \pi(x) \, dx = \int \frac{g(x) \pi(x)}{f(x)} f(x) \, dx = E_f [\omega(X) g(X)],$$

where $\omega(X) = \frac{\pi(X)}{f(X)}$. Thus, we can approximate the expectation by generating a sample of size $N$ from $f$ and using:

$$E_{\pi} [g(X)] = \frac{1}{N} \sum_{i=1}^{N} \omega(x_i) g(x_i),$$

where $\omega(x_i) = \frac{\pi(x_i)}{f(x_i)}$. 

Importance sampling

Furthermore, if the density $\pi$ is known only up to an integration constant such that $\pi(x) \propto \kappa(x)$, then we can extend the approximation to give:

$$
E_{\pi}[g(X)] = \int g(x) \pi(x) \, dx = \int g(x) \frac{\kappa(x)}{\int \kappa(y) \, dy} \, dx
$$

$$
= \frac{\int g(x) \frac{\kappa(x)}{f(x)} f(x) \, dx}{\int \frac{\kappa(x)}{f(x)} f(x) \, dx} \approx \frac{\sum_{i=1}^{N} \omega(x_i) g(x_i)}{\sum_{i=1}^{N} \omega(x_i)},
$$

where $\omega(x_i) = \frac{\kappa(x_i)}{f(x_i)}$.

Observe that the denominator divided by $N$ is an approximation of the integration constant.
Importance sampling

Example 6.2.

Consider Example 6.1, where $X \sim B(7, 10)$ so that $\pi(x) \propto x^6(1-x)^9$. Suppose now that we wish to estimate the beta function $B(7, 10)$ and:

$$E[X] = \frac{1}{B(7, 10)} \int_0^1 x^7 (1-x)^9 \, dx.$$ 

One possibility is to use a uniform importance function, such that,

$$\omega(x) = \frac{x^6(1-x)^9}{1},$$

and given a uniform sample of size $N$, we can estimate:

$$B(7, 10) \approx \frac{\sum_{i=1}^{N} \omega(x_i)}{N} \quad \text{and} \quad E[X] \approx \frac{\sum_{i=1}^{N} \omega(x_i) x_i}{\sum_{i=1}^{N} \omega(x_i)}.$$
Importance sampling

Example 6.2. (continued)

Given an importance sample of size $N = 1000$, the beta function was estimated to be $1.3574 \times 10^{-5}$ (true value $1.2488 \times 10^{-5}$) and the posterior mean was estimated at $0.4145$ (true mean $0.4118$). Sample sizes of over $100000$ are needed to achieve more than 3 figure accuracy.

Remarks

- In general, the choice of the importance function, $f$, will strongly influence the efficiency of this algorithm. Note that the variance of the importance sampling estimator of $E[g(X)]$ is finite only when:

$$E_f [\omega(X)^2 g(X)^2] = E_\pi [g(X)^2 \omega(X)] = \int g(x)^2 \frac{\pi(x)^2}{f(x)} dx < \infty.$$ 

We cannot choose importance functions with lighter tails than $\pi$. 
Importance sampling

• Moreover, if the importance function, $f$, is not similar to $\pi$ (or $\kappa$) so that the centre of $\pi$ (or $\kappa$) is in the tail of $f$, this can lead to many of the importance weights being very small and thus, the integral estimates may be largely determined by a very small number of data.

• In the Bayesian context, where we often wish to estimate various posterior expectations, then an efficient importance function will be similar to the true posterior, but with heavier tails.

• Another problem is that the importance sampling method does not provide a sample from $\pi$. This can be remedied by using sampling importance resampling (Rubin, 1987).
SIR algorithm

It provides a way of obtaining an approximate sample from $\pi$ by subsampling. If the weights, $\omega(x_i)$ are normalized so that we define:

$$\omega_i = \frac{\omega(x_i)}{\sum_{i=1}^{N} \omega(x_i)}$$

then we can generate an approximate sample, $\tilde{x}$, of size $M < N$ from $\pi$ by setting $\tilde{x}_j = x_i$ with probability $\omega_i$ for $i = 1, \ldots, N$ and $j = 1, \ldots, M$. 
SIR algorithm

Example 6.3.

Recall Example 6.2. The following diagram shows the data simulated using a resample of size $M = 1000$ from an importance sample of size $N = 10000$. The sampled data well approximate the beta density.
Rejection algorithm

We assume we wish to generate a sample from $\pi(x)$, which is not easy to simulate from. The rejection approach chooses to generate data from a proposal distribution, $h(x)$, such that $\pi(x) < Mh(x)$ for some given $M > 0$. The algorithm proceeds as follows:

For $i = 1, \ldots, N$:

1. Generate $\tilde{x}_i \sim h$.
2. Generate $u_i \sim \mathcal{U}(0, 1)$.
3. If $Mu_i h(\tilde{x}_i) < \pi(\tilde{x}_i)$, set $x_i = \tilde{x}_i$.
4. Otherwise, repeat from step 1.

Note that this algorithm clearly reduces to standard Monte Carlo sampling when $h = \pi$. 
Rejection algorithm

Let’s see a proof that this algorithm generates a sample from $\pi$. Assume that $X$ is generated from the rejection algorithm, then,

$$P (X \leq c) = P \left( \tilde{X} \leq c \mid U < \frac{\pi (\tilde{X})}{Mh (\tilde{X})} \right),$$

where $\tilde{X} \sim h$ and $U \sim U(0,1)$. Then,

$$P (X \leq c) = \frac{\int_{-\infty}^{c} \int_{0}^{\pi (\tilde{x})/Mh (\tilde{x})} h (\tilde{x}) \, du \, d\tilde{x}}{\int_{-\infty}^{\infty} \int_{0}^{\pi (\tilde{x})/Mh (\tilde{x})} h (\tilde{x}) \, du \, d\tilde{x}}$$

$$= \frac{\int_{-\infty}^{c} \frac{\pi (\tilde{x})}{Mh (\tilde{x})} h (\tilde{x}) \, d\tilde{x}}{\int_{-\infty}^{\infty} \frac{\pi (\tilde{x})}{Mh (\tilde{x})} h (\tilde{x}) \, d\tilde{x}}$$

$$= \frac{\int_{-\infty}^{c} \pi (\tilde{x}) \, d\tilde{x}}{\int_{-\infty}^{\infty} \pi (\tilde{x}) \, d\tilde{x}} = \int_{-\infty}^{c} \pi (x) \, dx.$$
Rejection algorithm

Example 6.4.

In order to simulate from a truncated normal distribution $X \sim TN(0, 1)$, where $X > \alpha > 0$, we can sample directly from the $N(0, 1)$ density and simply reject those values that fall below $\alpha$. However, this method can be very inefficient if $\alpha$ is large.

Remark

The main problem with rejection sampling is to find a good proposal distribution so that only a small number of candidates are rejected. Note that the probability of accepting a draw is:

$$P \left( U < \frac{\pi (\tilde{X})}{Mh (\tilde{X})} \right) = \int_{-\infty}^{\infty} \int_{0}^{\frac{\pi (\tilde{x})}{Mh (\tilde{x})}} h (\tilde{x}) dud\tilde{x} = \int_{-\infty}^{\infty} \frac{\pi (\tilde{x})}{Mh (\tilde{x})} h (\tilde{x}) d\tilde{x} = \frac{1}{M}$$

so that we would like $M$ to be as close to 1 as possible.
Rejection algorithm

Example 6.5.

In Example 6.4., the probability of accepting a draw was $M_1^{-1} = 1 - \Phi(\alpha)$. Alternatively, we may propose a shifted exponential:

$$h(x) = \lambda e^{-\lambda(x-\alpha)}, \text{ for } x > \alpha,$$

which can produce a probability of acceptance, $M_2^{-1}$, given by,

$$\frac{\pi(x)}{h(x)} = \frac{1}{\lambda \sqrt{2\pi} (1 - \Phi(\alpha))} \exp \left( -\frac{x^2}{2} + \lambda (x - \alpha) \right) \leq \frac{1}{\lambda \sqrt{2\pi} (1 - \Phi(\alpha))} \exp \left( \frac{\lambda^2}{2} - \lambda \alpha \right) = M_2$$

For example, if $\alpha = 1$, the probability of acceptance using a normal proposal is $M_1^{-1} = 1 - \Phi(1) = 0.1533$, while using a shifted exponential with $\lambda = \alpha$ is $M_2^{-1} = 0.6557$. See more sophisticated algorithms in Geweke (1991) and Robert (1995).
Envelope methods

These are refinements of the basic rejection algorithm based on bounding the target density from above and below. Suppose that we can find a proposal density $h$ and a (non-negative) function $g$ such that:

$$g(x) < \pi(x) < Mh(x), \quad \text{for all } x,$$

Then, the following algorithm generates from $X \sim \pi$.

1. Generate $\tilde{x} \sim h$ and $u \sim \mathcal{U}(0,1)$.
2. If $u \leq \frac{g(\tilde{x})}{Mh(\tilde{x})}$, let $x = \tilde{x}$.
3. Otherwise, if $u \leq \frac{\pi(\tilde{x})}{Mh(\tilde{x})}$, let $x = \tilde{x}$.
4. Otherwise, repeat from step 1.
Envelope methods

The advantage of this algorithm is that the number of necessary evaluations of $\pi$ are reduced, and instead, we often only need to evaluate the (simpler) densities, $g$ and $h$. The probability that $\pi$ does not have to be evaluated is $\frac{1}{M} \int g(x) dx$, which reflects the potential gain in using this approach.

One particular case that allows for the simple construction of bounding functions is when the density, $\pi$, is log concave.

Definition

A density $f(x)$ is said to be log concave if $\frac{\partial^2}{\partial x^2} \log f(x) < 0$, $\forall x$.

Most exponential family densities are log-concave. For example, if $X \sim N(\theta, 1)$, then $\frac{\partial^2}{\partial x^2} \log f(x \mid \theta) = -1$.
Adaptive rejection sampling

This algorithm developed by Gilks (1992) and Gilks and Wild (1992) gives a general method of constructing the bounding functions $g$ and $h$ when the target density, $\pi$, is log concave.

Suppose that $S_n$ is a set of points $x_i$ for $i = 0, 1, \ldots, n+1$ in the support of $\pi$ such that $\log \pi(x_i)$ is known up to the same constant. As $\log \pi$ is concave, then the line $L_{i,i+1}$ going through $(x_i, \log \pi(x_i))$ and $(x_{i+1}, \log \pi(x_{i+1}))$ lies below the graph of $\log \pi$ in $(x_i, x_{i+1}]$ and lies above the graph outside this interval.

Thus, for the interval $(x_i, x_{i+1}]$, we can define $\underline{\phi}_n(x) = L_{i,i+1}(x)$ and $\overline{\phi}_n(x) = \min \{L_{i-1,i}(x), L_{i+1,i+2}(x)\}$, which bound $\log \pi$. Defining $g_n(x) = \exp(\underline{\phi}_n(x))$ and $H_n(x) = \exp(\overline{\phi}_n(x))$, we have:

$$g_n(x) \leq \pi(x) \leq H_n(x) = M_n h_n(x)$$

where $h_n$ is a density function.
Adaptive rejection sampling

The big advantage of this approach is its universality. As long as $\pi$ is known to be log concave, it can always be used.
ARS algorithm

1. Initialize $n$ and $S_n$.
2. Generate $\tilde{x} \sim h_n$ and $u \sim \mathcal{U}(0, 1)$.
3. If $u \leq \frac{g_n(\tilde{x})}{M_n h_n(\tilde{x})}$, let $x = \tilde{x}$.
4. Otherwise, if $u \leq \frac{\pi(\tilde{x})}{M_n h_n(\tilde{x})}$, let $x = \tilde{x}$.
5. Otherwise, set $n = n + 1$, $S_{n+1} = S_n \cup \tilde{x}$ and repeat from step 2.

An advantage of this approach is that if a generated value is rejected, it can then be added to the set $S_n$ which improves the bounds on $\pi$ at the next step.
MCMC methods

As simple Monte Carlo algorithms are not always straightforward to implement, another alternative is to use algorithms which generate approximate Monte Carlo samples.

The most popular approach is Markov chain Monte Carlo (MCMC) methods which samples from a Markov chain whose limit distribution is the distribution from which we wish to sample.

From a Bayesian viewpoint, the objective of the MCMC approach is thus to construct a Markov chain with a given stationary distribution, $\pi$, which is the Bayesian posterior distribution.
Markov chains

Definition

A Markov chain, \( \{X_t\} \), is defined to be a sequence of variables, \( X_0, X_1, X_2, \ldots \) such that the distribution of \( X_t \) given the previous values \( X_0, X_1, X_2, \ldots, X_{t-1} \) only depends on \( X_{t-1} \), so that,

\[
P ( X_t \in A \mid X_0 = x_0, X_1 = x_1, \ldots, X_{t-1} = x_{t-1} ) = P ( X_t \in A \mid X_{t-1} = x_{t-1} )
\]

for all \( A, x_0, \ldots, x_{t-1} \).

Most Markov chains that we deal with are time-homogeneous, that is:

\[
P ( X_{t+k} \in A \mid X_t = x ) = P ( X_k \in A \mid X_0 = x ), \quad \text{for any } k.
\]

A simple example of a time-homogeneous Markov chain is a random walk,

\[
X_t = X_{t-1} + \epsilon_t, \quad \text{where } \epsilon_t \sim N(0, \sigma^2).
\]
Markov chains

It is clear that a time-homogeneous Markov chain is completely defined by the initial state, $X_0$, and by the transition kernel,

\[ P(x, y) = P(X_{t+1} = y | X_t = x). \]

For most problems of interest, the Markov chain will take values in a continuous, multivariate state space. However, assume initially that the state space is finite and countable, so that $X_t \in \{1, 2, \ldots, k\}$, for some $k$. Then, we can define the t-step transition probabilities:

\[ p_{ij}(t) = P(X_t = j | X_0 = i). \]

and then, we can consider the conditions under which these probabilities converge, i.e. that:

\[ p_{ij}(t) \to \pi(j), \quad \text{as } t \to \infty \]
Markov chains

It can be shown that a sufficient condition for the existence of a unique stationary distribution is reversibility. A Markov chain with transition probabilities $p_{ij} = P(X_{t+1} = j | X_t = i)$ is said to be reversible if there exists a probability density $\pi$ that satisfies detailed balance, so that for any $i, j$, then,

$$p_{ij}\pi(i) = p_{ji}\pi(j).$$

It is possible to extend this previous argument to Markov chains with a continuous state space, although the conditions for the equilibrium distribution are slightly more technical, see e.g. Robert and Casella (2004). In this case, given a transition kernel, $P(x, y)$, then a stationary distribution $\pi$ must satisfy:

$$\pi(y) = \int P(x, y)\pi(x)dx.$$
Metropolis Hastings algorithm

This is a general algorithm for constructing a Markov chain and was introduced by Metropolis et al. (1953) and extended by Hastings (1970). The general algorithm for generating a chain with equilibrium distribution $\pi$ is as follows:
Metropolis Hastings algorithm

1. Given the current value, $X_t = x$, generate a candidate value, $y$, from a proposal density $q(y|x)$.

2. Calculate the acceptance probability:

$$
\alpha(x, y) = \min\left\{1, \frac{\pi(y) q(x|y)}{\pi(x) q(y|x)}\right\}.
$$

3. With probability $\alpha(x, y)$ define $X_{t+1} = y$ and otherwise reject the proposed value and set $X_{t+1} = x$.

4. Repeat until convergence is judged and a sample of the desired size is obtained.
Why does this algorithm work?

The transition kernel of a move from $x$ to $y$ is:

$$P(x, y) = \alpha(x, y) q(y|x) + \left(1 - \int \alpha(x, y) q(y|x) \, dy\right) \delta_x$$

where $\delta_x$ is the Dirac delta function at $x$. Now, noting that:

$$\frac{\pi(x) q(y|x) \alpha(x, y)}{\pi(y) q(x|y) \alpha(y, x)} = 1,$$

and that:

$$\frac{(1 - \int \alpha(x, y) q(y|x) \, dy) \delta_x}{(1 - \int \alpha(y, x) q(x|y) \, dx) \delta_y} = 1,$$

we have detailed balance:

$$\pi(x)P(x, y) = \pi(y)P(y, x),$$

so that $\pi$ is a stationary distribution of the chain.
Comments on the Metropolis Hastings algorithm

- The acceptance probability only depends on $\pi$ through the ratio $\pi(y)/\pi(x)$. This is particularly useful in the Bayesian context, where the posterior distribution is usually known up to a constant.

- When the proposal density $q(y|x) = \pi(y)$, the Metropolis Hastings acceptance probability is exactly 1 and the algorithm is the same as standard Monte Carlo sampling.

- One might expect that the Metropolis Hastings algorithm would be more efficient if $\alpha(x, y)$ was high. Unfortunately, this is not usually the case. In Roberts et al (1997), it is recommended that for high dimensional models, the acceptance rate for random-walk algorithms (see later) should be around 25% whereas in models of dimension 1 or 2, this should be around 50%.

- However, general results are not available and the efficiency of these algorithms are heavily dependent on the proposal density $q(y|x)$. 
The independence and Metropolis samplers

The **independence sampler** defines a proposal density $q(y|x) = q(y)$ independent of $x$. This will often work well if the density $q$ is similar to $\pi$, although with somewhat heavier tails, similarly to the Monte Carlo rejection sampler.

Another alternative is the **Metropolis sampler** which has the property that $q(x|y) = q(y|x)$. One small advantage of this approach is that the acceptance probability simplifies down to:

$$\alpha(x, y) = \frac{\pi(y)}{\pi(x)}.$$

A special case is the **random walk Metropolis algorithm**, which assumes that $q(y|x) = q(|y - x|)$. For example, in univariate problems, one might consider a normal proposal density $q(y|x) = N(x, \sigma^2)$, where the value of $\sigma$ can be adjusted to achieve an acceptable acceptance rate.
Metropolis Hastings algorithm

Example 6.6.

Consider a sample of size $n$ from a Cauchy distribution, $X|\theta \sim \mathcal{C}(\theta, 1)$:

$$f(x | \theta) = \frac{1}{\pi \left(1 + (x - \theta)^2\right)}, \quad \text{for } -\infty < \theta < \infty.$$ 

Given a uniform prior for $\theta$, the posterior distribution is:

$$p(\theta | x) \propto \prod_{i=1}^{n} \frac{1}{1 + (x_i - \theta)^2}.$$
Metropolis Hastings algorithm

Example 6.6. (continued)

One way of sampling this distribution is to use a random walk Metropolis algorithm. We could use a Cauchy proposal density, $\tilde{\theta}|\theta \sim C(\theta, \sigma)$:

$$q(\tilde{\theta} | \theta) = \frac{1}{\pi \sigma \left(1 + \left(\frac{\tilde{\theta} - \theta}{\sigma}\right)^2\right)}.$$  

The scale parameter, $\sigma$, can be adjusted to achieve the desired acceptance rate. In this case, the probability of accepting a proposed value, $\tilde{\theta}$, given the current value, $\theta$, is:

$$\alpha(\theta, \tilde{\theta}) = \min \left\{ \frac{1}{1 + \left(\frac{\tilde{\theta} - \theta}{\sigma}\right)^2} \prod_{i=1}^{n} \frac{1 + (x_i - \theta)^2}{1 + (x_i - \tilde{\theta})^2}, 1 \right\}.$$
Metropolis Hastings algorithm

Example 6.6. (continued)

As an alternative, an independence sampler could be proposed. In this case, we might assume a Cauchy proposal distribution, \( \tilde{\theta} \sim C(m, \tau) \), where the location parameter, \( m \), is the sample median. In this case, the acceptance probability is:

\[
\alpha(\theta, \tilde{\theta}) = \min \left\{ 1, \frac{1 + \left( \frac{\tilde{\theta} - m}{\tau} \right)^2}{1 + \left( \frac{\theta - m}{\tau} \right)^2} \prod_{i=1}^{n} \frac{1 + (x_i - \theta)^2}{1 + (x_i - \tilde{\theta})^2} \right\}
\]

A sample of 10 data were generated from a Cauchy distribution, \( X \sim C(1, 1) \), with the following results:

\[
x = \begin{bmatrix}
-5.1091 & -0.7651 & 0.9261 & 1.0232 & 1.1669 \\
1.2702 & 2.4846 & 2.5375 & 3.3476 & 3.6066
\end{bmatrix}
\]
Metropolis Hastings algorithm

Example 6.6. (continued)

Both the random walk sampler (with $\sigma = 0.3$) and the independence sampler (with $\tau = 0.5$) were run for 10000 iterations, starting from the sample median. In the first case 64.16% of the proposed values were accepted and in the second case 52.11%. Kernel density estimates of the posterior density of $\theta$ are given in the following diagram.
Block Metropolis Hastings

When the dimension of $X$ is large, it can often be difficult to find a reasonable proposal density. In this case, we can divide $X$ into blocks, say $X = (X_1, \ldots, X_k)$ and construct a chain with these smaller blocks.

Suppose initially that $X = (X_1, X_2)$ and define two proposal densities $q_1(y_1|x_1, x_2)$ and $q_2(y_2|x_1, x_2)$ to generate candidate values for each component. Then, define the acceptance probabilities:

$$
\alpha_1(x_1, y_1|x_2) = \min \left\{ 1, \frac{\pi(y_1|x_2) q_1(x_1|y_1, x_2)}{\pi(x_1|x_2) q_1(y_1|x_1, x_2)} \right\}
$$

$$
\alpha_1(x_2, y_2|x_1) = \min \left\{ 1, \frac{\pi(y_2|x_1) q_2(x_2|x_1, y_2)}{\pi(x_2|x_1) q_2(y_2|x_1, x_2)} \right\}
$$

where the densities $\pi(x_1|x_2)$ and $\pi(x_2|x_1)$ are the conditional densities and $\pi(x_1|x_2) \propto \pi(x_1, x_2)$.

The algorithm now proceeds by successively sampling from each block in turn.
Slice sampling

The slice sampler (Neal, 2003) is an attractive approach when the state space is relatively low dimensional.

The idea to sample from $\pi(x)$ is sampling uniformly from the region:

$$\mathcal{A} = \{(x, u) : 0 \leq u \leq \pi(x)\}$$

A Markov chain that converges to this uniform distribution can be constructed by alternating sampling from $(U|X = x)$ and $(X|U = u)$, which are both uniformly distributed.

Then, the general algorithm for sampling from $\pi$ is:

1. Given a current value, $x_t$, simulate $u_{t+1} \sim \mathcal{U}[0, \pi(x_t)]$.
2. Simulate $x_{t+1} \sim \mathcal{U}[\{x : \pi(x) < u_{t+1}\}]$.

It is not important whether the constant of integration is known or not.
Slice sampling

Example 6.7.

Suppose that we wish to sample from an exponential density $X \sim E(\lambda)$. Then, we know that $\pi(x) \propto e^{-\lambda x}$ and a slice sampler could proceed as:

1. Given $x_t$, generate $u_{t+1} \sim U[0, e^{-\lambda x_t}]$.
2. Generate $x_{t+1} \sim U[0, -\frac{1}{\lambda} \log u_{t+1}]$.

The following diagram shows the results of 10000 iterations when $\lambda = 2$. 
Gibbs sampling

Assume that $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_k)$ has joint distribution $\pi$ and that the conditional distributions $\pi(\mathbf{X}_i | \mathbf{X}_{-i})$ are all available, where $\mathbf{X}_{-i} = (\mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_k)$. Then, the Gibbs sampler generates an approximate sample from $\pi$ by successively sampling from these conditional densities. Thus, assuming that the current values are $\mathbf{x}_t$, then the algorithm is the following:

1. Generate $\mathbf{x}_{1,t+1} \sim \pi(\cdot | \mathbf{x}_{-1,t}).$
2. Generate $\mathbf{x}_{2,t+1} \sim \pi(\cdot | \mathbf{x}_{1,t+1}, \mathbf{x}_3,t, \ldots, \mathbf{x}_k,t).$
3. : 
4. Generate $\mathbf{x}_{k,t+1} \sim \pi(\cdot | \mathbf{x}_{-k,t+1}).$
Gibbs sampling

Remarks

• We can note that Gibbs sampling is a particular version of block Metropolis Hastings algorithm where the proposal distribution for $X_i$ is exactly the conditional distribution $\pi(X_i|X_{-i})$ so that the acceptance probability is always equal to 1.

• Gibbs sampling can be applied in a remarkably large number of problems.
Gibbs sampling

Example 6.8.

Suppose that the lifetime (in hours) of a machine, $X$, has normal distribution so that $X|\theta \sim N(\theta, 1)$ and that we observe $n$ machines during $\alpha$ hours. If, at the end of this time, $n_1$ machines have failed, with failure times $x_1, \ldots, x_{n_1}$ and $n_2 = n - n_1$ machines are still working, then the likelihood function is:

$$l(\theta|x) \propto \exp \left( -\frac{n_1}{2} (\theta - \bar{x}_1)^2 \right) \left( 1 - \Phi (\alpha - \theta) \right)^{n_2},$$

where $\bar{x}_1 = \frac{1}{n_1} \sum_{i=1}^{n_1} x_i$. Thus, an explicit form for the posterior of $\theta$ (supposing a uniform prior) is unavailable.

However, suppose that we knew the true values of the latent variables, $X_{n_1+1} = x_{n_1+1}, \ldots, X_n = x_n$. Then it is clear that $\theta|x \sim N(\bar{x}, \frac{1}{n})$ where $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$. 
Gibbs algorithm

Example 6.8. (continued)

Also, for $i = n_1 + 1, \ldots, n$, we have $X_i | \theta, X_i > \alpha \sim N(\theta, 1)$ truncated onto the region $X_i > \alpha$. Therefore, we can set up a simple Gibbs sampling algorithm to estimate the posterior density of $\theta$ as follows.

1. Set $t = 0$ and fix an initial value $\theta_0$.
2. For $i = n_1 + 1, \ldots, n$, generate $x_{i,t} \sim T \mathcal{N}(\theta_t, 1)$.
3. Calculate $\bar{x}_t = \sum_{i=1}^{n_1} x_i + \sum_{i=n_1+1}^n x_{i,t}$.
4. Generate $\theta_{t+1} \sim N(\bar{x}_t, \frac{1}{n})$.
5. $t = t + 1$. Go to 2.
Universal implementation of Gibbs sampling algorithms

In many cases, the conditional distributions used in Gibbs samplers are logconcave. In these cases, universal Gibbs samplers can be set up by using the ARS to sample these conditional distributions.

For non log-concave distributions, the adaptive rejection Metropolis sampler (ARMS) was introduced in Gilks et al. (1995).

These algorithms form the basis of the Winbugs program.

A disadvantage of such universal algorithms is however that they are often inefficient. It is generally preferable to implement specific algorithms tailored to the particular problem.
MCMC convergence assessment

When running an MCMC algorithm, it is important to assess when the sampled values $x_t$ have approximately converged to the stationary distribution $\pi$. This will depend on how well the MCMC algorithm is able to explore the state space also on the correlation between the $x_t$'s.

Secondly, we need to assess the convergence of MCMC averages, e.g. 
$$\frac{1}{n} \sum_{i=1}^{n} x_t \rightarrow E[X_t]$$
and finally we need to be able to assess how close a given sample is to being independent and identically distributed.

One possibility is to consider running the chain various times with different, disperse starting values. Then, we could assess the convergence of the chain by examining when sample means of the functions of interest generated from each run have converged. Other, formal diagnostics are given in Gelman and Rubin (1992).

The alternative is to use a single run of the chain.
MCMC convergence assessment

In this case, we can produce graphs of $x_t$ against $t$ to show the mixing of the chain and any deviations from stationarity.

The following graph shows the trace plot of the sample of $\theta$ in Example 6. The chain presents a good mixing performace.
MCMC convergence assessment

The following diagram from Wiper (2007) shows examples of badly mixing chains.
MCMC convergence assessment

Secondly, we can plot running means of the parameters of interest to see when they have converged. The following diagram shows the estimates of $E[\theta|x]$ obtained from the two algorithms used in Example 6.

![Graph showing MCMC convergence](image)

It can be seen that the means appear to have converged after about 3000 iterations. Thus, one possibility is to run the sampler for longer, using these initial iterations as a burn in period.
MCMC convergence assessment

Thirdly, we can plot the autocorrelation functions of the generated values. In general, as we are generating from a Markov chain, the successive values, $x_t$, will be positively correlated. The following diagram shows the ACF of the parameter $\theta$ obtained from the two algorithms used in Example 6.

In the random walk sampler, the autocorrelation disappears after about lag 9, while in the independence sampler about lag 4. One possibility is thus to thin the samples, choosing just every 9th and 4th datum, resp., which are now approximately independent.
Other algorithms

Reversible jump

This approach (Magenta 1995) is basically a Metropolis Hasting sampler, which allows the chain to move over a variably dimensioned model space.

Perfect sampling

This method, developed by Propp and Wilson (1996), uses the idea of coupling from the past in order to generate an exact MCMC sample from $\pi$, avoiding the need for convergence diagnostics. See e.g. http://dbwilson.com/exact/

Particle filtering

This is an alternative approach to MCMC based on importance sampling and particularly suitable for sequential inference problems. See e.g. Doucet et al (2000) or http://en.wikipedia.org/wiki/Particle_filter