Bayesian Inference

Chapter 3: MCMC and other simulation methods

Conchi Ausín and Mike Wiper
Department of Statistics
Universidad Carlos III de Madrid

Master in Business Administration and Quantitative Methods
Master in Mathematical Engineering

Chapter 3: MCMC and other simulation methods

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

- Wiper, M.P. (2007). Introduction to Markov chain Monte Carlo simulation. In *Encyclopedia of Statistics in Quality and Reliability,* Wiley, pp 1014-1020.

Introduction

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

$$f(\boldsymbol{\theta} \mid \mathbf{x}) = \frac{f(\mathbf{x} \mid \boldsymbol{\theta}) f(\boldsymbol{\theta})}{\int f(\mathbf{x} \mid \boldsymbol{\theta}) f(\boldsymbol{\theta}) d\theta},$$

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

The different techniques which might be applied are as follows:

- Numerical integration (not valid for large dimensions)
- Gaussian approximations (not valid for small samples)
- Monte Carlo approaches:
  - direct methods
  - via Markov chains

Contents

Monte Carlo approximation

Recall from Chapter 1 that if we can generate a sample, $\{\theta_1, \ldots, \theta_N\}$, directly from the posterior distribution, $f(\theta \mid \mathbf{x})$, we can estimate the mean of some functional, $g(\theta)$, using:

$$E\left[g\left(\theta\right) \mid \mathbf{x}\right] = \int g\left(\theta\right) f\left(\theta \mid \mathbf{x}\right) d\theta \approx \frac{1}{N} \sum_{i=1}^{N} g\left(\theta_i\right)$$

In many cases however, there is no straightforward way of generating a sample directly from $f(\theta \mid \mathbf{x})$. In such cases, two main alternatives can been considered: importance sampling and rejection sampling.

Importance sampling

Suppose that sampling from $f(\theta \mid \mathbf{x})$ is complicated. Suppose instead that we can easily sample from another density, $h(\theta)$, called the importance function. Now, we can write:

$$\begin{aligned} E\left[g(\theta) \mid \mathbf{x}\right] &= \int g(\theta) f\left(\theta \mid \mathbf{x}\right) d\theta \\ &= \int \frac{g(\theta) f(\theta|\mathbf{x})}{h(\theta)} h(\theta) d\theta \\ &= \int g(\theta)\omega(\theta)h(\theta)d\theta \end{aligned}$$

where

$$\omega(\theta) = \frac{f\left(\theta \mid \mathbf{x}\right)}{h(\theta)}.$$

Thus, we can generate a sample of size N from h and approximate:

$$E\left[g(\theta) \mid \mathbf{x}\right] \approx \frac{1}{N} \sum_{i=1}^{N} g\left(\theta_i\right) \omega\left(\theta_i\right),$$

where $\omega\left(\theta_i\right) = \frac{f(\theta_i|\mathbf{x})}{h(\theta_i)}$.

Importance sampling

Furthermore, if the posterior density is known only up to an integration constant such that:

$$f(\theta \mid \mathbf{x}) \propto k(\theta),$$

then, we can extend the approximation to give:

$$E[g(\theta) \mid \mathbf{x}] \approx \frac{\sum_{i=1}^{N} g(\theta_i)\,\omega(\theta_i)}{\sum_{i=1}^{N} \omega(\theta_i)},$$

where $\omega(\theta_i) = \frac{k(\theta_i)}{h(\theta_i)}$.

In this case, the denominator (divided by N) is an approximation of the integration constant.

Importance sampling

Example

Consider a coin tossing example where 13 heads are observed in 20 tosses and a uniform prior is assumed for θ then, the posterior density is:

$$f(\theta \mid \mathbf{x}) \propto \theta^{14-1}(1-\theta)^{8-1}.$$

Suppose now that we wish to estimate the beta function B(14, 8) and the posterior mean:

$$E[\theta \mid \mathbf{x}] = \frac{1}{B(14,8)} \int_0^1 \theta^{14} (1-\theta)^7 \, dx.$$

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

$$\omega(\theta) = \frac{\theta^{14}(1-\theta)^7}{1},$$

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

$$B(14,8) \approx \frac{\sum_{i=1}^N \omega(\theta_i)}{N} \quad \text{and} \quad E[\theta \mid \mathbf{x}] \approx \frac{\sum_{i=1}^N \omega(\theta_i)\,\theta_i}{\sum_{i=1}^N \omega(\theta_i)}$$

Importance sampling

Remarks

- In general, the choice of the importance function, h, will strongly influence the efficiency of this algorithm. Note that the variance of the estimator of $E[g(\theta) \mid \mathbf{x}]$ is finite only when:

$$\int g(\theta)^2 \omega(\theta)^2 h(\theta) d\theta = \int g(\theta)^2 \frac{f(\theta \mid \mathbf{x})^2}{h(\theta)} d\theta < \infty.$$

  We cannot choose importance functions with lighter tails than f.

- Moreover, if the importance function, h, is not similar to the posterior density, $f(\theta \mid \mathbf{x})$ (or k), so that the centre of f (or k) is in the tail of h, 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.

- Therefore, an an efficient importance function will be similar to the true posterior, but with heavier tails.

Sampling Importance Resampling (SIR)

The importance sampling algorithm does not provide a sample from the posterior distribution. This can be remedied by using the sampling importance resampling method where the weights, $\omega(\theta_i)$ are normalized so that:

$$\omega_i = \frac{\omega(\theta_i)}{\sum_{i=1}^{N} \omega\left(\theta_i\right).}$$

Then, we can generate an approximate sample, $\left\{\tilde{\theta}_1, \ldots \tilde{\theta}_M\right\}$, of size $M < N$ from $f(\theta \mid \mathbf{x})$ by setting $\tilde{\theta}_j = \theta_i$ with probability ω_i for $i = 1, \ldots, N$ and $j = 1, \ldots, M$.

Rejection algorithm

Assume the we wish to generate a sample from $f(\theta \mid \mathbf{x})$, which is not easy to simulate from. The rejection approach chooses to generate data from a proposal distribution, $h(\theta)$, such that $f(\theta \mid \mathbf{x}) \leq Mh(\theta)$ for some given $M \geq 1$. The algorithm proceeds as follows:

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

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

Note that this algorithm clearly reduces to standard Monte Carlo sampling when $h(\theta) = f(\theta \mid \mathbf{x})$.

Rejection algorithm

Let's see a proof that this algorithm generates a sample from $f(\theta \mid \mathbf{x})$. Assume that θ is generated from the rejection algorithm, then,

$$P(\theta \leq c) = P\left(\tilde{\theta} \leq c \mid u < \frac{f\left(\tilde{\theta} \mid \mathbf{x}\right)}{Mh\left(\tilde{\theta}\right)}\right),$$

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

$$P(\theta \leq c) = \frac{\int_{-\infty}^{c} \int_{0}^{f\left(\tilde{\theta}|\mathbf{x}\right)/Mh\left(\tilde{\theta}\right)} h\left(\tilde{\theta}\right) du d\tilde{\theta}}{\int_{-\infty}^{\infty} \int_{0}^{f\left(\tilde{\theta}|\mathbf{x}\right)/Mh\left(\tilde{\theta}\right)} h\left(\tilde{\theta}\right) du d\tilde{\theta}} = \frac{\int_{-\infty}^{c} \frac{f\left(\tilde{\theta} \mid \mathbf{x}\right)}{Mh\left(\tilde{\theta}\right)} h\left(\tilde{\theta}\right) d\tilde{\theta}}{\int_{-\infty}^{\infty} \frac{f\left(\tilde{\theta} \mid \mathbf{x}\right)}{Mh\left(\tilde{\theta}\right)} h\left(\tilde{\theta}\right) d\tilde{\theta}}$$

$$= \frac{\int_{-\infty}^{c} f\left(\tilde{\theta} \mid \mathbf{x}\right) d\tilde{\theta}}{\int_{-\infty}^{\infty} f\left(\tilde{\theta} \mid \mathbf{x}\right) d\tilde{\theta}} = \int_{-\infty}^{c} f(\theta \mid \mathbf{x}) d\theta.$$

Rejection algorithm

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:

$$P\left(u < \frac{f\left(\tilde{\theta} \mid \mathbf{x}\right)}{Mh\left(\tilde{\theta}\right)}\right) = \int_{-\infty}^{\infty} \int_{0}^{\frac{f\left(\tilde{\theta}|\mathbf{x}\right)}{Mh\left(\tilde{\theta}\right)}} h\left(\tilde{\theta}\right) du d\tilde{\theta}$$

$$= \int_{-\infty}^{\infty} \frac{f\left(\tilde{\theta} \mid \mathbf{x}\right)}{Mh\left(\tilde{\theta}\right)} h\left(\tilde{\theta}\right) d\tilde{\theta} = \frac{1}{M},$$

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

Rejection algorithm

Example

Suppose that the posterior distribution of θ follows a truncated normal $TN(0,1)$, where $\theta > \alpha > 0$, we can sample directly from the $N(0,1)$ density and simply reject those values that fall below α. However, this method can be very inefficient if α is large since the probability of accepting a draw is $M_1^{-1} = 1 - \Phi(\alpha)$, where Φ denotes the standard normal cdf.
Alternatively, we may propose a shifted exponential:

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

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

$$\begin{aligned} \frac{f(\theta \mid \mathbf{x})}{h(\theta)} &= \frac{1}{\lambda\sqrt{2\pi}\left(1-\Phi(\alpha)\right)} \exp\left(-\frac{\theta^2}{2} + \lambda(\theta-\alpha)\right) \\ &\leq \frac{1}{\lambda\sqrt{2\pi}\left(1-\Phi(\alpha)\right)} \exp\left(\frac{\lambda^2}{2} - \lambda\alpha\right) = M_2 \end{aligned}$$

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 $h(\theta)$ and a (non-negative) function $g(\theta)$ such that:

$$g(\theta) < f(\theta \mid \mathbf{x}) < Mh(\theta), \quad \text{for all } \theta.$$

Then, the following algorithm generates from $\theta \sim f(\theta \mid \mathbf{x})$.

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

Envelope methods

The advantage of this algorithm is that the number of necessary evaluations of $f(\theta \mid \mathbf{x})$ are reduced, and instead, we often only need to evaluate the (simpler) densities, $g(\theta)$ and $h(\theta)$. The probability that $f(\theta \mid \mathbf{x})$ does not have to be evaluated is $\frac{1}{M} \int g(\theta) d\theta$, 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, $f(\theta \mid \mathbf{x})$, is log concave.

Definition

A density $f(\theta)$ is said to be log concave if $\frac{\partial^2}{\partial \theta^2} \log f(\theta) < 0, \quad \forall \theta$.
Most exponential family densities are log-concave. For example, if $\theta \sim N(0, 1)$, then $\frac{\partial^2}{\partial \theta^2} \log f(\theta) = -1$

Adaptive rejection sampling

This algorithm gives a general method of constructing the bounding functions $g(\theta)$ and $h(\theta)$ when the posterior density, $f(\theta \mid \mathbf{x})$, is log concave.

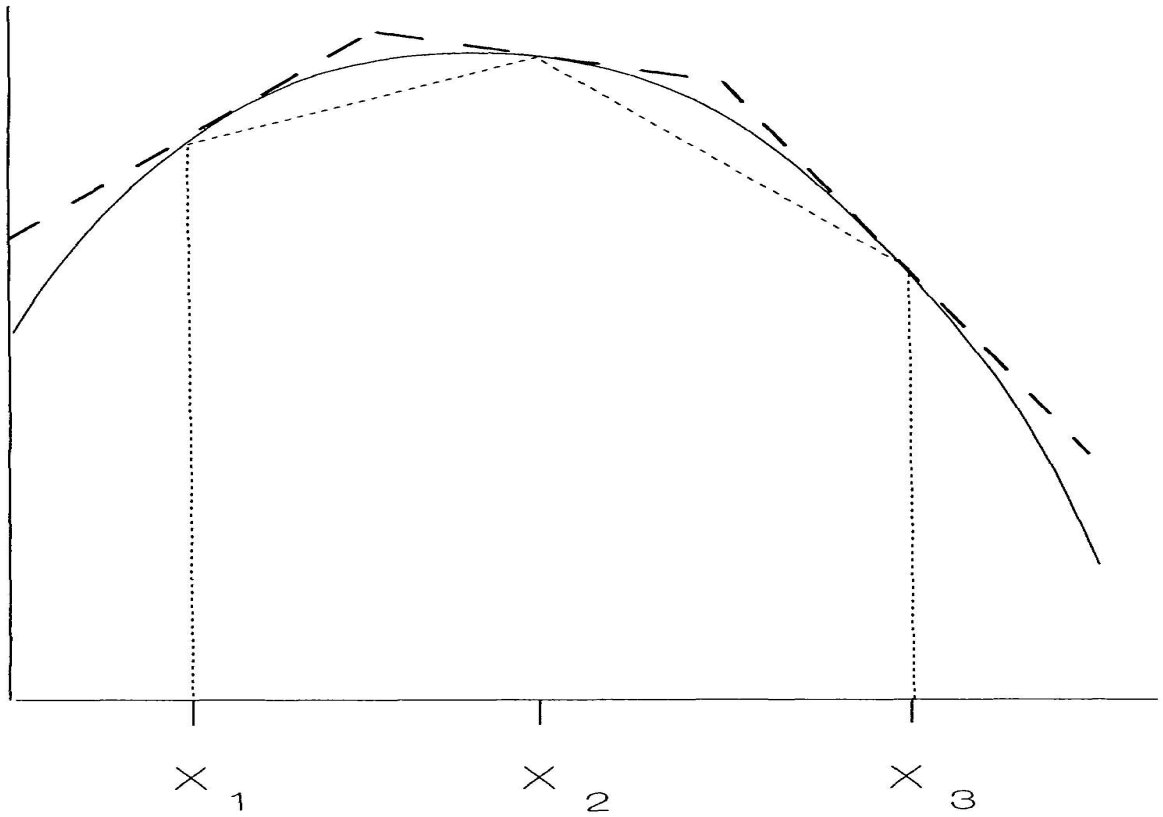
Suppose that S_n is a set of points θ_i, for $i = 0, 1, \ldots, n+1$, in the support of $f(\theta \mid \mathbf{x})$, which is known up to a constant and log-concave. Then, the line $L_{i,i+1}$ going through $(\theta_i, \log f(\theta_i \mid \mathbf{x}))$ and $(\theta_{i+1}, \log f(\theta_{i+1} \mid \mathbf{x}))$ lies below the graph of $\log f$ in $(\theta_i, \theta_{i+1}]$ and lies above the graph outside this interval.

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

$$g_n(\theta) \leq f(\theta \mid \mathbf{x}) \leq H_n(\theta) = M_n h_n(\theta)$$

where h_n is a density function.

Adaptive rejection sampling



The big advantage of this approach is its universality. As long as $f(\theta \mid \mathbf{x})$ is known to be log concave, it can always be used.

ARS algorithm

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

Thus, the objective of the MCMC approach is to construct a Markov chain with a given stationary distribution which is the posterior distribution, $f(\theta \mid \mathbf{x})$.

Markov chains

Definition

A Markov chain, $\{X_t\}$, is 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,

$$P\left(X_t \in A \mid X_0 = x_0, X_1 = x_1, ..., X_{t-1} = x_{x_{t-1}}\right) = P\left(X_t \in A \mid X_{t-1} = x_{x_{t-1}}\right)$$

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

A (time-homogeneous) Markov chain is completely defined by the initial state, X_0, and by the transition kernel,

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

Markov chains

Assume initially that the state space is finite and countable, so that $X_t \in \{1, 2, \ldots, k\}$, for some k with t-step transition probabilities:

$$p_{ij}(t) = P(X_t = j | X_0 = i).$$

It can be shown that these probabilities will converge to a stationary distribution, i.e.,

$$p_{ij}(t) \to \pi(j), \qquad \text{as } t \to \infty$$

if the Markov chain is reversible, that is, if there exists a probability density π 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.

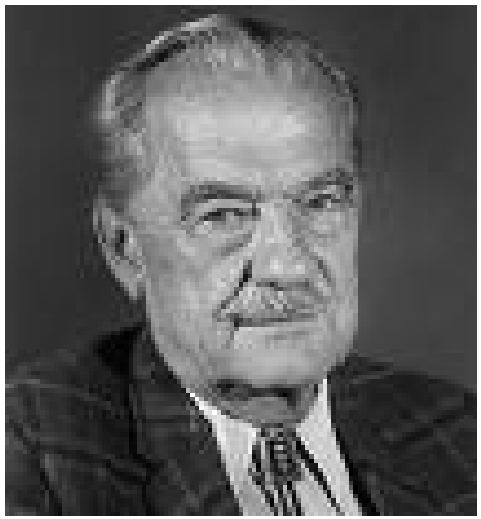
Monte Carlo Markov chains

- Assume now that we wish to simulate values from a posterior distribution:

$$f(\theta \mid \mathbf{x}) \propto f(\mathbf{x} \mid \theta) f(\theta).$$

- The idea of MCMC methods is to sample values from a Markov Chain whose stationary distribution is $f(\theta \mid \mathbf{x})$.
- Then, each simulated value, θ_t, from such a Markov chain will only depend on the previous generated value, θ_{t-1}.
- If the MCMC algorithm is implemented correctly, convergence is ensured for any initial values.
- A large number of iterations will be required to approximate the stationary distribution.
- An initial number of iterations, called burn-in interations, will be discarded in order to allow the Markov chain to reach the stationary state.

Metropolis Hastings algorithm



Metropolis

The Metropolis Hastings (MH) algorithm is a general algorithm for constructing Monte Carlo Markov chains.

Metropolis Hastings algorithm

Set an initial value θ_0.

1. Given the current value, θ_t, generate a candidate value, $\tilde{\theta}$, from a proposal density $q(\tilde{\theta} \mid \theta_t)$.

2. Calculate the acceptance probability:

$$\alpha(\theta_t, \tilde{\theta}) = \min \left\{ 1, \frac{f(\tilde{\theta}) f\left(\mathbf{x} \mid \tilde{\theta}\right) q\left(\theta_t \mid \tilde{\theta}\right)}{f(\theta_t) f\left(\mathbf{x} \mid \theta_t\right) q\left(\tilde{\theta} \mid \theta_t\right)} \right\}.$$

3. With probability $\alpha(\theta_t, \tilde{\theta})$ define $\theta_{t+1} = \tilde{\theta}$ and otherwise reject the proposed value and set $\theta_{t+1} = \theta_t$.

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 θ_t to $\tilde{\theta}$ is:

$$P\left(\theta_t, \tilde{\theta}\right) = \alpha\left(\theta_t, \tilde{\theta}\right) q\left(\tilde{\theta}|\theta_t\right) + \left(1 - \int \alpha\left(\theta_t, \tilde{\theta}\right) q\left(\tilde{\theta}|\theta_t\right) d\tilde{\theta}\right) \delta_{\theta_t}$$

where δ_{θ_t} is the Dirac delta function at θ_t. Now, noting that:

$$f\left(\theta_t\right) f\left(\mathbf{x} \mid \theta_t\right) q\left(\tilde{\theta}|\theta_t\right) \alpha\left(\theta_t, \tilde{\theta}\right) = f\left(\tilde{\theta}\right) f\left(\mathbf{x} \mid \tilde{\theta}\right) q\left(\theta_t|\tilde{\theta}\right) \alpha\left(\tilde{\theta}, \theta_t\right),$$

and that:

$$\left(1 - \int \alpha\left(\theta_t, \tilde{\theta}\right) q\left(\tilde{\theta}|\theta_t\right) d\tilde{\theta}\right) \delta_{\theta_t} = \left(1 - \int \alpha\left(\tilde{\theta}, \theta_t\right) q\left(\theta_t|\tilde{\theta}\right) d\theta_t\right) \delta_{\tilde{\theta}},$$

we have detailed balance:

$$\pi(\theta_t) P(\theta_t, \tilde{\theta}) = \pi(\tilde{\theta}) P(\tilde{\theta}, \theta_t),$$

so that π is a stationary distribution of the chain.

The independence and Metropolis samplers

The independence sampler defines a proposal density $q(\tilde{\theta}|\theta_t) = q(\tilde{\theta})$ independent of θ_t. This will often work well if the density q is similar to $f(\theta \mid \mathbf{x})$, although with somewhat heavier tails, similarly to the Monte Carlo rejection sampler.

Another alternative is the Metropolis sampler which has the property that $q(\tilde{\theta}|\theta_t) = q(\theta_t \mid \tilde{\theta})$. One advantage of this approach is that the acceptance probability simplifies down to:

$$\alpha(\theta_t, \tilde{\theta}) = \min\left\{1, \frac{f(\tilde{\theta})f\left(\mathbf{x} \mid \tilde{\theta}\right)}{f(\theta_t)f\left(\mathbf{x} \mid \theta_t\right)}\right\}.$$

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

Metropolis Hastings algorithm

Example

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

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

Given a uniform prior for θ, the posterior distribution is:

$$p(\theta \mid \mathbf{x}) \propto \prod_{i=1}^{n} \frac{1}{1 + (x_i - \theta)^2}.$$

We are going to develop two algorithms: a random walk Metropolis and an independence sampler. We will compare the obtained results.

Comments on the Metropolis Hastings algorithm

- One might expect that the Metropolis Hastings algorithm would be more efficient if α was high, but this is not usually the case.
- For example, in random walk Metropolis algorithms, it is recommended that the acceptance rate should be around 25% for high dimensional models, 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 $q(\tilde{\theta}|\theta_t)$.

Block Metropolis Hastings

When the dimension of $\boldsymbol{\theta}$ is large, it can often be difficult to find a reasonable proposal density. Then, we can divide $\boldsymbol{\theta}$ into blocks.

Suppose that $\boldsymbol{\theta} = (\theta_1, \theta_2)$ and define two proposals, $q_1(\tilde{\theta_1}|\theta_{1t}, \theta_{2t})$ and $q_2(\tilde{\theta_2}|\theta_{1t}, \theta_{2t})$, to generate candidate values for each component.

1. Generate a candidate $\tilde{\theta}_1 \sim q_1\left(\tilde{\theta}_1 \mid \theta_{1t}, \theta_{2t}\right)$ and accept with probability:

$$\alpha_1(\theta_{1t}, \tilde{\theta}_1 \mid \theta_{2t}) = \min\left\{1, \frac{f(\tilde{\theta}_1) f\left(\mathbf{x} \mid \tilde{\theta}_1, \theta_{2t}\right) q_1\left(\theta_{1t} \mid \tilde{\theta}_1, \theta_{2t}\right)}{f(\theta_{1t}) f\left(\mathbf{x} \mid \theta_{1t}, \theta_{2t}\right) q_1\left(\tilde{\theta}_1 \mid \theta_{1t}, \theta_{2t}\right)}\right\}$$

2. Generate a candidate $\tilde{\theta}_2 \sim q_2\left(\tilde{\theta}_2 \mid \theta_{1t+1}, \theta_{2t}\right)$ and accept with analogous probability $\alpha_2(\theta_{2t}, \tilde{\theta}_2 \mid \theta_{1t+1})$.

Block Metropolis Hastings

Example

Assume a sample of size n from a Gamma distribution whose density is given by:

$$f(x \mid \nu, \lambda) = \frac{\lambda^{\nu}}{\Gamma(\nu)} x^{\nu-1} \exp(-\lambda x), \qquad x > 0.$$

Assuming the following prior distributions:

$$\nu \sim \mathcal{U}(0, 100), \qquad \lambda \sim \mathcal{G}(a, b),$$

construct an MCMC algorithm to sample from the joint posterior distribution.

Gibbs sampling

Suppose that $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ and we can easily sample from all conditional posterior distributions, $f(\theta_i \mid \mathbf{x}, \theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_k)$, for $i = 1, \ldots, k$.

The Gibbs sampling algorithm is a particular version of block MH algorithm where the proposal distributions are exactly the conditional posterior distributions, so that the acceptance probability is always equal to 1.

1. Simulate $\theta_{1,t+1} \sim f(\theta_1 \mid \mathbf{x}, \theta_{2,t}, \ldots, \theta_{k,t})$
2. Simulate $\theta_{2,t+1} \sim f(\theta_2 \mid \mathbf{x}, \theta_{1,t+1}, \theta_{3,t} \ldots, \theta_{k,t})$
3. Simulate $\theta_{3,t+1} \sim f(\theta_3 \mid \mathbf{x}, \theta_{1,t+1}, \theta_{2,t+1}, \theta_{4,t}, \ldots, \theta_{k,t})$
4. $\vdots$
5. Simulate $\theta_{k,t+1} \sim f(\theta_k \mid \mathbf{x}, \theta_{1,t+1}, \ldots, \theta_{k-1,t+1})$

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

Gibbs sampling

Example

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 α 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:

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

where $\bar{x}_1 = \frac{1}{n_1}\sum_{i=1}^{n_1} x_i$. Thus, an explicit form for the posterior of θ (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 $\theta|\mathbf{x} \sim N(\bar{x}, \frac{1}{n})$ where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

Slice sampling

The idea to sample from $f(\theta \mid \mathbf{x})$ is based on generating values uniformly from the region:

$$\mathcal{A} = \{(\theta, u) : 0 \leq u \leq f(\theta) f(\mathbf{x} \mid \theta)\}$$

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

Then, the general algorithm for sampling from $f(\theta \mid \mathbf{x})$ is:

1. Given a current value, θ_t, simulate $u_{t+1} \sim \mathcal{U}[0, f(\theta_t) f(\mathbf{x} \mid \theta_t)]$.
2. Given a current value, u_{t+1}, simulate $\theta_{t+1} \sim \mathcal{U}[\{\theta : f(\theta) f(\mathbf{x} \mid \theta) > u_{t+1}\}]$.

Slice sampling

Example

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

1. Given θ_t, generate $u_{t+1} \sim \mathcal{U}[0, e^{-\lambda\theta_t}]$.
2. Given u_{t+1}, generate $\theta_{t+1} \sim \mathcal{U}[0, -\frac{1}{\lambda}\log u_{t+1}]$.

MCMC convergence assesment

When running an MCMC algorithm, it is important to assess when the sampled values θ_t have approximately converged to the stationary distribution $f(\theta \mid \mathbf{x})$. This will depend on how well the MCMC algorithm is able to explore the state space and also on the correlation between the $\theta_t's$.

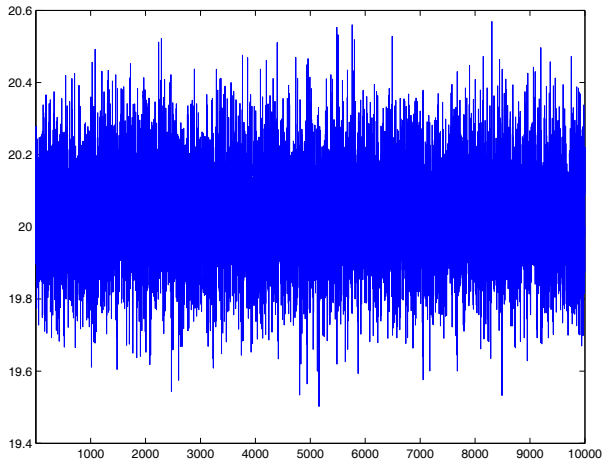
Also, we need to assess the convergence of MCMC averages, e.g. $\frac{1}{T}\sum_{t=1}^{T}\theta_t \rightarrow E[\theta \mid \mathbf{x}]$ and finally we need to be able to assess how close a given sample is to being independent and identically distributed.

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.

MCMC convergence assesment

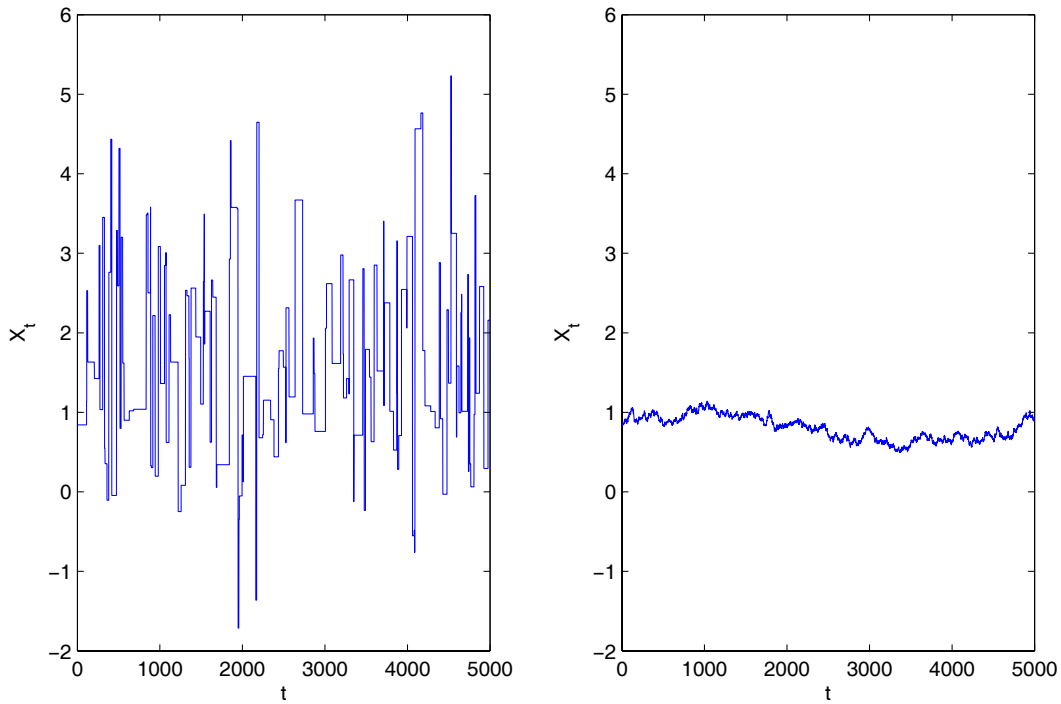
First, we can produce graphs of θ_t against t to show the mixing of the chain and any deviations from stationarity.

The following chain presents a good mixing performace.



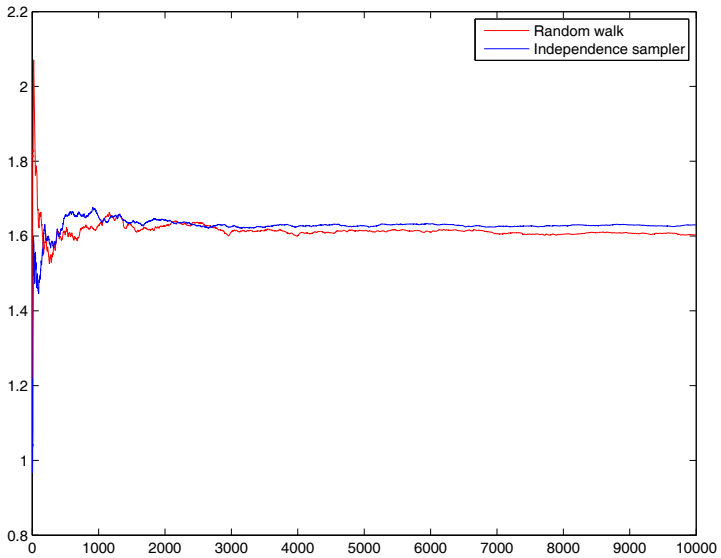
MCMC convergence assesment

The following diagram shows examples of badly mixing chains.



MCMC convergence assesment

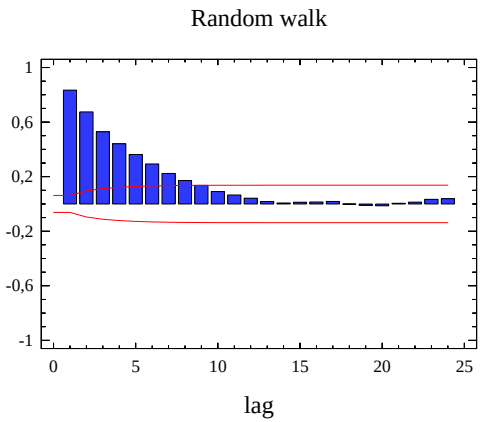
Secondly, we can plot running means of the parameters of interest to see when they have converged.

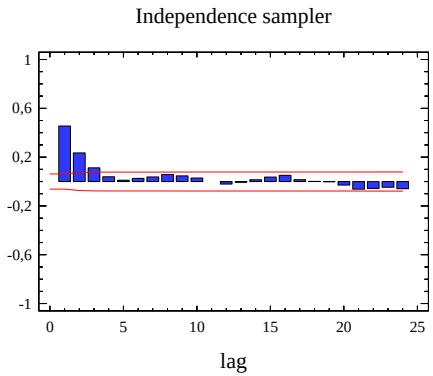


It can be seen that the means appear to have converged after about 3000 iterations. Thus, one possibility is to run the sampler for longer, using these initial iterations as a burn in period.

MCMC convergence assesment

Thirdly, we can plot the autocorrelation functions of the generated values. In general, as we are generating from a Markov chain, the successive values, θ_t, will be positively correlated.





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 4h datum, resp., which are now approximately independent.

Other algorithms: Particle filters

- MCMC algorithms are prohibitively costly when performing online estimation of latent variables and parameters, such as time series models.
- Alternatively, Sequential Monte Carlo methods, also known as particle filters, allow for on-line type inference by updating the posterior distribution as new data are observed.
- The basic idea consist in approximating the posterior distribution $f(\theta_i \mid x_1, \ldots, x_i)$ using a weighted set of J particles $\left\{\omega_i^{(j)}, \theta_i^{(j)}\right\}$, for $j = 1, \ldots, J$.
- These importance weights are updated sequentially as new observations arrive. Also, a resampling of the weigths is required to avoid that all but one of the importance weights are close to zero.
- Particle filter algorithms are especially suitable in dynamic models as the state-space models.