Markov Chain Monte Carlo

* Using the methods we have already seen we can easily generate from many univariate distributions.

* Another general technique which can be employed to generate observations from arbitrary distributions is Markov Chain Monte Carlo (MCMC).

* We will examine two of the most widely used MCMC methods.

* As the name suggests they are based on Markov chains so we will start by reviewing some of the basic properties of Markov chains.

Markov Chains

Definition 6

A Markov chain is a series of dependent random variables

$$X^{(0)}, X^{(1)}, \ldots, X^{(t)}, \ldots$$

such the conditional distribution of $X^{(t+1)}$ *given the previous observations in the sequence depends only on* $X^{(t)}$.

This conditional distribution is called the Transition Kernel or Markov Kernel of the chain and can be written as

$$X^{(t+1)} \mid X^{(0)}, X^{(1)}, \ldots, X^{(t)} \sim K(X^{(t)}, X^{(t+1)})$$

where for any fixed value $X^{(t)} = x$ *we have*

$$X^{(t+1)} \mid X^{(t)} = x \sim K(x, X^{(t+1)})$$

Stationary Distributions

Definition 7

A Markov chain is said to have a stationary distribution f if

$$X^{(t)} \sim f \quad \Rightarrow \quad X^{(t+1)} \sim f$$

* If K is the transition kernel of the Markov chain then the stationary distribution f satisfies

$$\int_{\mathcal{X}} K(x,y) f(x)\, dx \;=\; f(y)$$

Irreducible, Positive Recurrent Markov Chains

* For our purposes we need some conditions on the chain.

* For one thing we need to chain to be able to get from any set the sample space $\mathcal{X}$ to any other set in finite time. This is called irreducibility.

* Secondly we need the chain to return to any set with non-zero probability infinitely often in an infinite chain. This is called positive recurrence.

* The chains that we will examine are designed to have these two properties.

Convergence of Markov Chains

* If $X^{(1)}, X^{(2)}, \ldots,$ is positive recurrent and irreducible Markov chain then it can be shown that the marginal distribution of $X^{(t)}$ converges to the stationary distribution f as $t \to \infty$.

* This is a fundamental property which allows us to use a Markov chain to simulate at least approximately from the distribution f.

* We can start the Markov chain at an arbitrary point $x \in \mathcal{X}$ and allow it to run for a long time. Beyond some (unknonwn) burn-in period t_0 we will have that

$$X^{(t_0+1)}, X^{(t_0+2)}, \ldots$$

is a sequence of dependent random variables all with marginal distribution approximately equal to f.

The Ergodic Theorem

Theorem 11

Suppose that $X^{(0)}, X^{(1)}, \ldots, X^{(N)}$ *is a sequence of observations from a positive recurrent and irreducible Markov chain with stationary distribution* f *and suppose that* h *is a function such that* $\mathsf{E}_f[h(X)]$ *exists then*

$$\frac{1}{N}\sum_{t=1}^{N} h\left(X^{(t)}\right) \xrightarrow{p} \mathsf{E}_f[h(X)]$$

* This is basically the Weak Law of Large Numbers applied to a dependent sequence of random variables.

* Even though the Ergodic Theorem applies as stated above we will usually use the form that says that for any t_0

$$\frac{1}{N}\sum_{t=1}^{N} h\left(X^{(t_0+t)}\right) \xrightarrow{p} \mathsf{E}_f[h(X)]$$

The Metropolis–Hastings Algorithm

* Suppose that we want to simulate from a distribution f.

* Select a candidate family of distributions $q(y \mid x)$ which is easy to simulate from.

* Then set up the Markov chain as follows such that $X^{(t+1)}$ is found by
  1. Given $X^{(t)} = x$ simulate $Y_t = y$ from $q(y \mid x)$.
  2. Compute

$$\rho(x, y) \;=\; \min\left\{\frac{f(y)}{f(x)} \times \frac{q(x \mid y)}{q(y \mid x)},\; 1\right\}$$

  3. If $\rho(x, y) < 1$ then generate $U_t \sim \text{Uniform}(0, 1)$.
  4. Set

$$X^{(t+1)} \;=\; \begin{cases} y & \text{if } \rho(x, y) = 1 \text{ or } U_t < \rho(x, y) \\ x & \text{if } U_t \geqslant \rho(x, Y_t) \end{cases}$$

Acceptance Probabilities

* At any given iteration the Metropolis–Hastings acceptance probability is the quantity $\rho(x, y)$.

* The overall acceptance probability is the expected value of $\rho(x, y)$

$$\overline{\rho} \;=\; \int\int \rho(x,y) f(x) q(y \mid x)\, dy\, dx$$

* We can estimate this quantity using the Metropolis–Hastings acceptance probabilities and the Ergodic Theorem

$$\frac{1}{N} \sum_{t=0}^{N} \rho(X^{(t)}, Y_t) \;\xrightarrow{p}\; \overline{\rho}$$

Independence Metropolis–Hastings

* One method of generating the candidate variables is to do so with no reference to the current state of the chain

$$q(y \mid x) \;=\; g(y)$$

* In this case the acceptance probability at each iteration becomes

$$\begin{aligned} \rho(x, y) &= \min\left\{ \frac{f(y)}{f(x)} \times \frac{g(x)}{g(y)},\; 1 \right\} \\ &= \min\left\{ \frac{f(y)}{g(y)} \times \frac{g(x)}{f(x)},\; 1 \right\} \end{aligned}$$

Choice of Candidate Density

* For this to work well, the choice of g is quite similar to the choice of proposal distribution in the Accept-Reject algorithm.

* In particular we require that g has the same support as f.

* It is not essential that $f(x)/g(x)$ be bounded but the algorithm will not work well if it is not.

Independence Metropolis–Hastings and Accept-Reject Sampling

* For the same pair of target and candidate densities (f, g), the acceptance rate of the Metropolis–Hastings algorithm is higher than for the Accept-Reject algorithm.

* However, the resulting sequence of values from Metropolis–Hastings is a dependent sequence whereas the output of the Accept-Reject algorithm is an iid sample.

* The Metropolis–Hastings algorithm will generally result in many ties since if a candidate is rejected, the previous value is repeated in the chain.

* It is necessary to find $M \geqslant \sup f(x)/g(x)$ for the Accept-Reject algorithm but this is not necessary in the Metropolis–Hastings algorithm.

Independence Metropolis–Hastings

* In practice the Independence Metropolis–Hastings algorithm is very sensitive to the choice of g.

* Having a bad candidate density can result in the chain becoming stuck at a point x with $f(x) \gg g(x)$ for long periods of time.

* This is particularly prone to occur if the variability under the candidate distribution g is less than that under the target distribution f.

* We should choose a candidate g as similar to the target as possible and it is best to try to ensure that the variability of the candidate distribution is at least as large as the target.

* For complex high dimensional problems commonly encountered in MCMC, this can be very hard to do.

Random Walk Metropolis–Hastings

* An alternative to the Independence Metropolis–Hastings has the candidate density centred at the current value.

* In this way the chain will move more frequently although usually by smaller amounts.

* A random walk is defined by $Y_t = X^{(t)} + \varepsilon_t$ where $\varepsilon_t \sim g$.

* This results in the proposal kernel $q(y \mid x) = g(y - x)$.

* The choice of g is not so critical here since we are essentially exploring the target density locally.

* We still need to ensure that we run the chain long enough that the entire support of f is explored.

Random Walk Metropolis–Hastings

* It is very common to have g symmetric about 0.

* In that case $q(y \mid x) = g(y - x) = g(x - y) = q(x \mid y)$ and so the Metropolis–Hastings acceptance probability becomes

$$\rho(x, y) = \min\left\{\frac{f(y)}{f(x)}, 1\right\}$$

* This means that if the generated candidate Y_t is in a region with higher probability under f then we are guaranteed to move to it, but there is still a non-zero probability that we will move to regions with lower probability.

Random Walk Metropolis–Hastings

* Common choices of g are the Uniform$[-\delta, \delta]$ distribution or the Normal$(0, \sigma^2)$ distribution.

* In either case, the choice of scale (δ or σ^2) is crucial.

* If the scale is too small, then convergence will be very slow since the candidate will always be close to the current value of the chain.

* If the scale is too large, then the random walk algorithm becomes more like the independence algorithm and so the acceptance probability can be badly reduced and the chain can get stuck for long periods.

The Gibbs Sampler

* The Gibbs Sampler is designed to simulate from multivariate distributions based on simulations from univariate distributions.

* Unlike the Metropolis–Hastings algorithm the Gibbs Sampler is guaranteed to move at every iteration.

* Iterations are actually made up of a number of stages, typically as many stages as there are components in the random vector being considered.

* For simplicity we shall start by considering the two-stage version for a bivariate random vector.

The Two Stage Gibbs Sampler

* Suppose that (X, Y) is a bivariate random vector with joint density $f(x, y)$.

* The conditional distributions are then given by

$$f_{Y|X}(y \mid x) = \frac{f(x,y)}{\int f(x,y)\, dy} \qquad f_{X|Y}(x \mid y) = \frac{f(x,y)}{\int f(x,y)\, dx}$$

* In many situations, these conditional distributions are relatively easy to find and to simulate from.

* The two stage Gibbs sampler constructs a bivariate Markov Chain by alternately generating observations from these two univariate densities.

The Two Stage Gibbs Sampler

1. Initialise the chain at $X^{(0)}$.

2. At iteration $t = 1, 2, \ldots$

2.1 Generate $Y^{(t)}$ from $f_{Y|X}(y \mid X^{(t-1)})$.

2.2 Generate $X^{(t)}$ from $f_{X|Y}(x \mid Y^{(t)})$.

The Multi-Stage Gibbs Sampler

* The general multi-stage Gibbs Sampler is a natural extension to the 2-stage situation.

* Suppose that $\boldsymbol{X} = (X_1, \ldots, X_d)$ where the X_i are univariate.

* Define the p full conditional densities

$$f_i(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d) \;\propto\; f(\boldsymbol{x})$$

The Multi-Stage Gibbs Sampler

* The multi-stage Gibbs sampler is then
  1. Initialise the chain at $\boldsymbol{X}^{(0)} = (X_1^{(0)}, \ldots, X_d^{(0)})$.
  2. At each iteration $t = 1, 2, \ldots$

     2.1 Generate $X_1^{(t)}$ from $f_1(x_1 \mid X_2^{(t-1)}, \ldots, X_d^{(t-1)})$.

     2.2 Generate $X_2^{(t)}$ from $f_2(x_2 \mid X_1^{(t)}, X_3^{(t-1)}, \ldots, X_d^{(t-1)})$.

     $\vdots$

     2.i Generate $X_i^{(t)}$ from $f_i(x_i \mid X_1^{(t)}, \ldots, X_{i-1}^{(t)}, X_{i+1}^{(t-1)}, \ldots, X_d^{(t-1)})$.

     $\vdots$

     2.d Generate $X_d^{(t)}$ from $f_d(x_d \mid X_1^{(t)}, \ldots, X_{d-1}^{(t)})$.

Monitoring Convergence of MCMC

* It is very important to examine the output of the Markov chain to assess if convergence to the stationary distribution and/or ergodic convergence of estimates has occurred.

* The R package `coda` has a number of functions to do this.

* Plots of the components of the vector against iteration number are one important graphical diagnostic.

* Plotting the estimator of a function of interest against the iteration number can also be useful in examining ergodic convergence.

Multiple Chains

* One drawback of many methods for convergence checking is that we can often see "false convergence".

* This commonly occurs when a chain gets "stuck" in a region of the sample space and does not properly move over the space.

* For this reason it is usually suggested that multiple chains are run from different starting points.

* It is commonly suggested that the starting points for the chains come from a distribution which is over-dispersed relative to the target distribution.

* If multiple chains, run for the same length of time, display different behaviours then convergence is unlikely to have occurred.