A Markov chain Monte Carlo example

Summer School in Astrostatistics, Center for Astrostatistics, Penn State University
Murali Haran, Dept. of Statistics, Penn State University

This module works through an example of the use of Markov chain Monte Carlo for drawing samples from a multidimensional distribution and estimating expectations with respect to this distribution. The algorithms used to draw the samples is generally refered to as the Metropolis-Hastings algorithm of which the Gibbs sampler is a special case. We describe a model that is easy to specify but requires samples from a relatively complicated distribution for which classical Monte Carlo sampling methods are impractical. We describe how to implement a Markov chain Monte Carlo (MCMC) algorithm for this example.

The purpose of this is twofold: First to illustrate how MCMC algorithms are easy to implement (at least in principle) in situations where classical Monte Carlo methods do not work and second to provide a glimpse of practical MCMC implementation issues. It is difficult to work through a truly complex example of a Metropolis-Hastings algorithm in a short tutorial. Our example is therefore necessarily simple but working through it should provide a beginning MCMC user a taste for how to implement an MCMC procedure for a problem where classical Monte Carlo methods are unusable.

Datasets and other files used in this tutorial:

- COUP551_rates.dat
- MCMCchpt.R
- batchmeans.R

pdf files referred to in this tutorial that give technical details:

- chptmodel.pdf
- fullcond.pdf
- chptmodel2.pdf
- fullcond2.pdf

Introduction

Monte Carlo methods are a collection of techniques that use pseudo-random (computer simulated) values to estimate solutions to mathematical problems. In this tutorial, we will focus on using Monte Carlo for Bayesian inference. In particular, we will use it for the evaluation of expectations with respect to a probability distribution. Monte Carlo methods can also be used for a variety of other purposes, including estimating maxima or minima of functions (as in likelihood-based inference) but we will not discuss these here.

Monte Carlo works as follows: Suppose we want to estimate an expectation of a function g(x) with respect to the probability distribution f. We denote this desired quantity $m = E_f g(x)$. Often, m is analytically intractable (the integration or summation required is too complicated). A Monte Carlo estimate of m is obtained by simulating N pseudo-random values from the distribution f, say $X_1, X_2, .., X_N$ and simply taking the average of $g(X_1), g(X_2), .., g(X_N)$ to estimate m. As N (number of samples) gets large, the estimate converges to the true expectation m.
A toy example to calculate the $P(-1 < X < 0)$ when X is a Normal(0,1) random variable:
xs = rnorm(10000) # simulate 10,000 draws from N(0,1)

```
xcount = sum((xs>-1) & (xs<0)) # count number of draws between -1 and 0
xcount/10000 # Monte Carlo estimate of probability
pnorm(0)-pnorm(-1) # Compare it to R's answer (cdf at 0) - (cdf at -1)
```

Importance sampling: Another powerful technique for estimating expectations is importance sampling where we produce draws from a different distribution, say q, and compute a specific weighted average of these draws to obtain estimates of expectations with respect to f. In this case, A Monte Carlo estimate of m is obtained by simulating N pseudo-random values from the distribution q, say $Y_1,Y_2,..,Y_N$ and simply taking the average of $g(Y_1)w(Y_1)$,$g(Y_2)w(Y_1),..,g(Y_N)w(Y_1)$ to estimate m, where $W_1,W_2,..,W_N$ are weights obtained as follows: $W_i = f(Y_1)/q(Y_1)$. As N (number of samples) gets large, the estimate converges to the true expectation m. Often, when normalizing constants for f or q are unknown, and for numerical stability, the weights are `normalized' by dividing the above weights by the sum of all weights (sum over $W_1,..,W_N$).

Importance sampling is powerful in a number of situations, including:
(i) When expectations with respect to several different distributions (say $f_1,..,f_p$) are of interest. All these expectations can, in principle, be estimated by using just a single set of samples!
(ii) When rare event probabilities are of interest so ordinary Monte Carlo would take a huge number of samples for accurate estimates. In such cases, selecting q appropriately can produce much more accurate estimates with far fewer samples.

Discussions of importance sampling in astronomical Bayesian computation appear in Lewis & Bridle and Trotta for cosmological parameter estimation and Ford for extrasolar planet modeling.

R has random number generators for most standard distributions and there are many more general algorithms (such as rejection sampling) for producing independent and identically distributed (i.i.d.) draws from f. Another, very general approach for producing non i.i.d. draws (approximately) from f is the Metropolis-Hastings algorithm.

Markov chain Monte Carlo : For complicated distributions, producing pseudo-random i.i.d. draws from f is often infeasible. In such cases, the Metropolis-Hastings algorithm is used to produce a Markov chain say $X_1,X_2,..,X_N$ where the X_i's are *dependent* draws that are *approximately* from the desired distribution. As before, the average of $g(X_1),g(X_2),..,g(X_N)$ is an estimate that converges to m as N gets large. The Metropolis-Hastings algorithm is very general and hence very useful. In the following example we will see how it can be used for inference for a model/problem where it would otherwise be impossible to compute desired expectations.

Problem and model description

First, a five minute review of Bayesian inference
We begin by specifying a probability model for our data Y by assuming it is generated from some distribution h(theta;Y), where theta is a set of parameters for that distribution. This is written Y~h(theta). We want to infer theta from the fixed, observed dataset Y. First, consider likelihood inference. We find a value of theta where the likelihood L(theta;Y) (which is obtained from the probability distribution h(theta;Y)) is maximized; this is the maximum likelihood estimate (MLE) for theta. Now consider Bayesian inference. We assume a prior distribution for theta, p(theta), based on our previous knowledge. This prior may be based on astrophysical insights (e.g. no source can have negative brightness), past astronomical observation (e.g. stars have masses between 0.08-150 solar masses), and/or statistical considerations (e.g. uniform or Jeffreys priors) when it is difficult to obtain

good prior information. Inference is based on the posterior distribution Pi(thetalY) which is proportional to the product of the likelihood and the prior. It is only *proportional* to this product because in reality Bayes theory requires that we write down a denominator (the integral of the product of the likelihood and prior over the parameter space). Fortunately, Markov chain Monte Carlo algorithms avoid computation of this denominator while still producing samples from the posterior Pi(thetal Y). Note that the MCMC methods discussed here are often associated with Bayesian computation, but are really independent methods which can be used for a variety of challenging numerical problems. Essentially, any time samples from a complicated distribution are needed, MCMC may be useful.

Our example uses a dataset from the Chandra Orion Ultradeep Project (COUP). This is a time series of X-ray emission from a flaring young star in the Orion Nebula Cluster. More information on this is available at: CASt Chandra Flares data set . The raw data, which arrives approximately according to a Poisson process, gives the individual photon arrival times (in seconds) and their energies (in keV). The processed data we consider here is obtained by grouping the events into evenly-spaced time bins (10,000 seconds width).

Our goal for this data analysis is to identify the change point and estimate the intensities of the Poisson process before and after the change point. We describe a Bayesian model for this change point problem (Carlin and Louis, 2000). Let Yt be the number of occurrences of some event at time t. The process is observed for times 1 through n and we assume that there is a change at time k, i.e., after time k, the event counts are significantly different (higher or lower than before). The mathematical description of the model is provided in change point model (pdf) . While this is a simple model, it is adequate for illustrating some basic principles for constructing an MCMC algorithm.

We first read in the data:

```
chptdat = read.table("http://www.stat.psu.edu/~mharan/MCMCtut
/COUP551_rates.dat",skip=1)
```

Note: This data set is just a convenient subset of the actual data set (see reference below.)

We can begin with a simple time series plot as exploratory analysis.

```
Y=chptdat[,2] # store data in Y
ts.plot(Y,main="Time series plot of change point data")
```

The plot suggests that the change point may be around 10.

Setting up the MCMC algorithm

Our goal is to simulate multiple draws from the posterior distribution which is a multidimensional distribution known only upto a (normalizing) constant. From this multidimensional distribution, we can easily derive the conditional distribution of each of the individual parameters (one dimension at a time). This is described, along with a description of the Metropolis-Hastings