Bayesian Statistics

3: Simulation-Based Computation

David Draper

Department of Applied Mathematics and Statistics
University of California, Santa Cruz

draper@ams.ucsc.edu

http://www.ams.ucsc.edu/~draper

© 2003 David Draper (all rights reserved)
Contents

3.1 Introduction to Markov chain Monte Carlo (MCMC) methods

3.2 IID sampling; rejection sampling

3.3 Markov chains

3.4 The Metropolis-Hastings algorithm

3.5 Gibbs sampling. Case Study: the NB10 data revisited

3.6 Model expansion. User-friendly implementation of Gibbs sampling via BUGS and WinBUGS

3.7 MCMC implementation strategies

3.8 Practical monitoring and convergence diagnostics
   3.8.1 Geweke equilibrium diagnostic
   3.8.2 Gelman-Rubin diagnostic for finding multiple modes
   3.8.3 Raftery-Lewis diagnostic for estimating length of monitoring run
   3.8.4 Heidelberger-Welch diagnostic for testing stationarity

3.9 Bayesian model uncertainty

3.10 References
3.1 Introduction to Markov Chain Monte Carlo (MCMC) methods

Computation via conjugate analysis (part 2 of the lecture notes) produces **closed-form results** (good) but is **limited in scope** to a fairly small set of models for which straightforward conjugate results are possible (bad).

This was a **severe limitation** for Bayesians for almost 250 years (from the 1750s to the 1980s).

Over the past 10 years the Bayesian community has "discovered" and developed an entirely new computing method, **Markov chain Monte Carlo (MCMC)** ("discovered" because the physicists first figured it out about 50 years ago: Metropolis and Ulam, 1949; Metropolis et al., 1953).

We've seen that the **central Bayesian practical challenge** is the **computation of high-dimensional integrals**.

People working on the first atom bomb in World War II faced a **similar challenge**, and noticed that **digital computers** (which were then passing from theory (Turing 1943) to reality) offered an **entirely new approach** to solving the problem.

The idea (Metropolis and Ulam, 1949) was based on the observation that **anything you want to know about a probability distribution** can be learned to arbitrary accuracy by **sampling from it**.

Suppose, for example, that you’re interested in a posterior distribution $p(\theta|y)$ which **cannot be worked with (easily) in closed form**, and initially (to keep things simple) think of $\theta$ as a **scalar** (real number) rather than vector.
Simulation-Based Computation

Four things of direct interest to you about \( p(\theta|y) \) would be

- its **mean** \( \mu = E(\theta|y) \) and **standard deviation** \( \sigma = \sqrt{V(\theta|y)} \),

- its **shape** (basically you’d like to be able to trace out (an estimate of) the entire **density curve**), and

- one or more of its **quantiles** (e.g., to construct a 95% central posterior interval for \( \theta \) you need to know the **2.5% and 97.5% quantiles**, and sometimes the **posterior median** (the **50th percentile**) is of interest too).

Suppose you could take an **arbitrarily large random sample** from \( p(\theta|y) \), say \( \theta_1^*, \ldots, \theta_m^* \).

Then each of the above four aspects of \( p(\theta|y) \) can be **estimated** from the \( \theta^* \) sample:

- \( \hat{E}(\theta|y) = \bar{\theta}^* = \frac{1}{m} \sum_{j=1}^{m} \theta_j^* \),

- \( \sqrt{V(\theta|y)} = \sqrt{\frac{1}{m-1} \sum_{j=1}^{m} (\theta_j^* - \bar{\theta}^*)^2} \),

- the density curve can be estimated by a **histogram** or **kernel density estimate**, and

- percentiles can be estimated by **counting** how many of the \( \theta^* \) values fall below a series of specified points—e.g., to find an estimate of the 2.5% quantile you solve the equation

\[
\hat{F}_\theta(t) = \frac{1}{m} \sum_{j=1}^{m} I(\theta_j^* \leq t) = 0.025 \tag{1}
\]

for \( t \), where \( I(A) \) is the **indicator function** (1 if \( A \) is true, otherwise 0).
3.2 IID Sampling; Rejection Sampling

These are called Monte Carlo estimates of the true summaries of \( p(\theta|y) \) because they are based on the controlled use of chance.

Theory shows that with large enough \( m \), each of the Monte Carlo (or simulation-based) estimates can be made arbitrarily close to the truth with arbitrarily high probability, under some reasonable assumptions about the nature of the random sampling.

One way to achieve this, of course, is to make the sampling IID (this is sufficient but not necessary—see below).

If, for example, \( \bar{\theta}^* = \frac{1}{m} \sum_{j=1}^{m} \theta_j^* \) is based on an IID sample of size \( m \) from \( p(\theta|y) \), we can use the frequentist fact that in repeated sampling \( V(\bar{\theta}^*) = \frac{\sigma^2}{m} \), where (as above) \( \sigma^2 \) is the variance of \( p(\theta|y) \), to construct a Monte Carlo standard error (MCSE) for \( \bar{\theta}^* \):

\[
\hat{SE}(\bar{\theta}^*) = \frac{\hat{\sigma}}{\sqrt{m}},
\]

where \( \hat{\sigma} \) is the sample SD of the \( \theta^* \) values.

This can be used, possibly after some preliminary experimentation, to decide on \( m \), the Monte Carlo sample size, which later we'll call the length of the monitoring run.

**An IID example.** Consider the posterior distribution \( p(\lambda|y) = \Gamma(29.001, 14.001) \) in the LOS example in part 2.

We already know that the posterior mean of \( \lambda \) in this example is \( \frac{29.001}{14.001} \approx 2.071 \); let's see how well the Monte Carlo method does in estimating this known truth.
IID Example (continued)

Here's an R function to construct Monte Carlo estimates of the posterior mean and MCSE values for these estimates.

```r
gamma.sim <- function( m, alpha, beta, n.sim, seed ) {

    set.seed( seed )

    theta.out <- matrix( 0, n.sim, 2 )

    for ( i in 1:n.sim ) {

        theta.sample <- rgamma( m, alpha, 1 / beta )

        theta.out[ i, 1 ] <- mean( theta.sample )

        theta.out[ i, 2 ] <- sqrt( var( theta.sample ) / m )

    }

    return( theta.out )

}

This function simulates, n.sim times, the process of taking an IID sample of size \( m \) from the \( \Gamma(\alpha,\beta) \) distribution and calculating \( \bar{\theta}^* \) and \( SE(\bar{\theta}^*) \).

rosalind 296> R

R : Copyright 2001, The R Development Core Team
Version 1.2.1 (2001-01-15)

> m <- 1000

> alpha <- 29.001

> beta <- 14.001

> n.sim <- 500

> seed <- c( 6425451, 9626954 )
```
IID Example (continued)

> theta.out <- gamma.sim( m, alpha, beta, n.sim, seed )

# This took about 1 second at 450 Unix MHz.

> theta.out[ 1:10, ]

   [,1]        [,2]
[1,] 2.082105 0.01166379
[2,] 2.072183 0.01200723
[3,] 2.066756 0.01247277
[4,] 2.060785 0.01200449
[5,] 2.078591 0.01212440
[6,] 2.050640 0.01228875
[7,] 2.071706 0.01182579
[8,] 2.063158 0.01176577
[9,] 2.058440 0.01186379
[10,] 2.068976 0.01220723

The \( \bar{\theta}^* \) values fluctuate around the truth with a give-or-take
of about 0.01, which agrees well with the theoretical SE
\[
\frac{\sigma}{\sqrt{m}} = \frac{\sqrt{\alpha}}{\beta \sqrt{m}} \approx 0.01216
\]
(recall that the variance of a Gamma
distribution is \( \frac{\alpha}{\beta^2} \)).

> postscript( "gamma-sim1.ps" )

> theta.bar <- theta.out[, 1]

> qqnorm( ( theta.bar - mean( theta.bar ) ) /
            sqrt( var( theta.bar ) ) )

> abline( 0, 1 )

> dev.off( )

null device
  1

Each of the \( \bar{\theta}^* \) values is the mean of \( m = 1,000 \) IID draws,
so (by the CLT) the distribution of the random variable \( \bar{\theta}^* \)
should be closely approximated by a Gaussian.
> truth <- alpha / beta

> theta.bar.SE <- theta.out[, 2]

> qnorm( 0.025 )
[1] -1.959964

> sum( ( theta.bar - 1.96 * theta.bar.SE < truth ) *
    ( truth < theta.bar + 1.96 * theta.bar.SE ) ) / n.sim
[1] 0.972

Thus we can use frequentist ideas to work out how big $m$
needs to be to have any desired Monte Carlo accuracy for
$\tilde{\theta}^*$ as an estimate of the posterior mean $E(\theta|y)$.

In practice, with $p(\theta|y)$ unknown, you would probably take an
initial sample (of size $m = 1,000$, say) and look at the
MCSE to decide how big $m$ really needs to be.
IID Example (continued)

\[ \text{\texttt{theta.bar <- gamma.sim( m, alpha, beta, 1, seed )}} \]

\[ \text{\texttt{theta.bar}} \]

\[ \begin{bmatrix}
[1,] & [2,] \\
1 & 2.082105 & 0.01166379 \\
\end{bmatrix} \]

(1) Suppose you wanted the MCSE of \( \bar{\theta}^* \) to be (say) \( \epsilon = 0.001 \). Then you could solve the equation

\[ \frac{\hat{\sigma}}{\sqrt{m}} = \epsilon \quad \Leftrightarrow \quad m = \frac{\sigma^2}{\epsilon^2}, \]  \tag{3}

which says (unhappily) that the required \( m \) goes up as the square of the posterior SD and as the inverse square of \( \epsilon \).

The previous calculation shows that \( \frac{\hat{\sigma}}{\sqrt{1000}} \equiv 0.01166379 \), from which \( \hat{\sigma} \equiv 0.3688414 \), meaning that to get \( \epsilon = 0.001 \) you need a sample of size \( \frac{0.3688414^2}{0.001^2} \equiv 136,044 \equiv 136k \) (!).

(2) Suppose instead that you wanted \( \bar{\theta}^* \) to differ from the true posterior mean \( \mu \) by no more than \( \epsilon_1 \) with Monte Carlo probability at least \( 1 - \epsilon_2 \):

\[ P(| \bar{\theta}^* - \mu | \leq \epsilon_1 ) \geq 1 - \epsilon_2, \]  \tag{4}

where \( P(\cdot) \) here is based on the (frequentist) Monte Carlo randomness inherent in \( \bar{\theta}^* \).

We know from the CLT and the calculations above that in repeated sampling \( \bar{\theta}^* \) is approximately normal with mean \( \mu \) and variance \( \frac{\sigma^2}{m} \); this leads to the inequality

\[ m \geq \frac{\sigma^2 [ \Phi^{-1}(1 - \frac{\epsilon_2}{2})]^2}{\epsilon_1^2}, \]  \tag{5}

where \( \Phi^{-1}(q) \) is the place on the standard normal curve where \( 100q\% \) of the area is to the left of that place (the \( q \)th quantile of the standard normal distribution).
A Closer Look at IID Sampling

(5) is like (3) except that the value of $m$ from (3) has to be multiplied by $\left[\Phi^{-1}(1 - \frac{\epsilon_2}{2})\right]^2$, which typically makes the required sample sizes even bigger.

For example, with $\epsilon_1 = 0.001$ and $\epsilon_2 = 0.05$—i.e., to have at least 95% Monte Carlo confidence that reporting the posterior mean as 2.071 will be correct to about four significant figures—(5) says that you would need a monitoring run of at least $136,044(1.959964)^2 \div 522,608 \approx 523k$ (!).

(On the other hand, this sounds like a long monitoring run but only takes about 2.5 seconds at 500 Unix MHz on a SunBlade 100, yielding $[\bar{\theta}^*, \hat{SE}(\bar{\theta}^*)] = (2.0709, 0.00053)$.)

It’s evident from calculations like these that people often report simulation-based answers with numbers of significant figures far in excess of what is justified by the actual accuracy of the Monte Carlo estimates.

A Closer Look at IID Sampling. I was able to easily perform the above simulation study because R has a large variety of built-in functions like rgamma for pseudo-random-number generation.

How would you go about writing such functions yourself?

There are a number of general-purpose methods for generating random numbers (I won’t attempt a survey here); the one we need to look closely at, to understand the algorithms that arise later in this section, is rejection sampling (von Neumann 1951).