STAT40380/STAT40390/STAT40850 Bayesian Analysis

Dr Niamh Russell

School of Mathematics and Statistics
University College Dublin

niamh.russell@ucd.ie

April 2016

Simulating from the posterior distribution

- Once we have set up our MCMC we can iterate to obtain samples which will eventually be representative of the posterior distribution
- However there might be many reasons why the samples we take may be unrepresentative of the posterior:
  - We might have chosen poor starting values which strongly influence the bahaviour of the chain
  - There may be within-chain correlations due to the Markove nature of our sampling. We would ideally like to obtain independent samples from the posterior
- Once we have representative samples from the posterior (ie the chain has *converged*) we then need to run the MCMC to obtain enough samples for analysis

Tips for a productive MCMC algorithm

- A good MCMC chain will converge quickly to the posterior distribution
- To achieve efficient convergence we might try the following:
  - Write our models so that between-parameter correlation is kept at a minimum
  - Check that we are not including unnecessary terms in our complete conditional distributions
  - Monitor the chains to see how well they are performing
  - Change the algorithm (eg the proposal distribution) if the number of acceptances is too low
- An efficient MCMC will save a lot of time and effort when dealing with large problems

Discarding iterations

- We often remove a proportion of the earlier iterations to avoid dependence on our starting values. This period is known as the *burn-in* period
- Similarly, we may only keep every *k*th draw from the MCMC and discard the others. We call this process *thinning*. Thinning the iterations also solves some computer storage problems
- We will run our simulations with *multiple, over-dispersed starting values* to check that the sequences are mixing well (ie exploring all of the posterior distribution)
- We can run into severe problems with MCMC chains and convergence when the posterior distributions are multi-modal

Monitoring convergence

- Consider a set of m parallel chains for some parameter θ, each of length n
- Write θ_{ij} as the value of θ for iteration i on chain j
- Compute the *between* and *within-chain* variances

$$B = \frac{n}{m-1} \sum_{j=1}^{m} (\bar{\theta}_{.j} - \bar{\theta}_{..})^2$$

$$W = \frac{1}{m} \sum_{j=1}^{m} s_j^2 \text{ where } s_j^2 = \frac{1}{n-1} \sum_{i=1}^{n} (\theta_{ij} - \bar{\theta}_{.j}$$

with $\bar{\theta}_{.j} = \frac{1}{n} \sum_{i=1}^{n} \theta_{ij}$ and $\bar{\theta}_{..} = \frac{1}{m} \sum_{j=1}^{m} \bar{\theta}_{.j}$

- This formulation is very similar to the separation of variances approach found in the one-way ANOVA model

Monitoring convergence 2

- Now estimate the posterior variance of the parameter by a weighted average of B and W:

$$\widehat{var}^{+}(\theta|\mathbf{x}) = \frac{n-1}{n}W + \frac{1}{n}B$$

- Define the *Brooks-Gelman-Rubin* (BGR) diagnostic (also known as the scale reduction factor) as:

$$\hat{R} = \sqrt{\frac{\widehat{var}^{+}(\theta|\mathbf{x})}{W}}$$

- $\hat{R} \to 1^{+}$ as $n \to \infty$. If $\hat{R}$ is far from 1 then we have not sampled enough values from our chain: we should increase n

Some notes about $\hat{R}$

- The quantity $\widehat{var}^{+}(\theta|\mathbf{x})$ is an over-estimate of the posterior variance of θ when the starting values are over-dispersed (ie are more variable than the posterior)
- As n gets larger the contribution of the between chain variance 'should' be negligible and so $\widehat{var}^{+}(\theta|\mathbf{x})/W$ should tend to 1
- For a satssfactory convergence, $\hat{R}$ should be stable and near 1 for *all* of the parameters
- Similarly, W and B should also be stable
- In WinBUGS B is plotted in green, W in blue and $\hat{R}$ in red

How many iterations?

- Once we have satisfactorily removed a burn-in period and thinned the iterations suitably, we should aim to keep a large number of iterations for use in further analysis
- Suppose we estimate the sample mean $\bar{\theta}$ and standard deviation s_θ from a set of n samples. If the posterior distribution is approximately normal, then an appropriate estimate of the accuracy of the mean is $s_\theta/\sqrt{n}$
- This quantity represents the extra uncertainty introduced via our Monte Carlo sampling. We should aim to have the Monte Carlo error as small as possible, and much smaller than s_θ

How many iterations 2

- The total standard deviation of the parameter estimate is now $s_\theta \sqrt{1 + 1/n}$
- So if $n = 100$ the factor $\sqrt{1 + 1/n} = 1.005$, giving an impression that 100 iterations is a reaonable number to estimate the mean of this parameter
- For problems where the posterior is not approximately normally distributed, or where we are not interested in the mean, we may need many more samples

Making MH efficient

- Even when we are satisfied that our model is as well written as it could be we may still need to make many choices about our algorithms, eg the parameters of the proposal distribution
- In situations where the complete conditionals allow us to sample directly, this may still not be the most efficient algorithm. For example, the chain maight move slowly around the parameter space (known as *slow mixing*) and thus we may end up with strong autocorrelation
- When we must use the Metropolis or MH algorithm we must choose a proposal distribution. There are generally two classes:
  1. A *random walk proposal distribution*, eg $J(\theta^*|\theta^{t-1}) = N(\theta^{t-1}, a\sigma^2)$ where a is a fixed constant and σ^2 is the variance of the true posterior
  2. An *independence sampler proposal distribution*, eg $J(\theta^*|\theta^{t-1}) = N(a, b)$ where both a and b are fixed constants
- The independence sampler can be useful when we have good knowledge of the shape of the posterior distribution

Making MH efficient 2

- Some results exist for normal random walk jumping distributions (eg $J(\theta^*|\theta^{t-1}) = N(\theta^{t-1}, a\sigma^2)$) when the posterior distribution is approximately (multivariate) normal
- If a is set too large then we will not accept many values because we are moving too quickly around the posterior distribution
- If a is set too small then we will accept lots of values but only move very slowly around the posterior distribution In either case we will suffer from very high autocorrelation and will have to do a lot of thinning to get decent posterior samples
- IT can be shown that the optimal value of a for a problem with d parameters is around $2.4/\sqrt{d}$. This corresponds to an acceptance rate of around 0.44 in one dimension, falling to 0.23 in higher dimensions
- Often the algorithm will be run for a while before σ^2 is approximately estimated. We then run our algorithm with an adjusted $a\sigma^2$ variance

MCMC Tips and tricks

Some approximate steps to follow for large problems

1. Set up your MCMC algorithm using the standard complete conditional approach
2. Decide on a set of over-dispersed starting values
3. Run the model and monitor the chains, adjusting the proposal distributions where relevant
4. Check convergence using $\hat{R}$ and autocorrelation
5. Run the chains until the desired number of iterations are obtained

WinBUGS does all this for you!