Reversible Jump MCMC

Maggie Johnson
Iowa State University
majohnso@iastate.edu

Andee Kaplan
Iowa State University
ajkaplan@iastate.edu

Ian Mouzon
Iowa State University
imouzon@iastate.edu

Sam Tyner
Iowa State University
sctyner@iastate.edu

Abstract

Reversible jump MCMC is a mechanism designed to solve trans-dimensional problems, where the number of parameters one wishes to estimate, in addition to the parameter values themselves, is also unknown. RJMCMC has applications in variable selection, Bayesian model selection, multiple change-point problems, and many more. In this paper, we provide an introduction to RJMCMC, demonstrate its functionality in an example (or two?), discuss the problems that arise most frequently in RJMCMC, and present some extensions of the theory and application of RJMCMC.

1 Introduction

Reversible jump Markov Chain Monte Carlo sampling was developed to solve trans-dimensional problems. Trans-dimensional problems are those in which "the number of things you don't know is one of the things you don't know" (Green and Hastie 2009). These problems present great challenges to the usual statistical methods. With RJMCMC, however, you can jump from one parameter space to another without too much difficulty. Because it is also reversible, the probability of moving from state A to state B is the same as the probability of moving from state B to state A. This is important for jumping between parameter spaces because this reversibility is what gives the sampler its ability to explore the model space in its entirety.

2 Problem Structure

There are many different scenarios where using RJMCMC can be useful. In our example modeling soccer goals, we look at a choice between a Poisson model with one mean parameter and a Negative Binomial model with one mean parameter and one overdispersion parameter. The problem of choosing between two models with one and two parameters each is rather simplistic, and using RJMCMC is not necessary. The usefulness of RJMCMC sampling is emphasized in problems like the multiple change point problem given in Green (1995). In that scenario, the number of change points and the change point locations, as well as the parameters of the various possible change point models, are all unknown. In problems like that, RJMCMC is extremely useful and powerful, because selection of the number of parameters is done alongside the selection of the parameter values themselves.

2.1 Problem Set Up

As in all MCMC problems, there is a target distribution, which is denoted $\pi(\boldsymbol{\theta})$ below, and an aperiodic and irreducible transition kernel, denoted $\alpha(\boldsymbol{\theta}, d\boldsymbol{\theta}')$ below. With these two things in mind, our chain must satisfy the *detailed balance condition*:

$$\int_{(\boldsymbol{\theta},\boldsymbol{\theta}')\in A\times B} \pi(dx)\alpha(\boldsymbol{\theta}, d\boldsymbol{\theta}') = \int_{(\boldsymbol{\theta},\boldsymbol{\theta}')\in A\times B} \pi(d\boldsymbol{\theta}')\alpha(\boldsymbol{\theta}', d\boldsymbol{\theta})$$

where $A, B \subset \times$ are all Borel sets in our model spaces. This is also not unusual for an MCMC sampler, but for the RJMCMC, it becomes more important because we have to get it exactly right in order to correctly move between the possible models in our model space.

In defining our transition kernel, which will from now on just be denoted as α, we need a proposal distribution, $g(\boldsymbol{u})$, and a deterministic function, $h(\boldsymbol{\theta}, \boldsymbol{u})$ to take us from one model space to another for each possible model. h is a bijection from the current model to the next model, and g is a probability distribution from which we generate a random vector, $\boldsymbol{u}$, that allows us to transition between model spaces via the deterministic function h.

In the outline of the general algorithm given in the next subsection, all possible models come from the set $\mathcal{M}$, where the size of $\mathcal{M}$ is countable, and the k^{th} model in $\mathcal{M}$ is denoted $\mathcal{M}_{\|}$. The parameter(s) for each model are written as $(k, \boldsymbol{\theta}_k)$ in order to keep track of the model to which each set of estimated parameter values belongs.

2.2 A General Algorithm

The following is a general algorithm for implementation of an RJMCMC sampler. If the current state of the chain is $(k, \boldsymbol{\theta}_k)$, then:

1. Propose a new model $\mathcal{M}_{k^*}$ with probability $j(k^*|k)$.
2. Generate $\boldsymbol{u}$ from a specified proposal density $g(\boldsymbol{u})$
3. Set $(\boldsymbol{\theta}^*_{k^*}, \boldsymbol{u}^*) = h'(\boldsymbol{\theta}_k, \boldsymbol{u})$ where h' is a bijection between $(\boldsymbol{\theta}_k, \boldsymbol{u})$ and $(\boldsymbol{\theta}^*_*, \boldsymbol{u}^*)$ where the following must hold: $dim(\boldsymbol{\theta}_k) + dim(\boldsymbol{u}) = dim(\boldsymbol{\theta}^*_{k^*}) + dim(\boldsymbol{u}^*)$.
4. Accept the proposed move to $(k^*, \boldsymbol{\theta}^*_{k^*})$ with probability

$$\alpha = \min\left\{1, \frac{\pi(\theta^*_{k^*})j(k^*|k)g'(\boldsymbol{u}^*)}{\pi(\boldsymbol{\theta}_k)j(k|k^*)g(\boldsymbol{u})} \left|\frac{\partial h'(\theta^*_{k^*}, \boldsymbol{u}^*)}{\partial(\boldsymbol{\theta}_k, \boldsymbol{u})}\right|\right\}$$

Where $u^* \sim g'$ (Chen, Ibrahim, and Shao 2000, 303)

3 Example: Soccer Data

4 Challenges of Implementation

Implementing reversible algorithms may seem difficult for several reasons and in fact it is the appearance of difficulty that is one of the First, much of the work being done on the topic is from the perspective of MCMC "experts," and so their writings can be incredibly dense and make for very time-consuming reads. Additionally, the language required to present these samplers is necessarily complex, which also adds to the difficulty of interpretation. These issues, however, are not the true cause of difficulty. In the practical sense, implementation is *actually* fairly easy. Only a very few steps require a detailed understanding of the underlying theoretical framework: the rest is fairly straightforward computation. Furthermore, there is little justification needed to guarantee a sampler is able to simulate from a target. So, what are the true challenges of implementing an RJMCMC sampler?

4.1 Efficiency

The main issue is usually not whether a proposal mechanism will work, but whether it will work *effeciently*. The more a sampler rejects moves, the longer it takes for it to successfully explore the target distribution's support and as a result the number of samples needed to achieve some given level of convergence. As a result, it is possible to have a sampler that will theorhetically behave like the target distribution and never have enough time to actually produce valid samples. In the case of a specific problem, there are several places in the proposal mechanism that benefit from careful scrutiny and tuning (Green and Hastie 2009, 12), a process that tends to be arduous and the final solution may not be widely applicable. The attempt to find better ways to handle such issues has lead to work developing useful general techniques for selecting parts of the mechanism with efficiency in mind.

Improving efficiency requires the proposed state $(k', \theta'_{k'})$ and the existitng state (k, θ_k) have similar state spaces. There are two main classes of methods for ensuring this:

1. *Order methods*: parameterizing proposals (our $g(u)$) for a given $h(\theta, u) = (\theta', u')$.
2. *Saturated state*: augment the state space $\mathcal{X}$ with auxiliary variables

4.1.1 Order Methods

For a given initial state θ_k in model k we can find an equivalent state $c_{k,k'}(\theta_k)$ in model k'. We call these equivalent states our "centering points." If we constrain $A((k, \theta_k), (k', c_{k,k'}(\theta_k)$ to be, say, 1, then moving from one model (k) to another (k') will be encouraged and the state space will be more thoroughly explored.

The *order* of the method determines the type of constraint imposed. For the 0^{th}-order, $A((k, \theta_k), (k', c_{k,k'}(\theta_k) = 1$, while for the k^{th}-order, $\nabla A((k, \theta_k), (k', c_{k,k'}(\theta_k) = \mathbf{0}$.

4.1.2 Saturated State Space

For a given state space $\mathcal{X}$, we can add additional "auxiliary" variables so that each model has the same number of parameters as the largest model. This means that changing between models with different numbers of parameters is equivalent to changing the values of the auxiliary variables. Using this method, between state changes become more likely, and so the sampler covers the set of possible proposals more quickly.

4.1.3 Other Ways to Improve Proposals

Another way to improve the proposal is through adaptive sampling: using past observations, even rejected ones, to make mid-run adjustments to the proposal. There are two types of adaptive sampling. The first is *diminishing* adaptive sampling. In diminishing adaptive sampling, there is continuous adaptation, but at a decreasing rate. The second type is adaptive sampling *through regeneration*. In this method of adaptive sampling, if regions of the state space exist where incoming chains are likely to be independent of outgoing chains, adapt as chains enter and leave them. Finally, there is delayed rejection: if proposal x' is rejected, try a backup proposal x'' instead.

4.2 Finding Appropriate Diagnostics

The main issue in improving efficiency is in promoting transitions between models, namely transitions between state space dimensions. However, when the dimension of the state space is large, it is difficult to imagine any single scalar-valued statistic that could work as a gatekeeper in a general sense. *Transitioning* between models is not always the favored choice. Chains may "stabilize" quickly inside a model, so that chain will provide good diagnostics *until* the chain moves to sample from a different model. At that point, the diagnostics become much trickier.

Recent work has been focused on accounting for the differences in "within" run and "between" run variability: finding ways account for how much disruption in chain behavior is natural when switching dimensions. This idea is similar to "within" model and "between" model variability to account for expected departure from modeled behavior.

5 Extensions & Alternatives

6 Conclusion

7 Code Appendix

References

Carlin, Bradley P, and Siddhartha Chib. 1995. "Bayesian Model Choice via Markov Chain Monte Carlo Methods." *Journal of the Royal Statistical Society. Series B (Methodological)*. JSTOR, 473–84.

Chen, Ming-Hui, Joseph G Ibrahim, and Qi-Man Shao. 2000. *Monte Carlo Methods in Bayesian Computation*. Springer.

Geman, Stuart, and Donald Geman. 1984. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 6. IEEE: 721–41.

Green, Peter J. 1995. "Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination." *Biometrika* 82 (4): 711–32.

———. 2003. "Trans-Dimensional Markov Chain Monte Carlo." *Oxford Statistical Science Series*. OXFORD UNIV PRESS, 179–98.

Green, Peter J, and David I Hastie. 2009. "Reversible Jump MCMC." *Genetics* 155 (3): 1391–1403.

Grenander, Ulf, and Michael I Miller. 1994. "Representations of Knowledge in Complex Systems." *Journal of the Royal Statistical Society. Series B (Methodological)*. JSTOR, 549–603.

Hastie, David. 2005. "Towards Automatic Reversible Jump Markov Chain Monte Carlo." PhD thesis, University of Bristol.

Jasra, Ajay, Arnaud Doucet, David A Stephens, and Christopher C Holmes. 2008. "Interacting Sequential Monte Carlo Samplers for Trans-Dimensional Simulation." *Computational Statistics & Data Analysis* 52 (4). Elsevier: 1765–91.

Stephens, Matthew. 2000. "Bayesian Analysis of Mixture Models with an Unknown Number of Components-an Alternative to Reversible Jump Methods." *Annals of Statistics*. JSTOR, 40–74.