Metropolis-Coupled Markov chain Monte Carlo

Matthew Darlington

January 16, 2020

1 Introduction

Markov chain Monte Carlo (MCMC) is a class of methods used to simulate drawing samples from a probability distribution. This is done by constructing a Markov chain whose stationary distribution $\pi(x)$ is that of our target probability distribution.

I have decided to focus my project on looking at the case where our target distribution is multimodal. In particular when we have high probability regions separated by low probability regions it is difficult to have the Markov chain mix around the state space.

I shall just be focusing in one dimension but all the methods can easily be scaled to higher dimensions. I picked just one dimension since this reduces the other factors that have to be taken into account.

2 Random Walk Metropolis (RWM)

The simplest version of MCMC is just to add some random noise onto our current location:

$$g\left(x'|x_t\right) = x_t + \mathcal{N}(0, h) \tag{1}$$

Where h is a tuning parameter we can use to vary the size of the steps we are taking. There is then a Metropolis acceptance step to decide if we take this new position or stay where we are. If h is too large then we will never accept our new location and if h is too small then we won't move enough to explore the state space well.

Algorithm 1: Metropolis-Hastings

```
   Input: iterations N, $x_{init}$
1  t = 0
2  while $t < N$ do
3      Generate a candidate state x' according to $g\left(x'|x_t\right)$
4      Calculate the acceptance probability $A(x', x) = \min\left(1, \frac{P(x')}{P(x_t)}\frac{g(x'|x_t)}{g(x'|x_t)}\right)$
5      Generate $u \sim U[0, 1]$
6      if $u \leq A(x', x_t)$ then
7          $x_{t+1} = x'$
8      end
9      else
10         $x_{t+1} = x_t$
11     end
12     $t = t + 1$
13 end
```

The problem we face with using RWM on a multimodal distribution is if h is too small we are likely to get stuck around one mode and if h is too large then we will have a low acceptance probability and take a long time to take our samples. However, we can take RWM and modify it to overcome this issue.

3 Metropolis-Coupled Markov Chain Monte Carlo (MCMCMC)

The idea behind MCMCMC is to find easier distributions to sample from and use these to generate better samples from our target distributions.

We construct a sequence $\pi_1, \pi_2, \dots \pi_m$ of target densities where $\pi_1 = \pi$ our true density and as τ increases the distributions get flatter. [CR14]τ is known as the temperature of the density and it is thought as of going from π_1 being the 'coldest' to π_m being the hottest. As the temperature increases our distributions get easier to sample from, but they get less representative of the target.

MCMCMC considers a state space Θ^m corresponding to all the chains each having their own position simultaneously. Thus at time t the state of our chain will be $\boldsymbol{\theta}_t = (\theta_{t,1}, \theta_{t,2}, \dots, \theta_{t,m})$.

We use RWM on each temperature by itself at each time point independently, but then also introduce a chance to "swap" two of the positions. This can be done with a Metropolis acceptance step given by:

$$\min\left(1, \frac{\pi_\tau(\theta_{t,\tau'})}{\pi_\tau(\theta_{t,\tau})}\frac{\pi_{\tau'}(\theta_{t,\tau})}{\pi_{\tau'}(\theta_{t,\tau'})}\right)$$

How we choose to select τ and τ' something that I briefly looked into during my report, and as far as I could see there was not much difference between suggested methods. The two most popular were either just picking completely at random or picking one at uniform, and swapping with an adjacent. My simulations showed very little between them so for simplicity I stuck with swapping at random.

Algorithm 2: Metropolis-Coupled MCMC

```
Input: iterations N, $x_{init}$, $\{\tau_1, ..., \tau_m\}$
1 t = 0
2 while $t < N$ do
3     for $i$ in $\{\tau_1, ..., \tau_m\}$ do
4         Perform a RWM step on $\pi_i$
5     end
6     Perform a Metropolis step to swap $\theta_{t,\tau}$ with $\theta_{t,\tau'}$
7     $t = t + 1$
8 end
```

4 Numerical Study

4.1 Normal Mixture

I started off by comparing the methods on the sum of normal distributions mentioned earlier:

$$\pi(x) = \Phi(x; -1, 1) + \Phi(x; 1, 1)$$
$$\pi_i(x) = \Phi(x; -1, i) + \Phi(x; 1, i)$$

I used $i = 1, 2, 4, 8$ as my temperatures as these seem to cover a good range of distributions. Using RWM on $i = 8$ is an easy distribution to take samples from, but this is very far from what our target distribution $i = 1$ is. Thus by using these an some intermediary distributions we can encourage good mixing between them and hopefully get some good results. The results can be found in Figure 1.

The graphs in order display a histogram of the samples obtained overlaid with the true probability density function, an auto-correlation plot with lags going up to 50 and a trace plot of the first 300 samples.

Here MCMCMC is vastly outperforming RWM. Although the histograms of samples don't look too dissimilar once we start looking at the dependence of samples it becomes obvious what is going on. In the RWM we are taking lots of samples from one mode and then moving to the other and repeating. However, by using MCMCMC we are able to now switch between them with a much higher frequency leading to low auto correlation. This is desirable since we want our samples to appear as being independent.

4.2 Heavy tailed

I then also wanted to test out the methods on more of a heavy tailed distribution. What I came up with as a good test function was the sum of two log-normal distributions, with one reflected in the y-axis to get a symmetrical two modal distribution.

$$\pi(x) = \log(\Phi(x; -1, 1)) + \log(\Phi(-x; 1, 1))$$
$$\pi_i(x) = \log(\Phi(x; -1, i)) + \log(\Phi(-x; 1, i))$$

I used the same temperatures $i = 1, 2, 4, 8$ and kept $h = 1$. The results can be found in figure 2. Again we get the same clear conclusion that MCMCMC is a large increase in results.

4.3 A more mixed normal mixture

Another question I had was how well the method would scale with the number of modes. To do this I tried using a mixture of five standard normal variables:

$$\pi(x) = \Phi(x; -4, 1) + \Phi(x; -2, 1) + \Phi(x; 0, 1) + \Phi(x; 2, 1) + \Phi(x; 4, 1)$$
$$\pi_i(x) = \Phi(x; -4, i) + \Phi(x; -2, i) + \Phi(x; 0, i) + \Phi(x; 2, i) + \Phi(x; 4, i)$$

The results can be found in figure 3. MCMCMC is still performing very well on this distributions with large jumps across multiple modes. This means we are getting better samples as this would be something we would expect if we were sampling straight from the distribution. RWM performs very badly struggling to get out of one mode let alone move across multiple.

4.4 In the real world

Picking out our temperatures like this works nicely because we already know how to vary them to what we want. However, in practice when we want to do MCMC we will not know this so we need some way to pick out temperatures with no prior knowledge.

One approach to this is: [SR15]

$$\pi_i(x) = \pi(x)^{\beta_i} \tag{2}$$
$$\beta_i = \frac{1}{1 + (i-1)\Delta T} \tag{3}$$

I went back to the normal mixture example and applied this method for picking the functions π_i. I took ΔT to be 1 as it seemed to give sensible results but this is something that I could tune further. In the literature it suggests having between a 20% and 60% acceptance rate for the metropolis step between chains and this was in the region. However, I was still getting very good results this way far outperforming the RWM. The results can be found in figure 4.

Whilst not quite as good as when we could simply pick out what we wanted the distribution to be, we are still getting very good results compared to the standard method. The histograms fit the probability distribution function much nicer, the dependence is converging much faster and from the trace plot it is clear how we are getting better mixing.

5 Future Work

The MCMCMC algorithm can be written in parallel (sometimes called Multi-Core MCMCMC or MCMCMCMC), with each chain being calculated in a different processor of the computer. I had an attempt at this but couldn't get the code to work. [ADHR04] MCMCMC clearly has a large computational cost as compared to a standard algorithm as it has to run multiple chains rather than just one. However, by doing this we could collapse the time back down to just over the time to run one chain.

Table 1: Running times in seconds for 10000 samples

	Figure 1	Figure 2	Figure 3	Figure 4
RWM	0.96	0.94	0.92	1.57
MCMCMC	6.69	6.55	6.54	10.17

Looking at my running times shows how this would be desirable.

References

[ADHR04] Gautam Altekar, Sandhya Dwarkadas, John Huelsenbeck, and Fredrik Ronquist. Parallel metropolis coupled markov chain monte carlo for bayesian phylogenetic inference. *Bioinformatics (Oxford, England)*, 20:407–15, 03 2004.

[CR14] Radu V. Craiu and Jeffrey S. Rosenthal. Bayesian computation via markov chain monte carlo. *Annual Review of Statistics and Its Application*, 1(1):179–201, 2014.

[SR15] Jeff J. Shi and Daniel L. Rabosky. Speciation dynamics during the global radiation of extant bats. *Evolution*, 69(6):1528–1545, 2015.

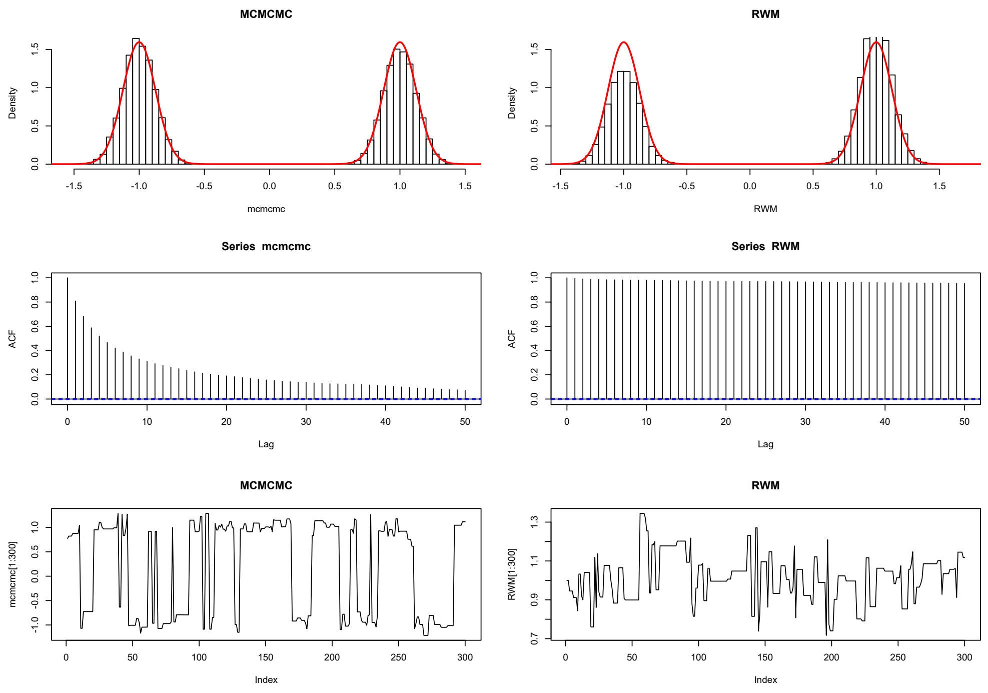

Figure 1: Graphs in the normal case

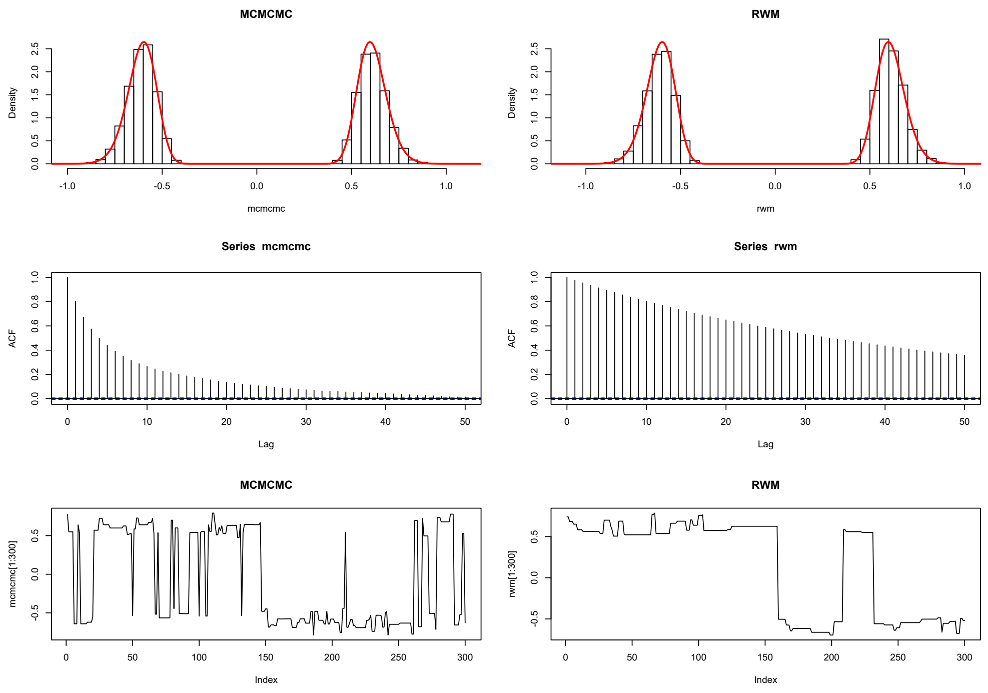

Figure 2: Graphs in the lognormal case

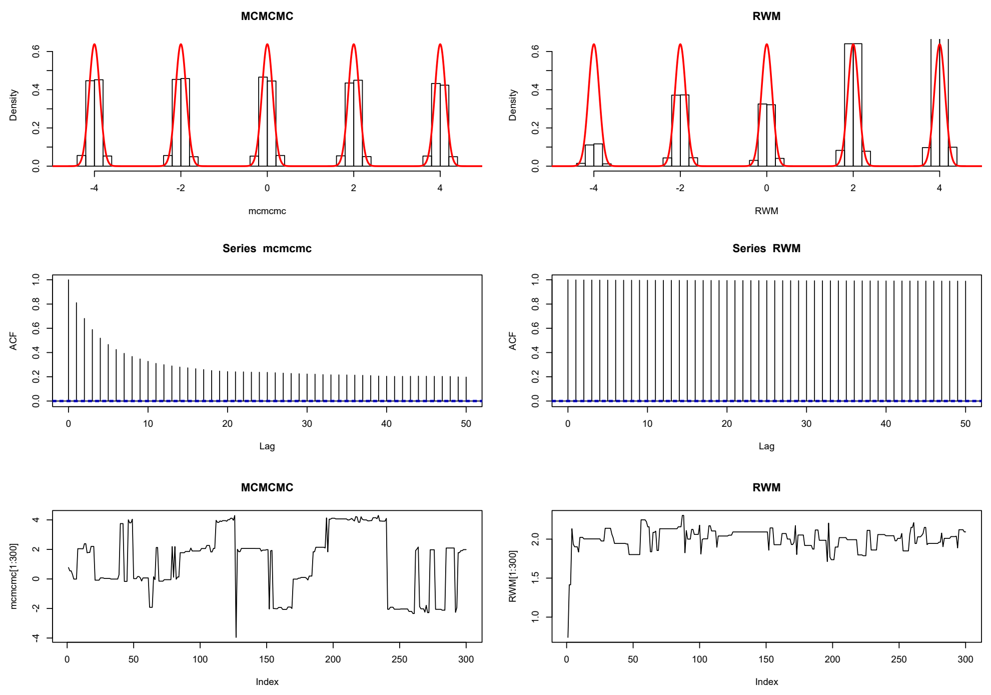

Figure 3: Plots for a five mixture

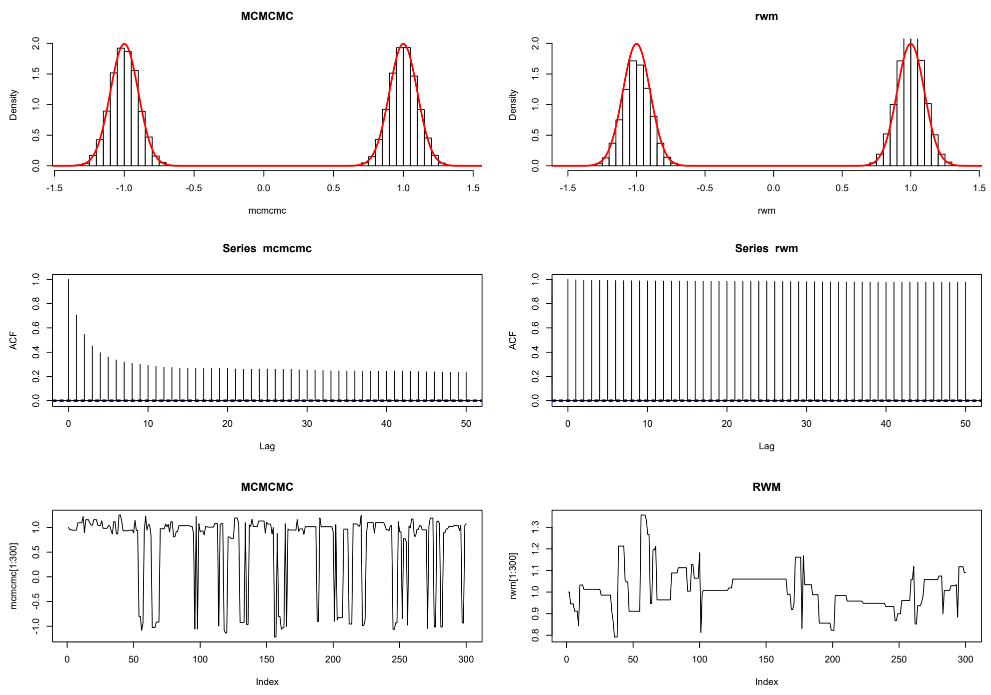

Figure 4: Using the power method of taking temperatures in the normal mixture case