STOR608 Sprint Report: MCMC Algorithms

L. M. Howell

1 Introduction

Markov Chain Monte Carlo (MCMC) algorithms are a helpful technique used when one wants to sample from a probability distribution that is unknown. Mainly this applies in the Bayesian setting, when the posterior distribution is only known up to a constant of proportionality or is intractable.

The idea is that, if a Markov chain is irreducible and aperiodic, then it has a unique stationary distribution. This stationary distribution will be denoted π and must satisfy the detailed balance equation

$$\pi(x)p(x|y) = \pi(y)p(y|x)$$

for all x, y in the state space of the Markov chain. This $p(\cdot)$ is the transition probability associated with the Markov chain. By running this Markov chain, after a burn-in period, the stationary distribution is reached. Then Monte Carlo estimation can be used to calculate estimates of the moments of the stationary distribution. This gives an effective way to sample from distributions with no known closed form.

It is important to stress the correlated nature of MCMC samples. Most of the time the goal is to draw i.i.d. samples from a target posterior distribution. Of course, given the definition of a Markov chain, samples drawn from MCMC algorithms are all dependant on the sample drawn prior to that one, but are independant from every other point in the chain. Therefore, one of the measures of an MCMC algorithm is autocorrelation. This is a measure of the correlation of a time series with a lagged version of itself. One could measure the accuracy of an MCMC algorithm by looking at the autocorrelation (after eliminating burn in) and seeing how close the autocorrelation is to the theoretical optimum. This idea won't be covered in detail here.

This report aims to cover some basic algorithms covered in the sprints, particularly the Metropolis-adjusted Langevin algorithm (MALA) and Gibbs Sampling. There will then be a brief mention of the various pros and cons associated with these methods. This will then be followed by an algorithm from the literature, Stochastic Gradient MCMC.

2 MALA

Langevin diffusion is defined by the Stochastic Differential Equation

$$dX_t = \frac{1}{2}\nabla \log \pi(X_t)\mathrm{d}t + \mathrm{d}B_t$$

where B_t is Brownian motion and π is the stationary distribution of the Markov chain. This equation can be discretised and becomes

$$X_{t+h} = x_t + \frac{h}{2}\nabla \log \pi(x_t) + \sqrt{h}\epsilon_t.$$

Here, ϵ_t is a vector of i.i.d. Normal random variables. However, just sampling from this doesn't produce samples from π exactly, as for a fixed h this only approximates Langevin diffusion. Only as h tends to zero will it be accurate. This can be corrected by introducing a Metropolis-Hastings accept/reject step.

An accept/reject step says that, when we generate a new sample x' from X', we will only accept that as a new sample with probability

$$\alpha(x' \mid x) = \min\left\{1, \frac{\pi(x')q(x \mid x')}{\pi(x)q(x' \mid x)}\right\}.$$

In full, we construct a proposal distribution

$$X' = x + \frac{h}{2}\nabla \log \pi(x) + \sqrt{h}\epsilon_t.$$

This gives

$$q(x'|x) = N(x'; x + \frac{h}{2}\nabla \log \pi(x), h)$$

which follows from ϵ_t being Gaussian. Due to the $\nabla \log \pi(x)$ term, the new sample moves in the direction of highest gradient. This means it converges faster than more basic algorithms such as a Random Walk, which just proposes new states at random.

The Random Walk-Metropolis algorithm (shortened here to just Random Walk, or RW) simply says that a new proposed state x' is obtained as a realisation of the random variable $X' = x+\epsilon$, where $\mathbb{E}(\epsilon) = 0$. This is then accepted or rejected using the Metropolis-Hastings step. Most commonly $\epsilon \sim N(\mathbf{0}, h\mathbf{I})$. The symmetric nature of the proposal density means that $q(x'|x) = q(x|x')$ and the acceptance probability cancels to just $\pi(x')/\pi(x)$ which speeds up the algorithm.

The other thing to consider when it comes to both the Random Walk and MALA is tuning parameters. The h used in both algorithms must be tuned appropriately; if it is to small then most proposed moves are accepted and the algorithm is faster. However the samples collected will take a lot longer to converge to the posterior distribution as it moves around less, slowing it down again. Comparatively, if h is too large, then too many proposed states are rejected and the algorithm is again slow [Brooks et al., 2011, p. 95]. This can often be seen in trace plots in the form of the Manhattan effect, where there are long stretches of no new state being accepted and the trace plot looks like city skyline.

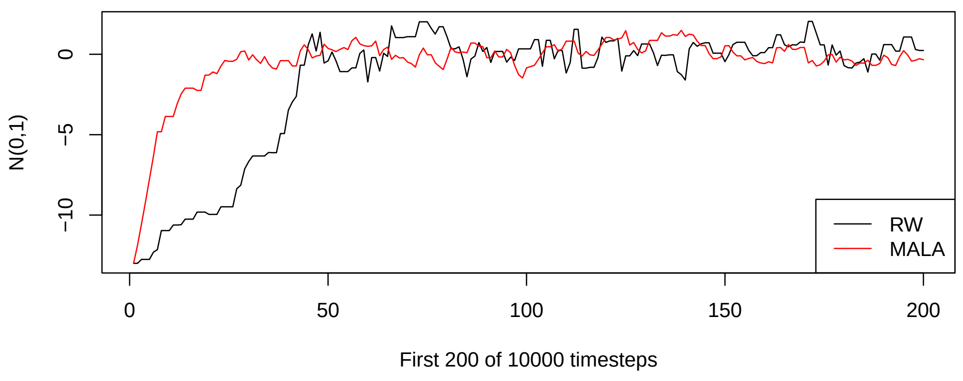

Figure 1: First 200 of 10000 iterations of the Random Walk algorithm against the MALA with target distribution $N(0, 1)$.

To clearly see the difference between the MALA and the more basic Random Walk, we show some examples. The first example is the most basic; a one dimensional target distribution of the standard normal. Figure 1 clearly shows that the MALA converges to the target distribution more quickly than the RW. This means that, with less burn in, there are more samples which can be used in Monte-Carlo estimation.

The next example is with a target distribution of the bivariate Normal with $\mu = \binom{2}{5}$ and $\Sigma = \left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)$.

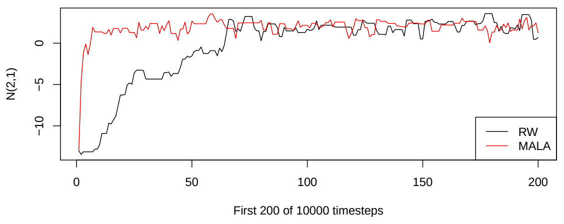

Figure 2: First 200 of 10000 iterations of the Random Walk algorithm against the MALA with target distribution $N(2, 1)$.

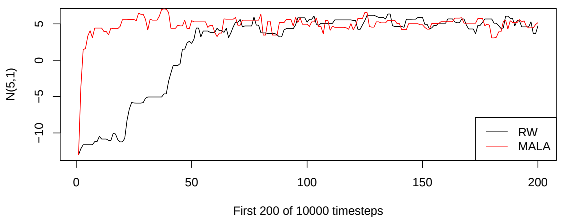

Figure 3: First 200 of 10000 iterations of the Random Walk algorithm against the MALA with target distribution $N(5, 1)$.

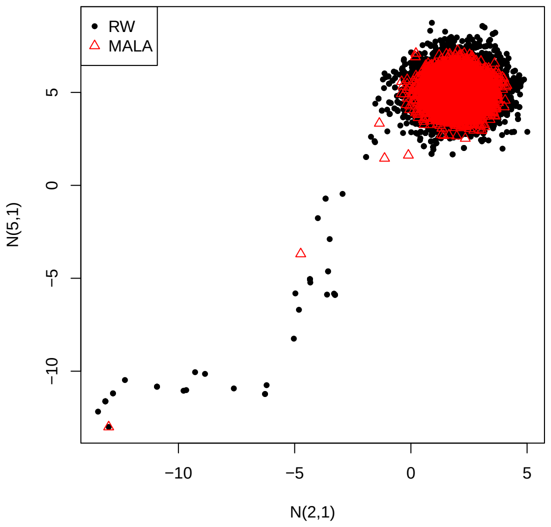

Figure 4: 10000 iterations of RW against the MALA with target distribution of bivariate Normal with means 2 and 5.

It's clear to see from these that as the dimension increases, the RW suffers in a way that MALA does not. Partially, this is to do with the fact that as the dimension of the problem increases, these examples used the same value for their respective tuning parameters: $h = 0.2$ was used for MALA in both cases, and $h = 1$ for the Random Walk. Roberts and Rosenthal, (2001) suggest that, under specific circumstances, as the dimensions increase, the optimal h for the Random Walk scales in proportion to d^{-1}. For MALA, h scales in proportion to $d^{-1/3}$. This means that MALA has a smaller convergence time [Roberts and Rosenthal, 2001, p. 359]

3 Gibbs Sampling

In some situations, the closed form of $\pi(x)$ may be unknown, but some information about the conditional distributions may be known. We can take advantage of this with something known as Gibbs Sampling. For $x = (x_1, x_2, \ldots, x_d)$ we denote $x_{-i} = (x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)$ i.e. all parameters except x_i. So we make a proposal density of $q(x'|x) = \pi(x_i|x_{-i})$.

Below is an example to illustrate, heavily taken from [Casella and George, 1992], though the algorithm written, values used and plot are original. Say we know that the joint distribution of two random variables X and Y is

$$\pi(x, y) \propto \binom{n}{x} y^{x+\alpha-1}(1-y)^{n-x+\beta-1}$$

and that we are interested in the marginal distribution of $\pi(x)$. This is analytically solvable, giving a marginal distribution of the beta-binomial with parameters n, α, β. We can also calculate conditional distributions

$$\pi(x \mid y) \sim \text{Bi}(n, y)$$
$$\pi(y \mid x) \sim \text{Beta}(x + \alpha, n - x + \beta).$$

We can then use Gibbs to alternate sampling from each conditional distribution and updating the parameters. Figure 5 is a histogram of 5000 samples from a beta-binomial distribution compared wiht 5000 draws of the x variable from a Gibbs Sampler. The parameters were $n = 10, \alpha = 2, \beta = 2$ and starting values $x = 1, y = 1$. The seed used was 1313.

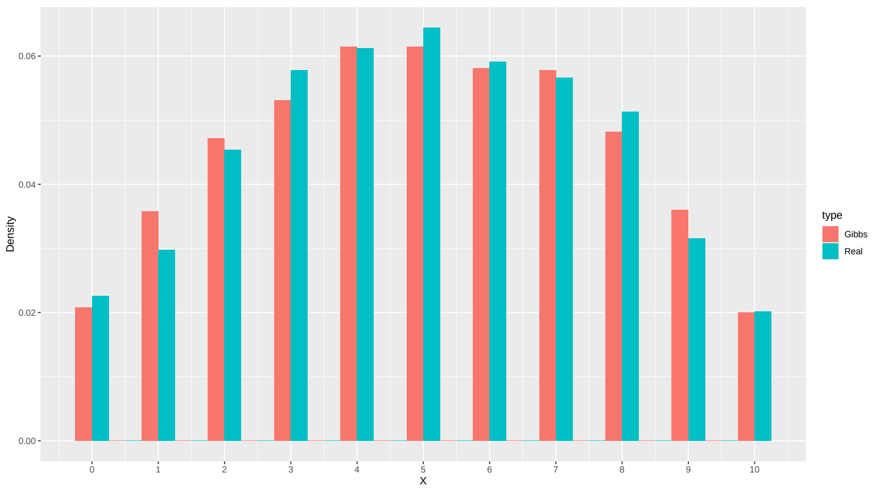

Figure 5: 5000 samples from a Gibbs Sampler versus 5000 samples from the analytical marginal distribution.

Gibbs Sampling is, in fact, a type of Metropolis-Hastings algorithm, however the algebra below demonstrates how the acceptance probability collapses to always accepting the proposed next state [Gelman et al., 2013, p. 281].

$$\begin{aligned}
\alpha(x' \mid x) &= \min\left\{1, \frac{\pi(x')q(x \mid x')}{\pi(x)q(x' \mid x)}\right\} \\
\alpha(x_i \mid x_{-i}) &= \min\left\{1, \frac{\pi(x_i)\pi(x_{-i} \mid x_i)}{\pi(x_{-i})\pi(x_i \mid x_{-i})}\right\} \\
&= \min\left\{1, \frac{\pi(x_i) \times \frac{\pi(x_{-i}, x_i)}{\pi(x_i)}}{\pi(x_{-i}) \times \frac{\pi(x_i, x_{-i})}{\pi(x_{-i})}}\right\} \\
&= \min\left\{1, \frac{\pi(x)}{\pi(x)}\right\} = 1
\end{aligned}$$

4 Stochastic Gradient MCMC

One of the main considerations when selecting an MCMC algorithm for a certain task is time, with faster algorithms being more desirable. A key example of this is ULA - the unadjusted Langevin algorithm. This is a more basic version of MALA, where there is no accept/reject step. As mentioned earlier, ULA does not produce samples that converge in the limit to π [Roberts and Tweedie, 1996]. However, due to the added efficiency of the algorithm gained by forgoing the accept/reject step, in large data settings it is sometimes appropriate to use ULA as the posterior distribution may be "close enough". However this is not always appropriate, and other ways of speeding up algorithms must be found.

Most commonly, there are two ways of speeding up an MCMC algorithm. The first is parallelising the algorithm across multiple computer cores. This involves outputting partial posteriors that rely on a subset of the data and then merging them again at the end to create the full posterior. The second is subsampling; for example some results show an estimate of the accept/reject step can be gained from a subset of the data [Bardenet et al., 2014, Quiroz et al., 2018]. A drawback to these methods is having to sacrifice accuracy for computational efficiency.

A popular type of method within the subsampling variety is Stochastic Gradient MCMC (SGMCMC). This involves using a subsample of the data to calculate an estimate of the gradient. This will be discussed in the context of the MALA.

With $\pi(x)$ being our posterior, we know that

$$\pi(x) = \frac{g(x)}{\int g(x)\mathrm{d}x}$$

where $g(x)$ is the prior $\times$ likelihood. Therefore, when we want to calculate $\nabla \log \pi(x)$ this shakes out to

$$\nabla \log \pi(x) = \nabla \log \frac{g(x)}{\int g(x)\mathrm{d}x} = \nabla \log \frac{g(x)}{C} = \nabla \log g(x) - \nabla \log C = \nabla \log g(x)$$

for C constant.

In very simple cases, this gradient term has a closed form. However, in high dimensions these gradients are calculated numerically and modern methods can do this very accurately. This does slow algorithms like ULA and MALA down, as calculating this estimate involves summing over the full data gathered so far. Therefore, using a reliable estimate of the gradient is used to streamline the algorithm.

If we assume that we have some data $x_1, x_2, \ldots, x_N$ which is independant, then the posterior distribution is

$$\pi(x;\theta) \propto p(\theta) \prod_{i=1}^{N} f(x_i \mid \theta),$$

where $p(\theta)$ is the prior and $f(x_i \mid \theta)$ is the likelihood of the ith observation. Of course, we are interested in the gradient of the log of $\pi(x)$. tarting with $\log \pi(x)$;

$$\begin{aligned}\log \pi(x) &\propto \log \left(p(\theta) \prod_{i=1}^{N} f(x_i \mid \theta) \right) \\ &= \log p(\theta) + \sum_{i=1}^{N} \log f(x_i \mid \theta) \\ &= \sum_{i=1}^{N} \left(\frac{1}{N} \log p(\theta) + \log f(x_i \mid \theta) \right).\end{aligned}$$

Since taking the gradient turns any constants to 0 as demonstrated above (which deals with the proportionality sign), the final result becomes

$$\begin{aligned}\nabla \log \pi(x) &= \nabla \sum_{i=1}^{N} \left(\frac{1}{N} \log p(\theta) + \log f(x_i \mid \theta) \right) \\ &= \sum_{i=1}^{N} \nabla \left(\frac{1}{N} \log p(\theta) + \log f(x_i \mid \theta) \right).\end{aligned}$$

This is where we can clearly see why calculating this gradient involves having to sum over the whole of the data, slowing down the algorithm. Nemeth and Fearnhead (2021) well explain in *Stochastic Gradient Markov Chain Monte Carlo* a method of approximating this gradient. They take a subset of the data $X_n \subset X_N$ where $n \ll N$. They then calculate the gradient as

$$\hat{\nabla} \log \pi(x) = \frac{N}{n} \sum_{x_i \in X_n} \nabla \left(\frac{1}{N} \log p(\theta) + \log f(x_i \mid \theta) \right)$$

which makes calculating the gradient an $\mathcal{O}(n)$ calculation rather than $\mathcal{O}(N)$. This estimator is described as valid if it unbiased and has finite variance [Ahn et al., 2014], which this does.

Naturally as the accuracy of the gradient estimator increases, we expect the accuracy and convergence time of the algorithm to increase and decrease respectively. Other valid estimators are available.

References

[Ahn et al., 2014] Ahn, S., Shahbaba, B., and Welling, M. (2014). Distributed stochastic gradient mcmc. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32(2) of *Proceedings of Machine Learning Research*, pages 1044–1052, Bejing, China. PMLR.

[Bardenet et al., 2014] Bardenet, R., Doucet, A., and Holmes, C. (2014). Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In *International conference on machine learning*, pages 405–413. PMLR.

[Brooks et al., 2011] Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, Boca Raton, Fla.

[Casella and George, 1992] Casella, G. and George, E. I. (1992). Explaining the Gibbs sampler. *The American Statistician*, 46(3):167–174.

[Gelman et al., 2013] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis*. Texts in statistical science. CRC Press, Boca Raton, Florida, third edition.

[Nemeth and Fearnhead, 2021] Nemeth, C. and Fearnhead, P. (2021). Stochastic gradient Markov Chain Monte Carlo. *Journal of the American Statistical Association*, 116(533):433–450.

[Quiroz et al., 2018] Quiroz, M., Kohn, R., Villani, M., and Tran, M.-N. (2018). Speeding up MCMC by efficient data subsampling. *Journal of the American Statistical Association*.

[Roberts and Rosenthal, 2001] Roberts, G. O. and Rosenthal, J. S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical science*, 16(4):351–367.

[Roberts and Tweedie, 1996] Roberts, G. O. and Tweedie, R. L. (1996). Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363.

[Welling and Teh, 2011] Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688.