STOR608: Adaptive Metropolis Algorithms

Peter Greenstreet

January 8, 2020

1 Introduction

The aim of MCMC (Markov chain Monte Carlo) algorithms is to allow us to get a sample from a probability distribution. In many cases $\pi(x)$ is a Bayesian posterior distribution which we only know up to proportionality. We use our MCMC algorithm to produce a random sample $x_1, x_2, ..., x_n$ from our probability distribution. Where x_j is a vector such that $x_j = (x^{(1)}, ..., x^{(d)})$, with d being the dimension of our probability distribution.

Once we have drawn our sample we need to remove the first b points $(x_1, ..., x_b)$. This is because we want a stationary distribution $\pi(x)$, so need to remove the data points before our algorithm has converged on the stationary distribution. For example, if we started in an area of low probability of our posterior distribution, the first b draws may be from an area which we are unlikely to ever visit again, resulting in bias in our sample if we included these draws.

During this report, we are going to use the Metropolis-Hastings Algorithm. This algorithm works by us choosing a proposal density $q(y|x)$. This is where y is our proposed next sample point. Next, we calculate:

$$\alpha(y|x) = \min\left(1, \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}\right),$$

where α is our acceptance probability of keeping the proposed next sample point.

We are going to focus on two methods of adaptive Metropolis algorithms. The first method is Adaptive Proposal (AP) algorithm from Haario et al. (1999), the second is Adaptive Metropolis (AM) algorithm from Haario et al. (2001). Both these methods build from the Random Walk Metropolis Algorithm (RWMA) which was first introduced by Metropolis et al. (1953). We will introduce a basic outline to this in section 2.

The main reason we are focusing on algorithms based on random walk Metropolis is that many other methods involve knowing further information about the underlying posterior distribution $\pi(x)$. For example, in the Gibbs sampler (Geman and Geman (1984)) we need to know $\pi(x^{(i)}|x^{(-i)})$ which is not always possible. In Metropolis Adjusted Langevin Algorithms (MALA) (Grenander and Miller (1994)) and Hybrid Monte Carlo algorithm (HMC) (Duane et al. (1987)) we need to be able to calculate the gradient at different points in our posterior distribution, which can be computationally very expensive if this is not possible to work out analytically.

2 Random Walk Metropolis Algorithm (RWMA)

RWMA is one of the most commonly used methods, due to its simplicity to implement and as all you need to know is the posterior distribution up to proportionality. The method works by beginning with steps 1,2 then doing n iterations of steps 3-5:

1. We have our initial value x_0. When choosing x_0 it is important to check that $\pi(x_0) \neq 0$. so that at step 4 we do not divide by 0.

2. Then choose an ϵ which is drawn from a symmetric distribution with mean 0. This is so $q(y|x) = q(x|y)$.

3. Next find a proposal y which equals $y = x_{i-1} + \epsilon$.

4. We decide whether we accept our proposal y by finding our acceptance probability which has been simplified by our choice of ϵ:

$$\alpha = \min\left(1, \frac{\pi(y)}{\pi(x_{i-1})}\right).$$

5. Then we test whether $\alpha > u$, where u is a random draw from a uniform$(0, 1)$. If $\alpha > u$ then $x_i = y$ if not $x_i = x_{i-1}$.

One of the biggest challenges with random walk Metropolis algorithm is choosing an ϵ so we have a 'good' acceptance rate. A 'good' acceptance rate is one where we have low correlation between each sample point. If we choose ϵ so that we barely move around our sample space, then we will have high correlation between each point. However if we choose ϵ so that we move a lot around our sample space, then we will rarely accept a proposed point which will also result in high correlation.

A commonly used and well studied distribution for ϵ is $\epsilon \sim N(0, \Sigma)$ which is the distribution we will use in this paper. Roberts et al. (1997) shows that ϵ should be chosen to give an acceptance rate of around 23% in high dimension. They then went on to show in further work (Roberts and Rosenthal (2001)) that an acceptance rate between 15% and 50% is still fairly efficient. Further in this paper they have shown that Σ is best when proportional to the true covariance matrix Σ_T. However the true covariance matrix is rarely known, therefore it can be impossible to choose Σ so that it is proportional to the true covariance matrix. This is why we need adaptive Metropolis algorithms.

3 Adaptive Proposal (AP) algorithm

In Haario et al. (1999) they have given the AP algorithm as a way to resolve the issue with the choice of Σ. This method does produce a slightly biased stationary distribution, as shown in an example in Haario et al. (2001). In most instances this bias is so small it can be ignored. The method works in a very similar way to that of the RWMA, however we will update our covariance matrix.

We choose an initial covariance matrix R_0 which can be chosen as the identity if we have no prior knowledge about what it should be. After t_0 time of RWMA where we have sampled at least H points so $\{x_1, ..., x_{t_0-H+1}, ..., x_{t_0}\}$ we then calculate the covariance matrix R_t which is $d \times d$ matrix by collecting the points $x_{t_0-H+1}, ..., x_{t_0}$ into a $H \times d$ matrix call K then we find:

$$R_t = \frac{1}{1-H}\widetilde{K}^T\widetilde{K},$$

where $\widetilde{K} = K - E(K)$. For the next U (update frequency) steps, we find proposal points $y = x_{t+i-1} + N(0, c_d^2 R_t)$, where $c_d = 2.4/\sqrt{d}$ from which Gelman et al. (1996) was shown to correspond to theoretically optimal mixing. We then update R_t again by looking at the last H points in our sample. We then continuously repeat this step, updating our R_t after every U points until we have our desired sample length.

4 Adaptive Metropolis (AM) algorithm

The AM algorithm uses a similar principle as AP algorithm. It works by us continuously updating our covariance matrix after some time t_0. t_0 is free for us to choose and it represents our trust in R_0. We find our covariance matrix in the following way:

$$R_t = \begin{cases} R_0 & t \leq t_0 \\ c_d \text{cov}(x_0, ..., x_{t-1}) + c_d \delta I_d & t > t_0 \end{cases}$$

This is where $\delta > 0$ is chosen to be so small it is negligible. The reason Haario et al. (2001) has set $\delta > 0$ rather than $\delta = 0$ is they were only able to prove the ergodicity properties for $\delta > 0$. The ergodicity property is that the AM provides the correct simulation of the target distribution, unlike the AP algorithm which is biased.

When calculating R_t for $t \geq t_0 + 1$ we use a recursion formula, which results in a decrease in computational cost of the algorithm compared to recalculating the covariance from scratch each time.

The issue with the AM algorithm is it is not Markovian. Markovian is when:

$$X_t | X_{t-1}, ..., X_1 = X_t | X_{t-1}$$

This does not hold for the AM algorithm as for $t > t_0$ the covariance is dependent on the values of $X_0, ..., X_{t-1}$. However, in Haario et al. (2001) they show that the asymptotic dependence between the elements of the chain is weak enough that it can be ignored.

5 Comparing methods

We are going to compare our different algorithms on 2 different target distributions: uncorrelated Gaussian distribution and correlated Gaussian distribution. For each target distribution we will look at a 2 dimensional example and a 16 dimensional example. We have chosen the 2 dimensional examples as these can be easily visually represented. We are looking at the 16 dimensional examples to get an understanding of how well the algorithms work in high dimension.

For the uncorrelated Gaussian distributions we will set $\mu = \mathbf{0}$ for both and set Σ_T to be:

2 Dimensional

$$\Sigma_T = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$$

16 Dimensional

$$\Sigma_T = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 16 \end{pmatrix}.$$

We have chosen these matrices to test how the algorithms perform when each parameter has a different variance.

For the correlated Gaussian distributions we will set $\mu = \mathbf{0}$ for both and set Σ_T to be:

2 Dimensional

$$\Sigma_T = \begin{pmatrix} 1 & 0.1 \\ 0.1 & 1 \end{pmatrix}$$

16 Dimensional

$$\Sigma_T = \begin{pmatrix} 1 & 0.1 & \dots & 0.1 \\ 0.1 & 1 & \dots & 0.1 \\ \vdots & \vdots & \ddots & \vdots \\ 0.1 & 0.1 & \dots & 1 \end{pmatrix}.$$

We have chosen these matrices to test how the algorithms perform when all the parameters are correlated with each other.

We will compare AM algorithm, AP algorithm and RWMA. We will set the starting Σ to be the identity matrix for the AM algorithm, AP algorithm and RWMA. Also an example of RWMA with Σ tuned to $c_d\Sigma_T$ (TRWMA).

In table 1 we have calculated the Mean($||E||$) and SD($||E||$) as suggested in Roberts et al. (1997), as a way to see how well our algorithm is getting its sample from the probability distribution. Mean($||E||$) is the mean distance the expected values are from the true values calculated over 100 repetitions.

$$\text{Mean}(||E||) = \frac{1}{100}\sum_{j=1}^{100}\left(\sum_{i=1}^{d}(E_j^i)^2\right)^{1/2},$$

where E_j is the mean vector of chain j. SD($||E||$) is the standard deviation of this distance. For the 16 dimensional cases we have made a sample of length 50000 and for the 2 dimensional cases we have made a sample of length 10000. The burn in period for all the algorithms is half the sample length. For the AP and AM we have set t_0 to equal 1000 and 5000 for 2 dimensional and 16 dimensional cases respectively. Also for the AP we set H= 200, U= 200 and H= 1200, U= 1000 for 2 dimensional and 16 dimensional cases respectively as suggested in Roberts et al. (1997).

2 Dimensional Case				
	Uncorrelated		Correlated	
Algorithm	Mean($||E||$)	SD($||E||$)	Mean($||E||$)	SD($||E||$)
AP	0.06874	0.03809	0.05447	0.02821
AM	0.05992	0.03015	0.04522	0.02432
TRWMA	0.05927	0.02556	0.04902	0.02193
RWMA	0.08114	0.04116	0.05482	0.02589
16 Dimensional Case				
	Uncorrelated		Correlated	
Algorithm	Mean($||E||$)	SD($||E||$)	Mean($||E||$)	SD($||E||$)
AP	0.61355	0.13876	0.21244	0.03993
AM	0.51490	0.10319	0.17917	0.03326
TRWMA	0.52643	0.11203	0.18173	0.03621
RWMA	0.80249	0.18983	0.25338	0.05219

Table 1: Mean($||E||$) and SD($||E||$) from our different probability distributions.

In the 2 dimensional cases we will use the ACF plots to study the effective sample size we have got for each algorithm. The effective sample size is calculated by working out how many points we need to remove, so each of our sample points is independent of one another. We do this by seeing at which lag l we fall within the 95% independence confidence interval. Then we keep only 1 in every l sample points, resulting in us having an independent sample. In figure 1 we have plotted for each algorithm, the ACF plot corresponding to the parameter with the worst lag in the non correlated example. The AP, AM, TRWMA and RWMA lags fall within the 95% CI at $l = 18, 16, 20, 36$ respectively which results in an effective sample size of $277, 312, 250, 138$.

In the correlated 2 dimensional case we found that AP, AM, TRWMA and RWMA lags fall within the 95% CI at $l = 13, 11, 20, 30$ respectively which results in an effective sample size of $384, 454, 250, 166$.

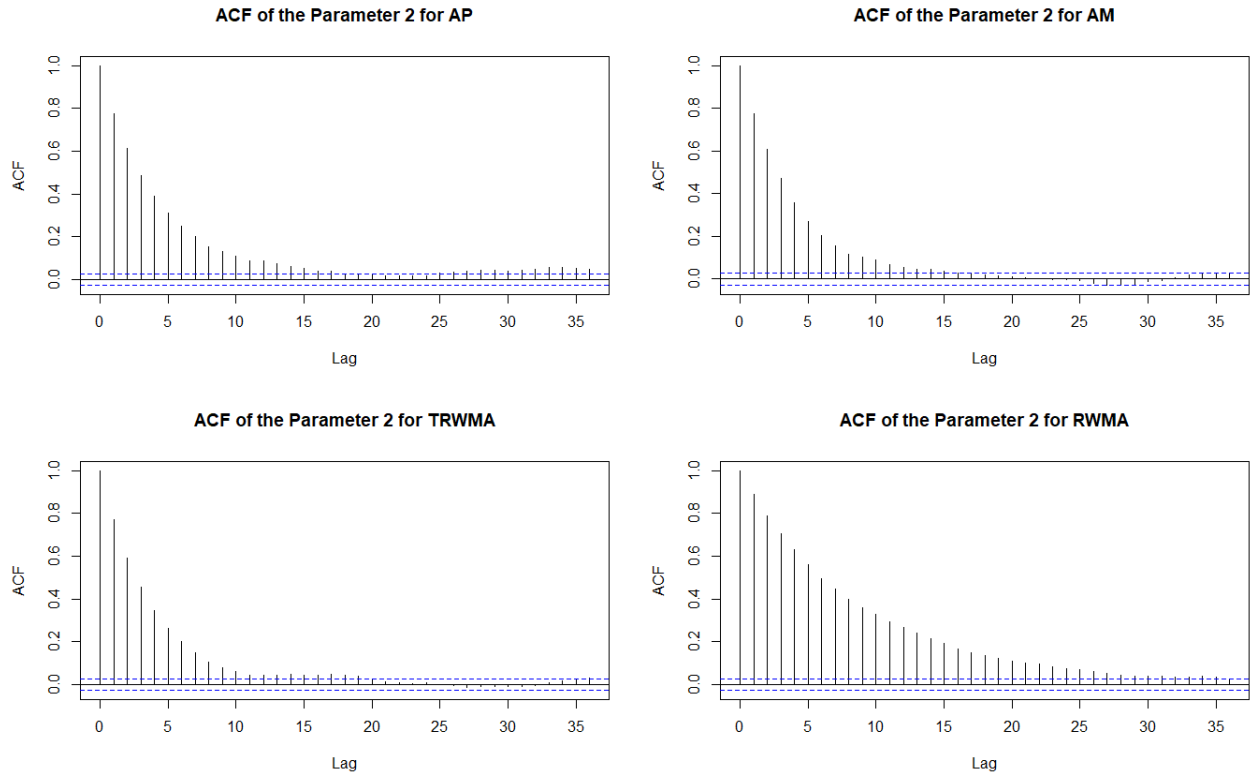

Figure 1: ACF plots for uncorrelated 2 dimensional case.

6 Conclusion

From section 5 we have found using the ACF plots that the AM algorithm has given us the largest effective sample size in the 2 dimensional cases. The AP algorithm also gave us a larger effective sample size than both TRWMA and RWMA. This is a useful feature as it reduces the length of the sample we need to get from our algorithm, resulting in faster computation.

Furthermore, as can be seen in table 1, the AM algorithm gives us a better Mean($||E||$) compared to the RWMA and AP. The true Mean($||E||$) of our distribution is 0 so we want our result to be close to 0. AM gives us results very close to TRWMA, sometimes with AM giving us the better result. This is potentially caused as we have tuned TRWMH by using the true covariance matrix times c_d^2 and as c_d^2 is not always the best tuning parameter, so our AM may be getting closer to the optimal tuning parameter.

When studying the SD($||E||$) in table 1 we have found a very similar pattern as for the Mean($||E||$). Either the AM or TRWMA give us the smallest SD($||E||$) with both being close to each another. We want a small SD($||E||$) as if we have a large SD($||E||$) there is a higher chance of the sample mean being further away from the true probability distributions mean.

To conclude, out of the algorithms we have discussed, we would choose the AM algorithm as our

method for drawing a sample from a probability distribution. This is due to it performing better when compared to the AP algorithm in every test, very similarly to the TRWMA without the need for us to manually tune Σ.

In future reports we would like to study how our algorithms work on different probability distributions, including looking at how they perform on ones which are bimodal, to see whether the algorithms are able to move between modes, or just get stuck in one area of the probability distribution. We would also study the effects of having different distribution for ϵ such as the logistic distribution, student's t distribution or Cauchy distribution. Further work would also include a look at comparing our algorithms to other methods such as the Gibbs, MALA and HMC. Finally we would also like to look into different types of adaptive Metropolis algorithms such as those discussed in Roberts and Rosenthal (2009).

References

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid monte carlo. *Physics letters B*, 195(2):216–222. 1

Gelman, A., Bois, F., and Jiang, J. (1996). Physiological pharmacokinetic analysis using population modeling and informative prior distributions. *Journal of the American Statistical Association*, 91(436):1400–1412. 2

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741. 1

Grenander, U. and Miller, M. I. (1994). Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581. 1

Haario, H., Saksman, E., and Tamminen, J. (1999). Adaptive proposal distribution for random walk metropolis algorithm. *Computational Statistics*, 14(3):375–395. 1, 2

Haario, H., Saksman, E., and Tamminen, J. (2001). An adaptive metropolis algorithm. *Bernoulli*, 7(2):223–242. 1, 2, 3

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092. 1

Roberts, G. O., Gelman, A., and Gilks, W. R. (1997). Weak convergence and optimal scaling of random walk metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120. 2, 4

Roberts, G. O. and Rosenthal, J. S. (2001). Optimal scaling for various metropolis-hastings algorithms. *Statistical Science*, 16(4):351–367. 2

Roberts, G. O. and Rosenthal, J. S. (2009). Examples of adaptive mcmc. *Journal of Computational and Graphical Statistics*, 18(2):349–367. 6