Optimising and Adapting the Metropolis Algorithm

by Jeffrey S. Rosenthal[1]
(February 2013; revised March 2013 and May 2013 and June 2013)

1 Introduction

Many modern scientific questions involve high dimensional data and complicated statistical models. For example, data on weather consist of huge numbers of measurements across spatial grids, over a period of time. Even in simpler settings, data can be complex: for example, Bartolucci et al. (2007) consider recurrence rates for melanoma (skin cancer) patients after surgery. The probability of recurrence for an individual may depend on physical or biological characteristics of their cancerous lesion, as well as other factors. A statistical model in this context may involve a large number of variables and a correspondingly large number of parameters, which are often represented by a vector θ of some dimension d. To assess the relevance of specific variables for disease recurrence, and to build models that give a risk of recurrence for any given individual, researchers often use *Bayesian analysis* (see e.g. Box and Tiao, 1973; Gelman et al., 2003; Carlin and Louis, 2008). In this framework, the parameter vector is assumed to follow some probability distribution (of dimension d), and the challenge is to combine a "prior" distribution for θ (typically based on background information about the scientific area) with data that are collected, so as to produce a "posterior" distribution for θ. This probability distribution (call it $\pi(\theta)$) can then be used to answer important scientific questions (e.g., is the size of a cancerous lesion related to the risk of recurrence after surgery?) and to calculate specific probabilities (e.g., this person has a 20% probability of a recurrence within the next five years).

One challenge for Bayesian analysis in situations where the data and parameter vectors are high dimensional is that it is difficult or impossible to compute probabilities based on the posterior distribution. If there is some outcome A of interest (e.g., the outcome that a specific individual's cancer

[1]Department of Statistics, University of Toronto, Toronto, Ontario, Canada M5S 3G3. Email: jeff@math.toronto.edu. Web: http://probability.ca/jeff/ Supported in part by NSERC of Canada.

will recur), then its probability is given by an integral (over A):

$$\Pi(A) \;:=\; \int_A \pi(x)\, dx\,. \tag{1}$$

Also, the expected (average) value of any particular quantity h (e.g., the size of the lesion if the cancer does recur) is also given by an integral (now over the state space $\mathcal{X}$ of all possible vectors):

$$\mathbf{E}_\pi(h) \;:=\; \int_{\mathcal{X}} h(x)\, \pi(x)\, dx\,. \tag{2}$$

So, to draw conclusions from a Bayesian statistical model, "all" we have to do is compute integrals like (1) and (2).

Unfortunately, integrals like (1) and (2) are sometimes very difficult to compute. For example, one commonly-used posterior density (corresponding to a "variance components model" in which individuals are divided into groups) is given by the formula:

$$\pi(V, W, \mu, \theta_1, \ldots, \theta_K) \;=\; C\, e^{-b_1/V} V^{-a_1-1} e^{-b_2/W} W^{-a_2-1} e^{-(\mu-a_3)^2/2b_3} V^{-K/2} W^{-\frac{1}{2}\sum_{i=1}^K J_i}$$

$$\times\; \exp\left[-\sum_{i=1}^{K} (\theta_i - \mu)^2/2V - \sum_{i=1}^{K}\sum_{j=1}^{J_i} (Y_{ij} - \theta_i)^2/2W \right].$$

Here K is the number of different groups (i.e., the number of different values θ_i), V and W and μ are additional unknown parameters of the model, and the a_i and b_i and Y_{ij} are known constants. A typical application might have, say, $K = 19$, so that π is a 22-dimensional function. In such cases, direct computation of integrals like (1) and (2), using calculus tricks or numerical integration or anything else, appears to be impossible, and alternative methods must be sought (Evans and Swartz, 2000). What can be done to facilitate estimates of quantities like $\mathbf{E}_\pi(h)$ in such complicated, high-dimensional cases?

2 Monte Carlo Algorithms

One answer to this question is provided by *Monte Carlo algorithms*. These algorithms, named after the famous casino in Monacco, use *randomness* to estimate quantities like (1) and (2). Surprisingly, this turns out to be very helpful!

The most basic ("classical") form of Monte Carlo requires us to *sample* from π, i.e. generate a sequence of independent d-dimensional random vectors (variables) $X_1, X_2, \ldots, X_M$ which each follow the density π, i.e. each have probabilities given by $\mathbf{P}(X_i \in A) = \int_A \pi(x)\, dx$ for all (measurable) $A \subseteq \mathcal{X}$. We can then use this random sample to estimate quantities like (2), by:

$$\mathbf{E}_\pi(h) \;\approx\; \frac{1}{M} \sum_{i=1}^{M} h(X_i)\,. \tag{3}$$

If the Monte Carlo sample size M is sufficiently large, then by the *Law of Large Numbers*, the estimate (3) will be close to the true expected value (2).

Classical Monte Carlo can be very effective, and is widely used for lots of different applications. However, in many cases (such as the above variance components example), it is infeasible to directly sample from π in this sense, i.e. there is no known way to run a computer program which will produce the required sequence of vectors $\{X_i\}$. This state of affairs presented a serious limitation to the use of Monte Carlo algorithms in Bayesian statistical inference problems, until it was largely solved with the introduction of *Markov chain Monte Carlo* algorithms, as we now discuss.

3 Markov chain Monte Carlo (MCMC)

Markov chain Monte Carlo (MCMC) algorithms were first developed in the statistical physics community by Metropolis et al. (1953), and later expanded by statisticians (e.g. Hastings, 1970) before being introduced into the wider statistical community by Gelfand and Smith (1990). They do not use or require a sequence of independent vectors $\{X_i\}$ as above. Instead, they define a *dependent* sequence, more precisely a *Markov chain*, with each new vector X_{i+1} constructed using the previously-constructed vector X_i.

Suppose we can define a sequence of vectors $X_0, X_1, X_2, \ldots$, where each X_{i+1} is constructed using the previous X_i, such that for large n, X_n is *approximately* a sample (i.e., observation) from π. That is, for large n, $\mathbf{P}(X_n \in A) \approx \int_A \pi(x)\, dx$ for all $A \subseteq \mathcal{X}$. This situation is worse than for classical Monte Carlo as in (3), since it is just approximate, and furthermore the random vectors $\{X_n\}$ are no longer independent but rather are each constructed sequentially using the previous vector. Nevertheless, it is sometimes

true that if M is sufficiently large, then we can still approximate $\mathbf{E}_\pi(h)$ by

$$\mathbf{E}_\pi(h) \;\approx\; \frac{1}{M-B} \sum_{i=B+1}^{M} h(X_i)\,. \tag{4}$$

That is, we still average many values together, similar to (3), even though they are no longer independent. Also, by convention we drop the first B "burn-in" observations since they might be too heavily influenced by our initial values X_0 and hence bias our estimate.

So when is an approximation like (4) valid? It turns out that this approximation does hold as $M \to \infty$ provided that the Markov chain is *ergodic* for π, i.e. that the chain's probabilities *converge* to those of π in the sense that $\lim_{n\to\infty} \mathbf{P}(X_n \in A) = \int_A \pi(x)\,dx$ for all $A \subseteq \mathcal{X}$.

This raises the question of how to define a computationally simple Markov chain $\{X_n\}$ which guarantees that $\mathbf{P}(X_n \in A) \to \int_A \pi(x)\,dx$. By standard Markov chain theory, this will hold if the updating rules for $\{X_n\}$ are *irreducible* (i.e., the chain can eventually get to anywhere in $\mathcal{X}$), and *aperiodic* (i.e., the chain doesn't have any forced cyclical behaviour), and leave the density π *stationary* (i.e. if it starts distributed according to π, then it will remain distributed according to π at all future times too). How can we ensure that those conditions hold?

4 The Metropolis Algorithm

An answer to this question was developed in the physics community over fifty years ago (Metropolis et al., 1953). Specifically, given a (possibly important and complicated and high-dimensional) target density π on some state space $\mathcal{X} \subseteq \mathbf{R}^d$ (with $\pi(x) = 0$ for $x \in \mathbf{R}^d \setminus \mathcal{X}$), the original *Metropolis algorithm* proceeds as follows. First, choose a symmetric d-dimensional increment distribution Q; the most common choice is $Q = N(0, \Sigma)$ for some fixed covariance matrix Σ like $\Sigma = c\,I_d$. (Here $N(0,\Sigma)$ is a d-dimensional normal (Gaussian) distribution, and I_d is the $d \times d$ identity matrix.) Also, choose some initial vector X_0. Then, iteratively for $n = 1, 2, 3, \ldots$, compute X_n from X_{n-1} by:

1. Let $Y_n = X_{n-1} + Z_n$, where $\{Z_n\} \sim Q$ are i.i.d. ("proposal")

2. Let $\alpha = \min\left(1,\ \frac{\pi(Y_n)}{\pi(X_{n-1})}\right)$. ("acceptance probability")

3a. With probability α, let $X_n := Y_n$. ("accept")

3b. Otherwise, with probability $1 - \alpha$, let $X_n := X_{n-1}$. ("reject")
(Steps 2, 3a, and 3b can all be accomplished by letting $U_n \sim \text{Uniform}[0,1]$, and setting $X_n = Y_n \mathbf{1}_{U_n \leq \frac{\pi(Y_n)}{\pi(X_{n-1})}} + X_{n-1} \mathbf{1}_{U_n > \frac{\pi(Y_n)}{\pi(X_{n-1})}}$. That is, we first sample $U_n \sim \text{Uniform}[0,1]$, and then if $U_n \leq \frac{\pi(Y_n)}{\pi(X_{n-1})}$ we accept the proposal and set $X_n = Y_n$, otherwise we reject it and set $X_n = X_{n-1}$.)

Intuitively, the above acceptance probabilities α are useful because they encourage the algorithm to accept more moves towards larger values of π. More precisely, the formula for α turns out to be exactly the right one to ensure that the Markov chain $\{X_n\}$ leaves the density π stationary, a key property for convergence (as discussed at the end of the previous section). Furthermore, the irreducibility property will almost always hold, e.g. it is guaranteed if Q has an everywhere-positive density like $N(0, \Sigma)$. And, the aperiodicity property is essentially never a problem since the algorithm will eventually reject and thus avoid cyclic behaviour. So, this simple algorithm has all the right properties to guarantee that $\mathbf{P}(X_n \in A) \to \pi(A)$. It follows that we can use this algorithm to estimate $\mathbf{E}_\pi(h)$ as in (4). Good!

The only problem is that sometimes the Metropolis algorithm will be too *inefficient*, i.e. it will take far too long (i.e., require too many iterations) to provide a decent approximation to π, which is a very important consideration (see e.g. Rosenthal, 1995). In some cases, even running the algorithm for months on the world's fastest computers would not provide a remotely reasonable approximation to π. Overcoming such problems has often necessitated new and more complicated MCMC algorithms (see e.g. Bélisle et al., 1993; Neal, 2003; Jain and Neal, 2004; Hamze and de Freitas, 2010). In a different direction, detecting convergence of MCMC to π is so challenging that some authors have developed *perfect sampling* algorithms which guarantee complete convergence at the expense of a more complicated algorithm (e.g. Propp and Wilson, 1996; Fill et al., 2000; Murdoch and Green, 1997); however, such perfect sampling algorithms are often infeasible to run, so we do not discuss them further here.

All of this raises the question of how to *improve* or *optimise* the speed of convergence of the Metropolis algorithm, for example by modifying the increment distribution Q, as we discuss next.

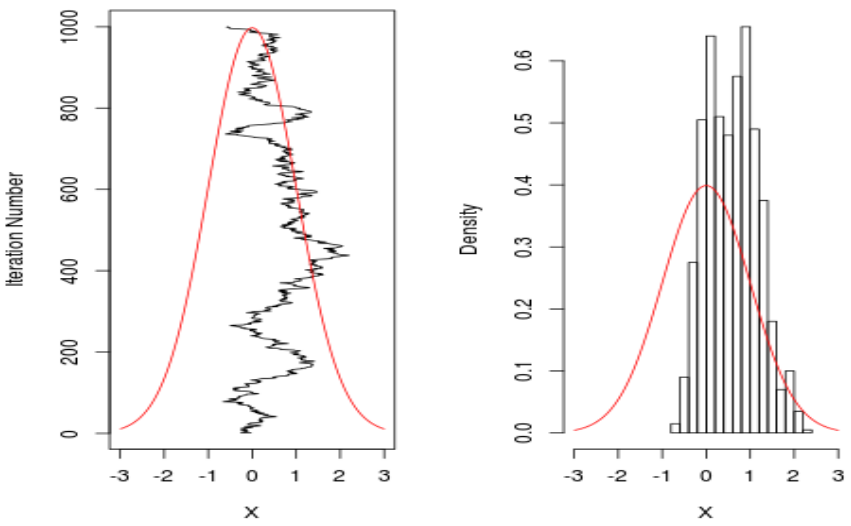

Figure 1: The trace plot (left) and histogram (right) and target density (red) for a one-dimensional Metropolis algorithm with too small a proposal scaling σ, showing slow mixing and poor convergence.

5 The Goldilocks Principle

To illustrate the Metropolis optimisation issues, consider the very simple case where $\pi = N(0, 1)$, i.e. where the target density is just the standard normal distribution. (Of course we wouldn't actually *need* to use MCMC in such a simple case.) Assume that the proposal distribution is given by $Q = N(0, \sigma^2)$. Our question of interest is, how should we choose σ?

As a first try, let's choose a small value of σ, say $\sigma = 0.1$, and run the Metropolis algorithm for 1000 iterations with that σ. The corresponding *trace plot*, graphing the values of the Markov chain (horizontal axis) at each iteration n (vertical axis), is shown in Figure 1 (left panel). Looking at this trace plot, we can see that the chain moves very *slowly*. It starts at the value zero, and takes many hundreds of iterations before it moves appreciably away from zero. In particular, it does not do a very good job of exploring the target density (shown in red). This is also illustrated by the histogram of those 1000 iterations, in Figure 1 (right panel), which does not match up very well with the target density.

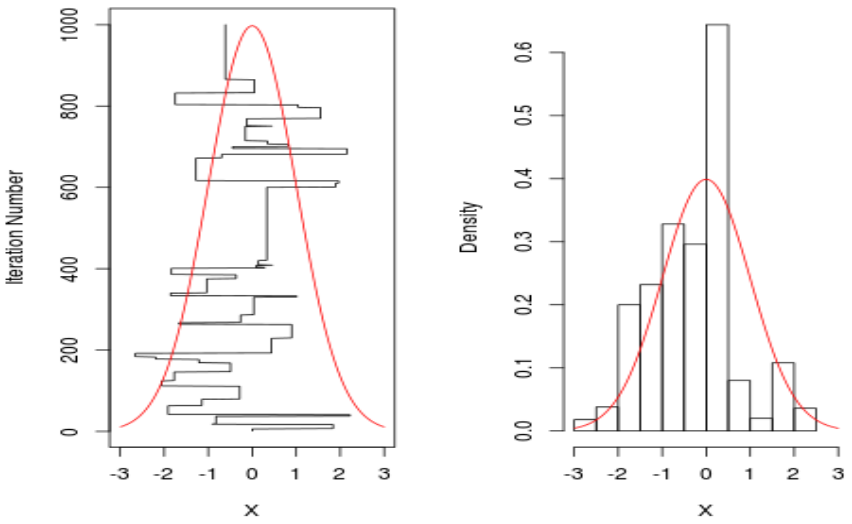

Figure 2: The trace plot (left) and histogram (right) and target density (red) for a one-dimensional Metropolis algorithm with too large a proposal scaling σ, again showing slow mixing and poor convergence.

As a second try, let's choose a large value of σ, say $\sigma = 25$, and again run the algorithm for 1000 iterations. The trace plot in this case is shown in Figure 2 (left panel). In this case, when the chain finally accepts a move, it jumps quite far which is good. However, since it proposes such large moves, it hardly ever accepts them. (Indeed, it accepted just 5.4% of the proposed moves, compared to 97.7% when $\sigma = 0.1$.) So, this chain doesn't perform very well either, as illustrated by the histogram in Figure 2 (right panel), which again does not match up very well with the target density.

As a third try, let's choose a compromise value of σ, say $\sigma = 2.38$, and again run the algorithm for 1000 iterations. In this case, the chain performs very well. It accepts a medium fraction of its proposals (44.5%), and moves reasonably far when it does accept. It thus explores the target density efficiently and well, as illustrated by the trace plot in Figure 3 (left panel). Furthermore it now provides fairly good samples from the target distribution, as illustrated by the histogram in Figure 3 (right panel).

We learn from this that it is best to choose values of the proposal increment scaling σ which are between the two extremes, i.e. not too small

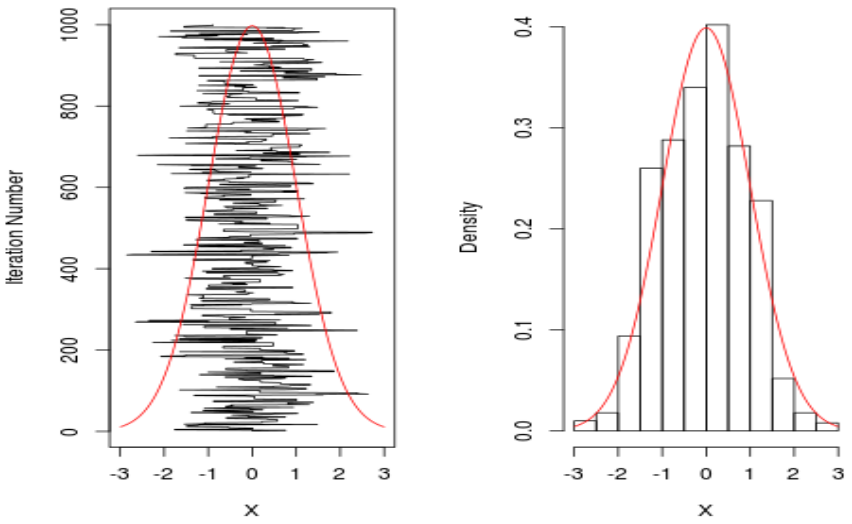

Figure 3: The trace plot (left) and histogram (right) and target density (red) for a one-dimensional Metropolis algorithm with a good choice of proposal scaling σ, showing much better mixing and convergence properties.

and not too big, but rather "just right" (as the little girl *Goldilocks* says in the classic children's fairy tale *The Three Bears*; Rhetorist, 2013; Wikipedia, 2013). Correspondingly, the acceptance rate (i.e., the percentage of proposed moves which are accepted) should be far from 0% but also far from 100%.

6 Optimal Scaling

The above intuition was made more precise in a pioneering paper by Roberts et al. (1997). They decided to consider a Metropolis algorithm $\{X_n\}$ in dimension d, with increment distribution $Q = N(0, \frac{\ell^2}{d} I_d)$ for some fixed scaling constant $\ell > 0$, and take the *limit* as $d \to \infty$. The beauty of their approach was that they could compute the *speed* of the algorithm in this limit, as an explicit (but messy and uncomputable) function $h(\ell)$ of the scaling constant ℓ. They then argued that *the best choice of ℓ is the one which maximises the limiting speed $h(\ell)$*. This provided a clear standard for how to optimise the Metropolis algorithm.

The story gets even better. Roberts et al. also considered the *asymptotic acceptance rate*, i.e. the fraction of Metropolis proposals that would be accepted in the limit as $d \to \infty$, and they computed an explicit function $A(\ell)$ for this as well. They then showed that *the limiting speed $h(\ell)$ has a simple relation to the asymptotic acceptance rate $A(\ell)$*. This in turn allowed them to compute that if ℓ_{opt} is the value of ℓ which maximises the speed $h(\ell)$, then $A(\ell_{opt}) \approx 0.234 = 23.4\%$, a specific number that does not depend on any unknown quantities about π or anything else. This means that, at least under their (strong) assumptions, *the optimal acceptance probability is 23.4%, which leads to the fastest limiting speed regardless of the target density π*.

This provides a clear, simple rule for tuning Metropolis algorithms: adjust the proposal scaling ℓ so that the acceptance rate is approximately 23.4%. This rule appears to be quite robust, i.e. 0.234 is often a *nearly* optimal acceptance rate even if the theorem's formal assumptions are not satisfied. It has been implemented in numerous applied papers and software, including the hugely popular *WinBUGS* computer package (Lunn et al., 2000). A number of authors have attempted to weaken and generalise the original strong assumptions, see e.g. Bédard (2007, 2008), Bédard and Rosenthal (2008), Beskos et al. (2009), and Sherlock and Roberts (2009). Corresponding results have been developed for Langevin MCMC algorithms (Roberts and Rosenthal, 1998), and for simulated tempering algorithms (Atchadé et al., 2011; Roberts and Rosenthal, 2012). It is also known (e.g. Roberts and Rosenthal, 2001, Figure 3) that any acceptance rate between about 15% and 50% is still reasonably efficient (though 23.4% is best). Overall this school of research has been very influential in guiding both applied usage and theoretical investigations for MCMC.

7 Proposal Shape

Despite the 0.234 rule's great success, an algorithm's acceptance rate is just a single scalar quantity which does not completely control the algorithm's efficiency.

To illustrate this, consider the 20-dimensional target density $\pi = N(0, \Sigma_*)$, where Σ_* is a 20-dimensional covariance matrix generated randomly as $\Sigma_* = M^\top M$ where M is a 20×20 matrix consisting of i.i.d. $N(0, 1)$ entries, which shall remain fixed throughout the remainder of this article, and $M^\top$ is the transpose of M. We shall try running Metropolis algorithms on this density.

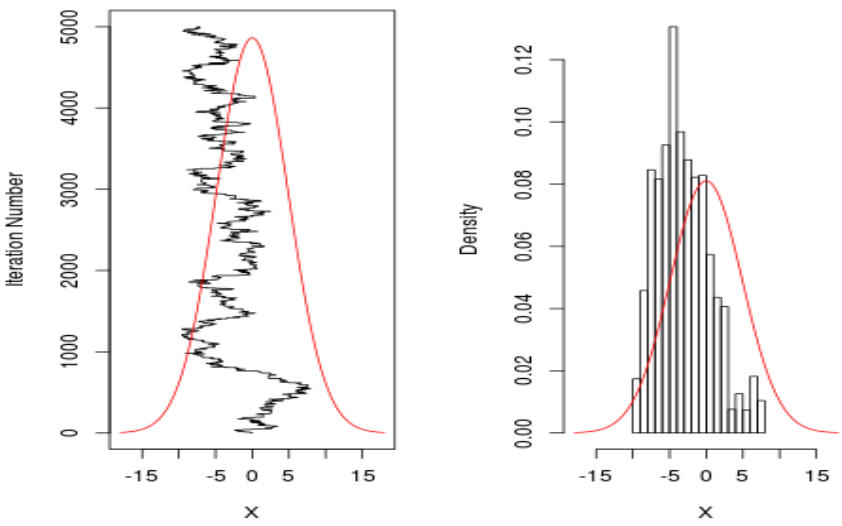

Figure 4: The trace plot (left) and histogram (right) and target density (red) of the first coordinate of a twenty-dimensional Metropolis algorithm with proposal scaling proportional to the identity matrix, showing slow mixing and poor convergence.

Based on the discussion in Section 6, we use a increment distribution of the form $Q = N(0, \sigma^2 I_{20})$, where we try to adjust σ so that the resulting acceptance rate is approximately 0.234. After some experimenting, we take $\sigma = 0.5$, leading to an acceptance rate of 0.228 (close enough). Figure 4 shows the trace plot (left panel) and histogram (right panel) of the *first coordinate* of this run. The mixing and convergence aren't *so* bad, given the relatively large dimension, but they aren't great either. Indeed, the performance appears to be similar to our earlier one-dimensional example's first attempt with $\sigma = 0.1$: it only *slowly* explores the support of π.

Next, we instead try the increment distribution $Q = N(0, [(2.38)^2/20]\,\Sigma_*)$ (which we shall justify later). This leads to an acceptance rate of 0.252 (again, close enough). Figure 5 shows the trace plot (left panel) and histogram (right panel) of the first coordinate of this run. Direct inspection (as well as more precise measurements like squared jumping distance and functional variance, not discussed here) indicate that this choice of Q, despite leading to a very similar acceptance rate (approximately 0.234), actually performs much

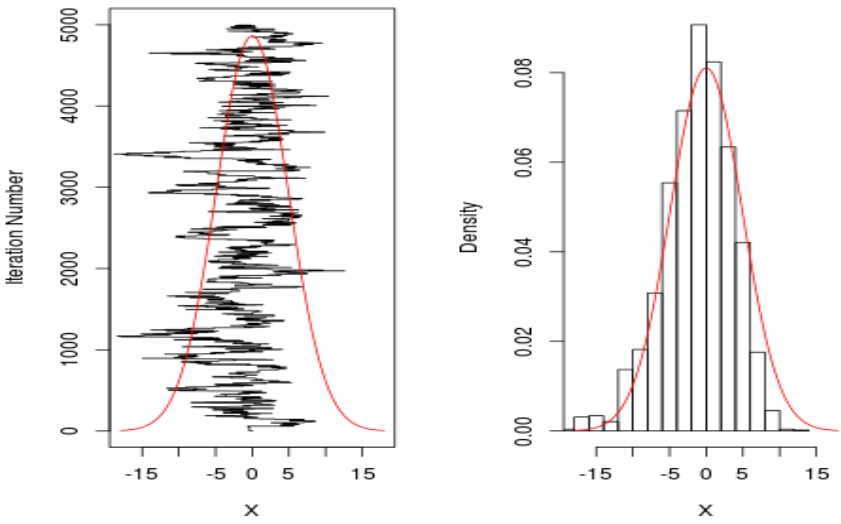

Figure 5: The trace plot (left) and histogram (right) and target density (red) of the first coordinate of a twenty-dimensional Metropolis algorithm with proposal scaling proportional to the target covariance matrix Σ_*, showing much faster mixing and much better convergence properties.

better, exploring the support of π and converging to the correct probabilities much more efficiently. This confirms that there is more to the story than just acceptance rate, and indeed that the *shape* of the proposal distribution (determined by the proposal covariance matrix, in this case $[(2.38)^2/20]\,\Sigma_*$) is also very important.

This concept was formalised by Roberts and Rosenthal (2001). They proved that under strong assumptions (similar to before), the optimal Gaussian proposal distribution is given (to three significant figures) by

$$Q \;=\; N\Big(0,\; [(2.38)^2/d]\,\Sigma_\pi\Big)$$

(where Σ_π is the $d \times d$ covariance matrix of the target density π), rather than $Q = N(0,\, \sigma^2 I_d)$ for some σ^2. Furthermore, with this choice, the asymptotic acceptance rate will again be approximately 0.234. And, as before, this result appears to be robust in the sense of being *nearly* optimal even when the strong assumptions do not hold.

8 Adaptive MCMC

The optimisation result of the previous section requires us to know and use the target covariance matrix Σ_π. Now, in most realistic situations, Σ_π would not be known in advance, and indeed would be at least as difficult to estimate as the quantity $\mathbf{E}_\pi(h)$ of ultimate interest. How can we optimise the Metropolis algorithm in such situations?

In a pioneering paper, Haario et al. (2001) proposed to optimise the algorithm *adaptively*. That is, even if we don't know the optimal algorithm at the beginning, the computer can *learn* it during the run, and *update* the algorithm "on the fly" as it proceeds.

In its simplest form, this algorithm finds an approximately optimal increment distribution by replacing the unknown target $d \times d$ covariance matrix Σ_π by the sample $d \times d$ covariance $\Sigma_n := \mathrm{Cov}(X_0, X_1, \ldots, X_{n-1})$ of the vectors visited so far during the run. If those X_i are indeed good approximate samples from π, then Σ_n will be a good approximation to Σ_π, and hence $Q = N(0,\ ((2.38)^2/d)\,\Sigma_n)$ can be used (after an initial phase, e.g. for $n \geq 40$ only) as a good approximation to the optimal proposal $Q = N(0,\ ((2.38)^2/d)\,\Sigma_\pi)$. If not, then the algorithm will not work well initially, but it will hopefully improve as it goes.

Now, Σ_n is easily computed, so this algorithm is quite feasible to run in practice. Running it on the above 20-dimensional target density $\pi = N(0, \Sigma_*)$, the resulting trace plot and histogram of the first coordinate are shown in Figure 6. Direct inspection, and precise measurements, both indicate that this algorithm performs poorly during the initial phase (bottom of left plot), but then performs very well later on (top of left plot), exploring the support of π nearly as efficiently as the optimal algorithm presented above, even though it does not require any prior knowledge about Σ_π.

Adaptive MCMC algorithms have recently been used in a number of different statistical applications, and often lead to significant speed-ups, even in hundreds of dimensions (Roberts and Rosenthal, 2009; Craiu et al., 2009; Giordani and Kohn, 2010; Richardson et al., 2011). On the other hand, adaptive MCMC algorithms use previous iterations to determine their future transitions, so they violate the *Markov property* which provides the justification for conventional MCMC. This raises the question of whether adaptive MCMC algorithms are valid, i.e. whether they converge (asymptotically, at least) to the target density π.

The answer to this question is no in general (see e.g. Rosenthal, 2004),

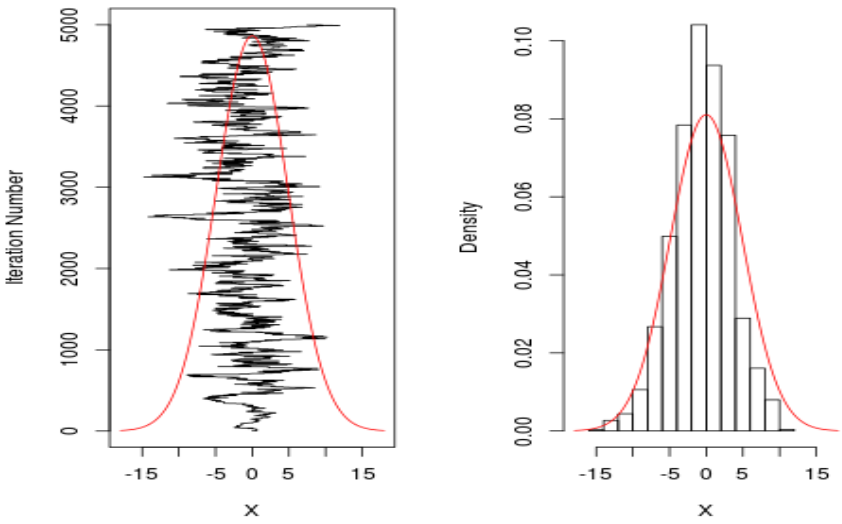

Figure 6: The trace plot (left) and histogram (right) and target density (red) of the first coordinate of a twenty-dimensional adaptive Metropolis algorithm with proposal scaling computed from previous iterations, showing fairly rapid mixing and good convergence properties, especially in later iterations (top of left plot).

but it is yes under various conditions (Haario et al., 2001; Atchadé and Rosenthal, 2005; Andrieu and Moulines, 2006; Roberts and Rosenthal, 2007; Saksman and Vihola, 2010; Fort et al., 2012; Latuszynski et al., 2011). In particular, Roberts and Rosenthal (2007) show that adaptive algorithms will still converge to the target density π provided they satisfy two fairly mild conditions: "Diminishing Adaptation" (the algorithm adapts by less and less as time goes on), and "Containment" (the chain never gets *too* lost, in the sense that it remains bounded in probability). Conditions such as these have been used to formally justify adaptive algorithms in many examples (e.g. Roberts and Rosenthal, 2009; Richardson et al., 2011). Adaptive MCMC appears to hold great promise for improving statistical computation in many application areas in the years ahead.

9 Summary

We summarise the points made in this article as follows.

- The Metropolis algorithm is very important in numerous applications.
- This algorithm sometimes runs so slowly and inefficiently that computations are infeasible; thus, *optimisation* of the algorithm can be crucial.
- The simplest optimisation result is the *Goldilocks Principle* that the acceptance rate should be far from 0, but also far from 1.
- A more detailed theorem says that the optimal acceptance rate is 0.234, at least under certain strong assumptions (though the conclusion appears to be fairly robust even when the assumptions are not satisfied).
- Another theorem says that the optimal increment distribution is $N(0,\ (2.38)^2\Sigma_\pi\,/\,d)$, again under certain strong assumptions (though again with a fairly robust conclusion).
- When certain key optimisation information is unknown (e.g., Σ_π), it may still be possible to *adapt* towards the optimal algorithm. Such adaption is not valid in general, but is valid under various conditions such as "Diminishing Adaptation" and "Containment". It can eventually lead to tremendous speed-ups, even in very high dimensions.
- In short, to greatly improve statistical computation in important applied areas, optimisation and adaption may well be worth the trouble!

Acknowledgements. I thank the editors and reviewers for many detailed comments.

References

(Note: Canadian authors are indicated with **[C]**.)

J. Albert (1992), A Bayesian analysis of a Poisson random effects model for home run hitters. The American Statistician **46(4)**, 246–253.

C. Andrieu and E. Moulines (2006), On the ergodicity properties of some adaptive Markov Chain Monte Carlo algorithms. Ann. Appl. Prob. **16**, 1462–1505.

Y. Atchadé**[C]** and G. Fort (2010), Limit Theorems for some adaptive MCMC algorithms with sub-geometric kernels. Bernoulli **16**, 116–154.

Y. Atchadé[**C**], G.O. Roberts, and J.S. Rosenthal[**C**] (2011), Towards Optimal Scaling of Metropolis-Coupled Markov Chain Monte Carlo. Stat. and Comput. **21(4)**, 555–568.

Y.F. Atchadé[**C**] and J.S. Rosenthal[**C**] (2005), On Adaptive Markov Chain Monte Carlo Algorithms. Bernoulli **11(5)**, 815–828.

Y. Bai[**C**], G.O. Roberts, and J.S. Rosenthal[**C**] (2011), On the containment condition for adaptive Markov chain Monte Carlo algorithms. Adv. Appl. Stat. **21**, 1–54.

A. Bartolucci, K. Singh, and S.J. Bae (2007), Analyzing Clinical Trial Data via the Bayesian Multiple Logistic Random Effects Model. In L. Oxley and D. Kulasiri (eds), MODSIM 2007 conference proceedings, Modelling and Simulation Society of Australia and New Zealand.

M. Bédard[**C**] (2007), Weak convergence of Metropolis algorithms for non-iid target distributions. Ann. Appl. Prob. **17**, 1222–1244.

M. Bédard[**C**] (2008). Optimal acceptance rates for Metropolis algorithms: moving beyond 0.234. Stoch. Proc. Appl. **118**, 2198–2222.

M. Bédard[**C**] and J.S. Rosenthal[**C**] (2008). Optimal scaling of Metropolis algorithms: heading toward general target distributions. Can. J. Stat. **36**, 483–503.

C. Bélisle[**C**], H.E. Romeijn, and R.L. Smith (1993), Hit-and-Run algorithms for generating multivariate distributions. Math. Oper. Res. **18**, 255–266.

A. Beskos, G.O. Roberts, and A.M. Stuart (2009), Optimal scalings of Metropolis-Hastings algorithms for non-product targets in high dimensions. Ann. Appl. Prob. **19**, 863–898.

C.E.P. Box and G.C. Tiao (1973), Bayesian Inference in Statistical Analysis. Addison-Wellesley, Reading, Massachusetts.

B.P. Carlin and T.A. Louis (2008), Bayesian Methods for Data Analysis, 3rd ed. Chapman & Hall, New York.

R.V. Craiu[**C**], J.S. Rosenthal[**C**], and C. Yang[**C**] (2009), Learn From Thy Neighbor: Parallel-Chain Adaptive MCMC. J. Amer. Stat. Assoc. **488**, 1454–1466.

M. Evans[**C**] and T. Swartz[**C**] (2000), Approximating Integrals via Monte Carlo and Deterministic Methods. Oxford University Press.

J.A. Fill, M. Machida, D.J. Murdoch[**C**], and J.S. Rosenthal[**C**] (2000), Extension of Fill's perfect rejection sampling algorithm to general chains. Random Struct. Alg. **17**, 290–316.

G. Fort, E. Moulines, and P. Priouret (2012). Convergence of adaptive and interacting Markov chain Monte Carlo algorithms. Ann. Stat. **39**, 3262–3289.

A.E. Gelfand and A.F.M. Smith (1990), Sampling based approaches to calculating marginal densities. J. Amer. Stat. Assoc. **85**, 398–409.

A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin (2003), Bayesian Data Analysis (2nd ed.). Chapman & Hall, New York.

P. Giordani and R. Kohn (2010), Adaptive independent Metropolis-Hastings by fast estimation of mixtures of normals. J. Comp. Graph. Stat. **19(2)**, 243–259.

H. Haario, E. Saksman, and J. Tamminen (2001), An adaptive Metropolis algorithm. Bernoulli **7**, 223–242.

F. Hamze[**C**] and N. de Freitas[**C**] (2010), Intracluster Moves for Constrained Discrete-Space MCMC. Proc. 26th Annual Conf. on Uncertainty in Artificial Intel. (UAI-10), 236–243.

W.K. Hastings[**C**] (1970), Monte Carlo sampling methods using Markov chains and their applications. Biometrika **57**, 97–109.

S. Jain[**C**] and R.M. Neal[**C**] (2004), A Split-Merge Markov Chain Monte Carlo Procedure for the Dirichlet Process Mixture Model. J. Comp. Graph. Stat. **13**, 158–182.

K. Latuszynski, G.O. Roberts, and J.S. Rosenthal[**C**] (2011), Adaptive Gibbs samplers and related MCMC methods. Ann. Appl. Prob., to appear.

D.J. Lunn, A. Thomas, N. Best, and D. Spiegelhalter (2000), WinBUGS – a Bayesian modelling framework: concepts, structure, and extensibility. Stat. and Comput. **10**, 325–337.

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller (1953), Equations of state calculations by fast computing machines. J. Chem. Phys. **21**, 1087–1091.

D.J. Murdoch[**C**] and P.J. Green (1997), Exact sampling from a continuous state space. Scand. J. Stat. **25**, 483–502.

R.M. Neal[**C**] (2003), Slice sampling (with discussion). Ann. Stat. **31**, 705–767.

J.G. Propp and D.B. Wilson (1996), Exact sampling with coupled Markov chains and applications to statistical mechanics. Random Struct. Alg. **9**, 223–252.

T. Rhetorist (retrieved May 8, 2013), The Three Bears (illustrated version). Available at: http://w8r.com/the-colorful-story-book/the-three-bears

S. Richardson, L. Bottolo, and J.S. Rosenthal[**C**] (2011), Bayesian models for sparse regression analysis of high dimensional data. *Bayesian Statistics 9*

conference proceedings, Oxford University Press, 539–560.

G.O. Roberts, A. Gelman, and W.R. Gilks (1997), Weak convergence and optimal scaling of random walk Metropolis algorithms. Ann. Appl. Prob. **7**, 110–120.

G.O. Roberts and J.S. Rosenthal[**C**] (1998), Optimal scaling of discrete approximations to Langevin diffusions. J. Roy. Stat. Soc. B **60**, 255–268.

G.O. Roberts and J.S. Rosenthal[**C**] (2001), Optimal scaling for various Metropolis-Hastings algorithms. Stat. Sci. **16**, 351–367.

G.O. Roberts and J.S. Rosenthal[**C**] (2007), Coupling and Ergodicity of Adaptive MCMC. J. Appl. Prob. **44**, 458–475.

G.O. Roberts and J.S. Rosenthal[**C**] (2009), Examples of adaptive MCMC. J. Comp. Graph. Stat. **18(2)**, 349–367.

G.O. Roberts and J.S. Rosenthal[**C**] (2012), Minimising MCMC variance via diffusion limits, with an application to simulated tempering. Ann. Appl. Prob., tentatively accepted.

J.S. Rosenthal[**C**] (1995), Minorization conditions and convergence rates for Markov chain Monte Carlo. J. Amer. Stat. Assoc. **90**, 558–566.

D.B. Rubin (1980), Using empirical Bayes techniques in law school validity studies (with discussion). J. Amer. Stat. Assoc. **75**, 801–827.

E. Saksman and M. Vihola (2010), On the ergodicity of the adaptive Metropolis algorithm on unbounded domains. Ann. Appl. Prob. **20**, 2178-2203.

C. Sherlock and G.O. Roberts (2009), Optimal scaling of the random walk Metropolis on elliptically symmetric unimodal targets. Bernoulli **15(3)**, 774–798.

Wikipedia (retrieved May 8, 2013), The Story of the Three Bears. Available at: http://en.wikipedia.org/wiki/The_Story_of_the_Three_Bears