International Journal of Statistical Sciences *ISSN 1683–5603*
Vol. 3 (Special Issue), 2004, pp 221–239

Comparison of two Markov chain Monte Carlo (MCMC) methods

Azad Mahbubur Rashid Kazi
Department of Statistics
Shah Jalal University of Science & Technology
Sylhet 3114, Bangladesh.

[Received April 12, 2004; Accepted August 17, 2004]

Abstract

As the world advances, statisticians/mathematicians are being involved into more and more complex surveys for the development of society and human beings. Consequently, these complex survey data requires complicated and high-dimensional models for final analysis. We need sophisticated and efficient statistical/mathematical tools for estimation and prediction of these models. Frequently, we simulate samples from these complicated models to estimate parameters because direct estimation is sometimes not efficient. Markov chain Monte Carlo (MCMC) methods have been developed for simulation and efficient estimation. In these times of availability of high-speed computing facilities, MCMC methods are popular tools to generate samples from these complex and high-dimensional distributions. There are some MCMC methods available for application, for instance, Gibbs sampler, Metropolis-Hastings algorithm, etc. In this paper we compare the performance of two MCMC methods, namely the Hybrid (HY) algorithm and the Random Walk Metropolis-Hastings (M-H) algorithm, by employing two logistic regression models and a multivariate normal distribution. We get data used in one of the examples from a Prostate Cancer Study presented in a book by D.W. Hosmer and S. Lemeshow. The Bayesian approach is followed to fit the two logistic regression models. Evaluation of performance is based on convergence of Markov chains, efficiency of estimation and Monte Carlo error variances of the two methods.

Keywords and Phrases: Bayesian inference, Markov Chain Monte Carlo methods, simulation, logistic regression, Markov chains, stationary distributions, convergence.

AMS Classification: 62F15, 62C05, 37M05, 62Jxx, 65C40.

1 Introduction

In recent times, *Markov chain Monte Carlo* (MCMC) methods are being used extensively to generate samples from complicated, high-dimensional distributions. In many situations, the target distributions do not have closed mathematical conditional distributions, especially in Bayesian framework. MCMC techniques provide an answer to the difficult problem of simulation from the high-dimensional distribution of the unknown quantities that appear in complex models (Gamerman, 1997; Gilks *et. al.*, 1998). MCMC is essentially Monte Carlo integration using Markov chains. We need to integrate over the target posterior distribution of model parameters given the data. Monte Carlo integration draws samples from the required distribution until it approaches equilibrium, known as the limiting distribution, and then forms sample averages to approximate expectations. So our limiting distribution is usually the posterior distribution when we apply Bayesian statistics or could be any other distribution. *Markov chain Monte Carlo* (MCMC) draws samples by running a cleverly constructed Markov chain for a long time.

Some MCMC algorithms use more information about the posterior distribution or any other target distribution than others; generally one expects algorithms that use more information to be more efficient. The Gibbs sampler (Geman and Geman, 1984; Gelfand and smith, 1990) requires draws from the posterior conditional distribution of each parameter, given the other parameters. Hybrid algorithm (Duane *et. al.*, 1987; Neal, 1993) requires evaluation of the log un-normalized posterior density along with its first partial derivatives. In contrast, simpler forms of the Metropolis algorithm (Metropolis *et.al.*, 1953; Hastings, 1970) require only un-normalized posterior density evaluations. For instance, the random walk Metropolis algorithm operates by proposing that the Markov chain move to a candidate state obtained by adding noise to the current state. This algorithm is commonly used in practice, often to update a few 'tricky' parameter components that are not amenable to Gibbs sampling. Before discussing these two selected algorithms, namely the Hybrid (HY) algorithm and the Random Walk Metropolis-Hastings (M-H) algorithms in detail, we are going to define Bayes' estimator and Monte Carlo integration which will be useful to understand the methodologies of this paper.

1.1 Bayes' Estimator

A Bayes' estimator of θ, say, is the mean of the posterior distribution of $\theta \sim p(\theta|y)$, say, called the posterior expectation, i.e.

$$\begin{aligned} \hat{\theta}_B &= E(\theta|y) \\ &= \int \theta p(\theta|y) d\theta \\ &= \frac{\int \theta p(\theta) p(y|\theta) d\theta}{\int p(\theta) p(y|\theta) d\theta}, \end{aligned} \tag{1}$$

where $p(\theta)$ is the prior distribution of θ, and $p(y|\theta)$, is the sampling distribution of the observed data y.

The integrations in (1) have until recently been the source of most of the practical difficulties in Bayesian inference, especially in high dimensions. In most applications, analytic evaluation of $E(\theta|y)$ is impossible. The best alternative way of evaluation is the MCMC.

1.2 Monte Carlo Integration

Let X be a vector of k random variables, with distribution $\pi(.)$, where X consists of model parameters and missing data. Our task is to evaluate the expectation

$$E[f(X)] = \frac{\int f(x)\pi(x)dx}{\int \pi(x)dx},$$

for some function of interest $f(.)$. Monte Carlo integration evaluates $E[f(x)]$ by drawing samples $\{X_t, t = 1, 2, \ldots, n\}$ from $\pi(.)$ and then approximating

$$E(f(X)] \approx \frac{1}{n}\sum_{t=1}^{n} f(X_t).$$

So the population mean of $f(X)$ is estimated by a sample mean. When the samples $\{\mathrm{X}_t\}$ are independent, the laws of large numbers ensure that the approximation can be made as accurate as desired by increasing the sample size n.

In general, drawing samples $\{\mathrm{X}_t\}$ independently from $\pi(.)$ is difficult. However, the samples $\{\mathrm{X}_t\}$ need not necessarily be independent. They can be generated by any process, which draws samples throughout the support of $\pi(.)$ in the correct proportions. One way of doing this is through a Markov chain having $\pi(.)$ as its stationary distribution. This is then *Markov chain* Monte Carlo.

2 The Metropolis-Hastings (M-H) algorithm

The Metropolis-Hastings algorithm is the fundamental building block of most MCMC algorithms. Given a target distribution Π, which would be the posterior distribution in Bayesian applications, we wish to construct a Markov chain $\{X_i\}_{i=0}^{\infty}$ with Π as its stationary distribution. If $X_n = x_n$is the current state of the chain, then the Metropolis-Hastings algorithm proceeds by simulating a candidate or proposal value y from a transition density, $q(x, .)$. The next state X_{n+1}, is then randomly assigned to be either y with probability $\alpha(x_n, y)$, or x_n with probability $1 - \alpha(x_n, y)$, where

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

is the acceptance probability. By choosing different proposal transition densities $q(.,.)$ we obtain different MCMC algorithms, including Gibbs sampler, the Langevin algorithm and the random walk Metropolis algorithm amongst others.

In the case that Π is a continuous univariate distribution on R, the random walk Metropolis algorithm works as follows. The candidate state is obtained by adding noise to the current state; specifically, $q(x,y) = f(y-x)$, for some density f which is symmetric about zero. Commonly f is taken to be normal (mean zero and variance σ^2), in which case the algorithm for updating from $X_n = x_n$ to $X_{n+1} = x_{n+1}$ can be expressed as:

$$\begin{aligned} y &\leftarrow x_n + z, \text{ where } z \sim N(0,\sigma^2) \\ \alpha &\leftarrow \min\left(\frac{\pi(y)}{\pi(x_n)}, 1\right) \\ x_{n+1} &\leftarrow \begin{cases} y & \text{with probability } \alpha, \\ x_n & \text{with probability } 1-\alpha. \end{cases} \end{aligned}$$

Note that the symmetry property of the proposal transition, $q(x,y) = q(y,x)$, leads to a simple form for the acceptance probability.

This method randomly searches for regions of high probability, that is, the direction in which the Markov chain (MC) attempts to move is randomized at each transition. This behavior could be inefficient, because it might take many iterations to converge to the target stationary distribution. For some target distributions the MC might not converge after many iterations and we may not get the desired sampler output. A natural modification of this behavior is to bias the noise distribution in favor of candidate states which lies 'uphill' from the current state. Hybrid (HY) algorithm is the modification of random walk M-H algorithm in which the following two ideas are implemented:

(i) to incorporate derivative evaluations of the target density to use more information, and

(ii) to suppress random walk behavior of the MC, that is, to avoid the random search for high probability regions we attempt to force moves in more consistent manner for faster convergence and efficiency.

3 Hybrid (HY) algorithm

Let $X \sim \Pi$ be the target density having an unnormalized density function $\pi(x)$ on a subset of $\Re^k$. The algorithm works by extending the state from X to (X,Y), and the

unnormalized target density from $\pi(x)$ to

$$\begin{aligned}\pi(x,y) &= \pi(x)\pi(y) \\ &= \pi(x)\exp\left(-\frac{1}{2}\sum_{i=1}^{k} y_i^2\right) \end{aligned} \tag{2}$$

where Y has a $N(0, I_k)$ distribution independent of X. Thus we can sample from $\pi(x)$ by sampling (X, Y) from (2) and simply discarding the Y values.

The following three steps should be followed to construct a Markov chain for (X, Y) having (2) as its stationary distribution. Also it is necessary to specify a step size $\varepsilon > 0$, a function $g : \Re^k \to \Re^k$, and a constant $\delta \in [0, 1)$.

1. Determine a candidate state (x^*, y^*) as

$$\begin{aligned} x^* &\leftarrow x + \varepsilon[y + (\varepsilon/2)g(x)], \\ y^* &\leftarrow -y - (\varepsilon/2)[g(x) + g(x^*)], \end{aligned}$$

   and randomly assign

$$(x, y) \leftarrow \begin{cases} (x^*, y^*) & \text{with probability } p, \\ (x, y) & \text{with probability } 1 - p, \end{cases}$$

   where

$$p = \min\left\{\frac{\pi(x^*, y^*)}{\pi(x, y)},\ 1\right\}.$$

2. Unconditionally negate y, i.e.

$$y \leftarrow -y$$

3. Perform an autoregressive update to y, i.e.

$$y \leftarrow N(\delta y, (1 - \delta^2)^{1/2} I_k).$$

We will choose $g(x) = \Delta \log \pi(x)$, the gradient vectors of the parameters from the target distribution and δ close to one to suppress the random walk behavior of the MC.

The above random walk M-H and HY algorithms can be applied in a univariate fashion, to update individual components of a multivariate state vector one-by-one, or in a multivariate fashion, to update all the components simultaneously. They are sensitive to choice of step sizes. Tuning is based on trial and error. To get a reasonable acceptance rate while ensuring a proper mixing of the sampler output, step sizes should be tuned very carefully.

Now to evaluate the performances of these two algorithms, firstly we draw samples from the log posterior densities of Ordinary Logistic Regression Model (OLRM) and a modified Logistic Regression Model. Bayesian methods are followed for estimation. Then we compare the two methods on the basis of acceptance rate of samples and convergence of the respective Markov chains. In the next two sections, we introduce briefly the two logistic regression models.

4 Ordinary Logistic Regression

Let Y be a binary response variable indicating presence ($Y = 1$) and absence ($Y = 0$) of a disease. Let $X = (X_1, X_2, \ldots, X_p)$ denote a p explanatory variables or risk factors. The X's may be continuous or binary. For simplicity, we assume that X's are binary. Further assume $\beta = (\beta_0, \beta_1, \beta_2, \cdots, \beta_p)$ are regression coefficients to be estimated. The linear logistic regression model in the logit scale can be written as:

$$\begin{aligned} \text{logit } \Pr(Y = 1|X) &= \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p, \\ &= g(x, \beta), \quad \text{say} \end{aligned}$$

where

$$\begin{aligned} \Pr(Y = 1|X) &= \pi(x) \\ &= \frac{1}{1 + e^{-g(X,\beta)}}. \end{aligned}$$

4.1 Likelihood Function and the Posterior distribution

Suppose that we have total n number of cases in the study. The responses $y_1, y_2, \cdots, y_n$ are the observed values of independent random variables $Y_1, Y_2, \cdots, Y_n$. The Y_i's, for $i = 1, 2, \cdots, n$, are distributed as binomial with index m_i, the number of observations in each group and parameter π_i. We assume $m_i = 1$. Let us assume a diffuse prior for β, that is, $\pi(\beta) \propto 1$. The likelihood function may be written in the form:

$$L(\beta) = \prod_{i=1}^{n} \pi(x_i)^{y_i} [1 - \pi(x_i)]^{1-y_i}.$$

Therefore, the posterior distribution is: $P(\beta|y, x) \propto L(\beta)\, \pi(\beta)$.

Rearranging terms, we get the log posterior distribution as:

$$\log P(\beta|y, x) = K + \sum_{i=1}^{n} \left[y_i \left(\sum_{j=0}^{p} \beta_j x_{ij} \right) - \log \left\{ 1 + \exp \left(\sum_{j=0}^{p} \beta_j x_{ij} \right) \right\} \right], \quad (3)$$

where K is an unknown constant and $x_{i0} = 1$. As before let $g(x, \beta) = \sum_{j=0}^{p} \beta_j x_{ij}$. For implementing the HY algorithm we need to evaluate derivatives of (3) with respect to $\beta_0, \beta_1, \beta_2, \cdots, \beta_p$. The derivatives are computed as follows:

$$\begin{aligned} \frac{\delta \log P}{\delta \beta_0} &= \sum_{i=1}^{n} \left[y_i - \frac{\exp(g(x, \hat{\beta}))}{1 + \exp(g(x, \hat{\beta}))} \right] \\ &= \sum_{i=1}^{n} (y_i - \hat{\pi}_i), \end{aligned}$$

where $\hat{\pi}_i = \frac{\exp(g(x,\hat{\beta}))}{1+\exp(g(x,\hat{\beta}))}$. Similarly,

$$\frac{\delta \log P}{\delta \beta_j} = \sum_{i=1}^{n} (y_i - \hat{\pi}_i) x_{ij} \text{ for } j = 1, 2, \cdots, p.$$

5 New Logistic Regression Model

The ordinary logistic regression model is extended to include non-additive interaction effects. Let us assume that the probability of presence of a disease is a non-linear function of the risk factors in the logit scale. Then the non-linear regression model is defined as:

$$\text{logit } \Pr(Y = 1|X) = \beta_0 + \{\beta_1^{\lambda} X_1 + \cdots + \beta_p^{\lambda} X_p\}^{1/\lambda}, \tag{4}$$

where λ is an additional parameter.

The likelihood function can be obtained by following the previous procedure. We re-parameterize λ by defining $\phi = log\lambda$ for numerical simplicity. To get the posterior distribution we assume a diffuse prior for β and a normal prior for ϕ, that is, $\pi(\phi) \sim N(0, c^2)$. To avoid some numerical complexities associated with enormous λ values we assume $c = \log(2)$. The joint posterior distribution is defined as:

$$P(\beta, \phi|y) \propto L(\beta, \phi)\ \pi(\beta)\ \pi(\phi).$$

Therefore the log posterior distribution is:

$$\begin{aligned} \log P(\beta, \phi|y) &= K + \sum_{i=1}^{n} y_i \left[\beta_0 + \left(\sum_{j=1}^{p} \beta_j^{e^{\phi}} x_{ij} \right)^{e^{-\phi}} \right] \\ &\quad - \sum_{i=1}^{n} \log \left[1 + \exp \left\{ \beta_0 + \left(\sum_{j=1}^{p} \beta_j^{e^{\phi}} x_{ij} \right)^{e^{-\phi}} \right\} \right] - \frac{1}{2c^2} \phi^2, \end{aligned} \tag{5}$$

where K is an unknown constant.

The derivatives of the new logistic regression model can be obtained by differentiating Equation (5) with respect to the parameters. They are omitted here and given in the appendix. In the next section we draw samples from Equations (3) and (5) by using both random walk M-H and HY algorithms, and compare their efficiency.

6 Example 1: The Prostate Cancer Study

This example is taken from the book "Applied Logistic Regression", by Hosmer and Lemeshow, John Wiley & Sons, 2^{nd} Edition (2000). In this study data was collected from 380 male patients who had cancer of the prostate.

The goal of the study was to determine whether variables measured at a baseline examination could be used to predict whether the tumor has penetrated the prostatic capsule. Of these 380 subjects considered, 153 had a cancer that penetrated the prostatic capsule. The description of the measured variables is given in the following table 1.

Table 1: Code Sheet for the Prostate Cancer Study.

Variable #	Variable Description	Codes/Values	Name
1	Tumor Penetration of the Prostatic Capsule	0 = No Penetration 1 = Penetration	CAPSULE
2	Age	Years	AGE
3	Race	1 = White, 2 = Black	RACE
4	Results of the Digital Rectal Examination	1 = No Nodule 2 = Unilobar Nodule (Left) 3 = Unilobar Nodule (Right) 4 = Bilobar Nodule	DPROS
5	Detection of Capsular involvement in Rectal Exam.	1 = No 2 = Yes	DCAPS
6	Prostatic Specific Antigen Value	mg/ml	PSA
7	Tumor Volume Obtained from Ultrasound	cm^3	VOL
8	Total Gleason Score	0 – 10	GLEASON

Here CAPSULE is the response variable. There are seven risk factors measured in this study; two of them are binary and the remaining five are continuous. To make this data compatible with the new logistic regression model we convert the five continuous risk factors into binary variables by thresholding in comparison to their means.

Firstly, we draw samples from (3) by using the random walk M-H algorithm by updating parameters component-wise. Since we have eight parameters for this data including the intercept, the step/jump sizes used are 0.20, 0.25, 0.25, 0.25, 0.35, 0.25, 0.25, and 0.30, respectively. The initial values are taken to be 0.001 for all eight parameters. After iterating the algorithm 15,000 times the acceptance rates were around 60% and the chain was stabilized as can be seen from Figure 1 which shows the sample path of the MCMC output. Secondly, we draw samples from (3) by using the HY algorithm with the same initial values for the parameters. We assume $\varepsilon = 0.07$ and $\delta = 0.90$ to get a reasonable acceptance rate while ensuring proper mixing of the sampler output. The acceptance rate was 88% after 15,000 iterations and the chain was stabilized as can be seen from Figure 2 which shows the sample path of the MCMC output from HY algorithm. Comparing Figure 1 and 2 we observe that the mixing of the sampler output from M-H algorithm is good except for the intercept and the last parameter, whereas mixing of the sampler output from HY algorithm is good too for

all of the parameters.

We throw away 500 iterations as 'burn-in' samples and compute the posterior means and standard deviations from the remaining samples. These are given in Table 2.

Table 2: Summary results from the posterior distribution of the ordinary logistic regression model by using both random walk M-H and HY algorithms.

	Random Walk M-H algorithm		HY algorithm	
	Posterior		Posterior	
Variable	Mean	Std Dev.	Mean	Std Dev.
Intercept	-4.1868	0.7045	-4.2507	0.7439
AGE	0.0339	0.2463	0.0414	0.2479
RACE	0.5586	0.4387	0.5867	0.4357
DPROS	1.3055	0.3178	1.2902	0.3082
DCAPS	1.0737	0.4370	1.0931	0.4293
PSA	1.5253	0.2892	1.5287	0.2873
VOL	-0.0933	0.2493	-0.0911	0.2578
GLEASON	2.0062	0.4424	2.0517	0.4934

Scrutinizing Table 2, we see that the posterior means and standard deviations are almost the same obtained from both algorithms. But after observing the sample path of the MCMC outputs it can be concluded that most of the times hybrid algorithm produces satisfactory results that uses the derivative evaluations of the target posterior distribution.

We now draw samples from Equation (5) by using both M-H and HY algorithms. The initial values used are from the ordinary logistic regression fit, but we fix $\beta_j > 0$, for $j = 1, 2, \cdots, p$ for this model. For the M-H algorithm the jump sizes used are 0.20, 0.35, 0.20, 0.20, 0.40, 0.20, 0.35, 0.15, and 0.06, respectively for nine parameters. The acceptance rates were around 60% for the eight parameters and 50% for λ after iterating the algorithm 15,000 times. For HY algorithm we assume $\varepsilon = 0.064$, $\delta = 0.90$. The acceptance rate is 85% after 15,000 iterations. The sample plots are given in Figures 3 and 4, respectively for M-H and HY algorithms. From Figure 3 we see that mixing of the sampler output for most of the parameters is poor and the chains might not converge after many many iterations. On the other hand, although the mixing of the sampler output in the plots of Figure 4 is slow, the chains will be stationary after a long number of iterations.

In computing posterior means and standard deviations we throw away 2000 iterations as 'burn-in' from the sample MCMC output of the random walk M-H, but we use 15,000 samples from the sample MCMC output of the HY algorithm to compute posterior means and standard deviations. These are given in Table 3.

Table 3: Summary results from the posterior distribution of the new logistic regression model using both RW M-H and HY algorithm.

	Random Walk M-H algorithm		HY algorithm	
	Posterior		Posterior	
Variable	Mean	Std Dev.	Mean	Std dev.
Intercept	-14.7076	4.0054	-6.6912	1.7979
AGE	4.1086	3.1278	0.7550	0.7699
RACE	9.3113	4.0591	2.1640	1.3934
DPROS	10.8454	3.7660	3.0940	1.3635
DCAPS	9.9646	4.0384	2.8229	1.4379
PSA	11.6413	3.8572	3.5141	1.4509
VOL	4.2971	3.2191	0.6827	0.7246
GLEASON	11.7252	3.7871	3.9308	1.5205
λ	1.3365	0.2854	1.5797	0.3799

Observing Table 3 we see that the posterior means and standard deviations obtained from the HY algorithm are much smaller and more stable than those obtained from the M-H algorithm. Therefore, we may conclude that the new model is challenging for the M-H algorithm and might not provide satisfactory results. On the other hand, use of derivative evaluations in the HY algorithm provides better output.

7 Example 2: Multivariate Normal Distribution

Let our target distribution Π for X is now the k-variate normal distribution with mean vector zero and covariance matrix Σ. Instead of assuming uniform correlation between each pair of elements of X, we will consider that the pair of elements of X has exponential correlation. In contrast to the uniform correlation model, the correlation between a pair of measurements on the same subject decays towards zero as the time separation between the measurements increases. Let us denote the correlation by ρ. In this scenario Σ has the form:

$$\begin{pmatrix} 1 & \rho & \rho^2 & \rho^3 & \dots \\ \rho & 1 & \rho & \rho^2 & \dots \\ \rho^2 & \rho & 1 & \rho & \dots \\ \rho^3 & \rho^2 & \rho & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \dots \end{pmatrix}_{k\times k}$$

In general, multivariate distributions with high correlations tend to be challenging for MCMC methods. Let the likelihood function of the k-variate normal distribution is

proportional to $\exp(-\frac{1}{2}\theta'\Sigma^{-1}\theta)$, where θ is $k\times 1$ vector of parameters and Σ is the $k\times k$ covariance matrix of θ. Our goal is to sample θ from the k-variate normal distribution. Therefore the log likelihood function is:

$$\log L = K - \frac{1}{2}\theta'\Sigma^{-1}\theta, \tag{6}$$

where K is an unknown constant. Differentiating (6) with respect to θ we have,

$$\frac{\partial \log L}{\partial \theta} = -\Sigma^{-1}\theta.$$

We will simulate samples from (6) by using both the M-H and HY algorithms assuming moderate and high correlations, e.g., $\rho = 0.70$, $\rho = 0.90$, etc. For simplicity, the number of parameters is taken to be eight, i.e. $k = 8$. To draw samples from (6) using the M-H (component-wise updating) algorithm with $\rho = 0.70$ the jump sizes are set to 1.0, 0.80, 0.90, 0.90, 0.90, 0.90, 0.90, and 1.2, respectively for eight parameters. We iterate the algorithm 5,000 times with the initial values -3.5 for all eight parameters. The acceptance rates were around 60%. The sample plots are given in Figure 5.

For HY algorithm we assume $\varepsilon = 0.40$. The acceptance rate is 90% after 5,000 iterations with the same initial values as above. The sample plots from the MCMC output are given in Figure 6. Comparing Figure 5 and 6 we see that mixing is better in plots from HY algorithm.

Next we draw samples from (6) assuming very high correlation, that is, $\rho = 0.90$ with the same initial values as before and iterate for 5,000 times. The jump sizes for M-H algorithm are 0.70, 0.60, 0.60, 0.60, 0.60, 0.60, 0.60, and 0.70, respectively for eight parameters. The acceptance rates are between 50% and 60%. For the HY algorithm the step size is 0.20 and the acceptance rate is 92% after 5,000 iterations. The sample plots from the MCMC outputs are given in Figures 7 and 8, respectively for M-H and HY algorithms. For $\rho = 0.90$, mixing of the sampler output from M-H algorithm is very slow as can be seen from Figure 7. But from Figure 8 we see that the mixing of the sampler output from HY algorithm is much better. All of the chains are converging to zero. Thus our intuition is that HY algorithm performs better than random walk M-H as the correlation between pair of elements getting higher and higher.

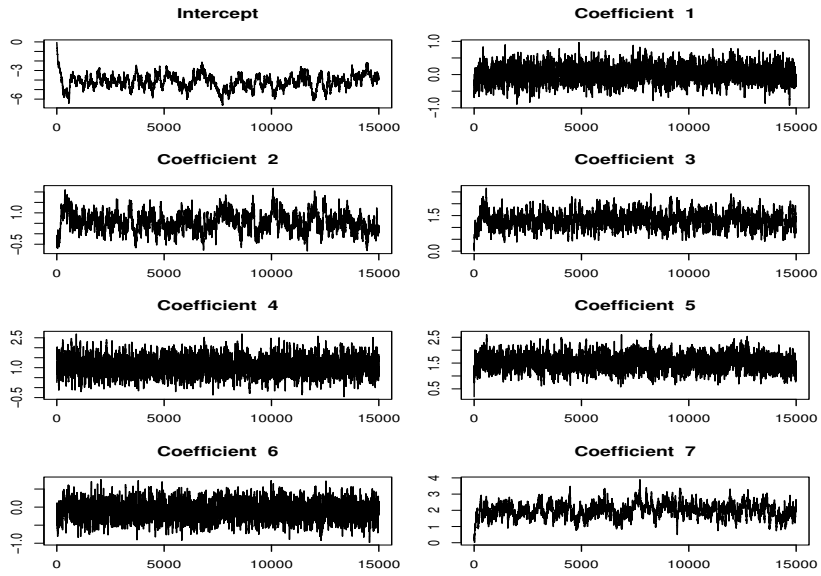

Figure 1: Sample path of the MCMC output from Equation (3) by using the random walk M-H algorithm.

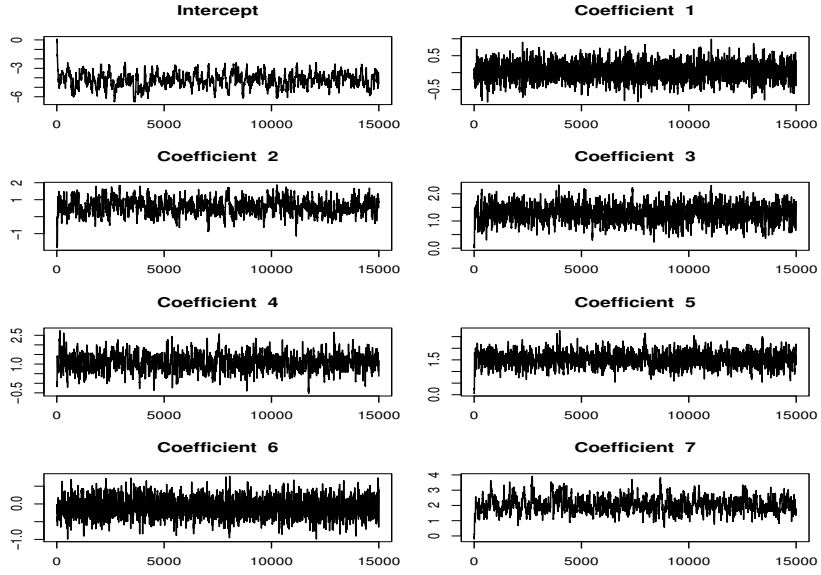

Figure 2: Sample path of the MCMC output from Equation (3) by using the hybrid (HY) algorithm.

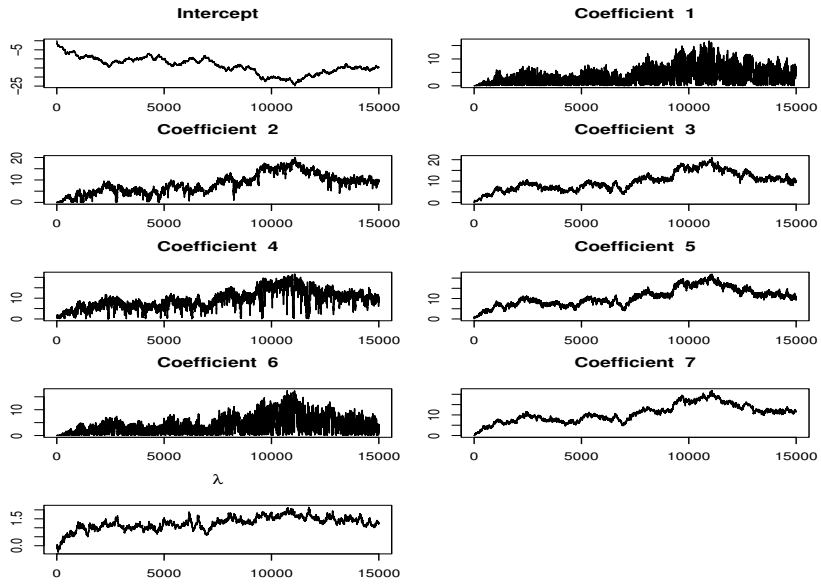

Figure 3: Sample path of the MCMC output from Equation (5) by using the random walk M-H algorithm.

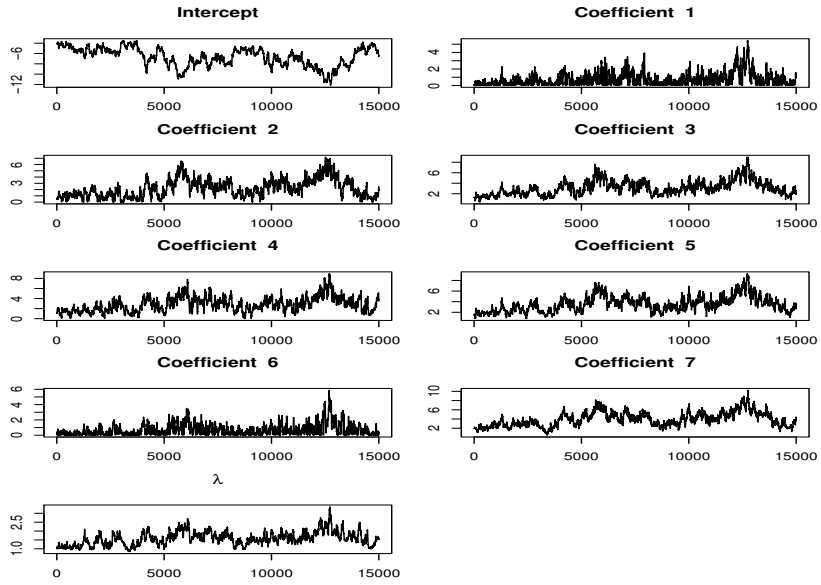

Figure 4: Sample path of the MCMC output from Equation (5) by using the hybrid (HY) algorithm.

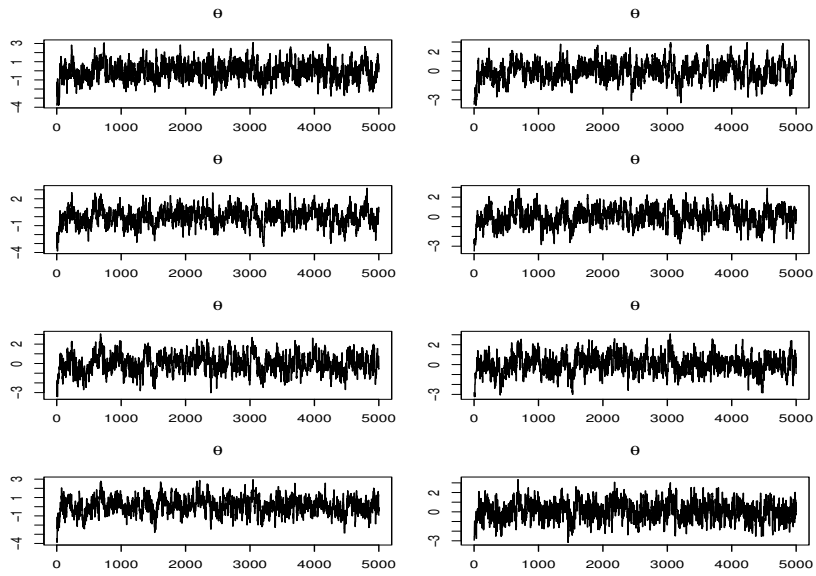

Figure 5: Sample path of the MCMC output from Equation (6) by using the M-H algorithm with $\rho = 0.70$. The eight panels correspond to eight parameters of θ.

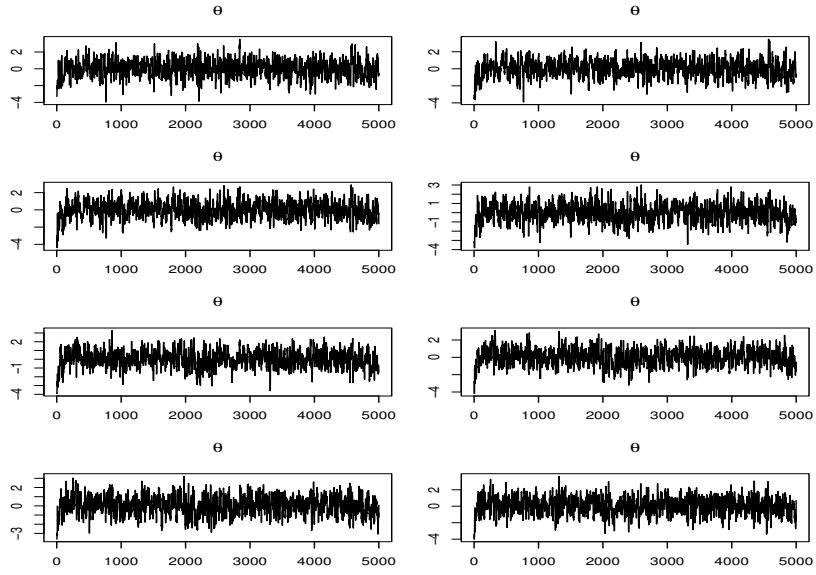

Figure 6: Sample path of the MCMC output from Equation (6) by using the HY algorithm with $\rho = 0.70$. The eight panels correspond to eight parameters of θ.

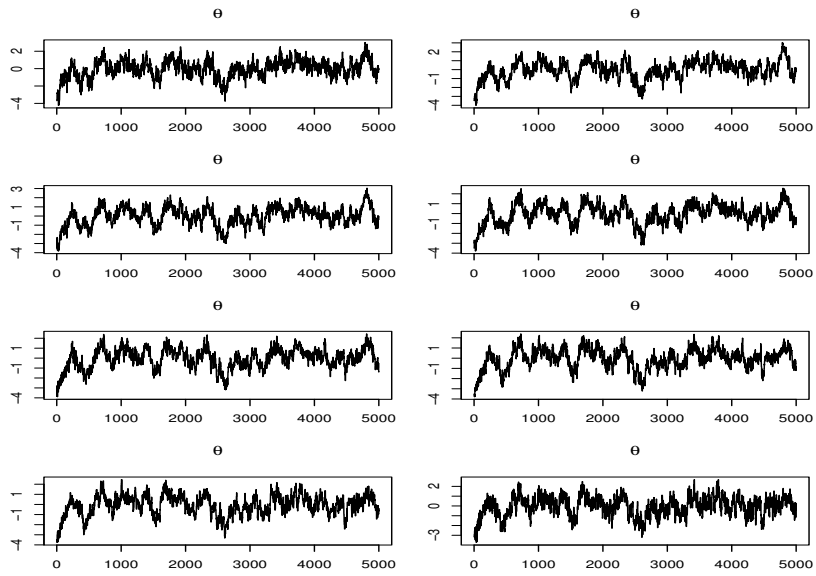

Figure 7: Sample path of the MCMC output from Equation (6) by using the M-H algorithm with $\rho = 0.90$. The eight panels correspond to eight parameters of θ.

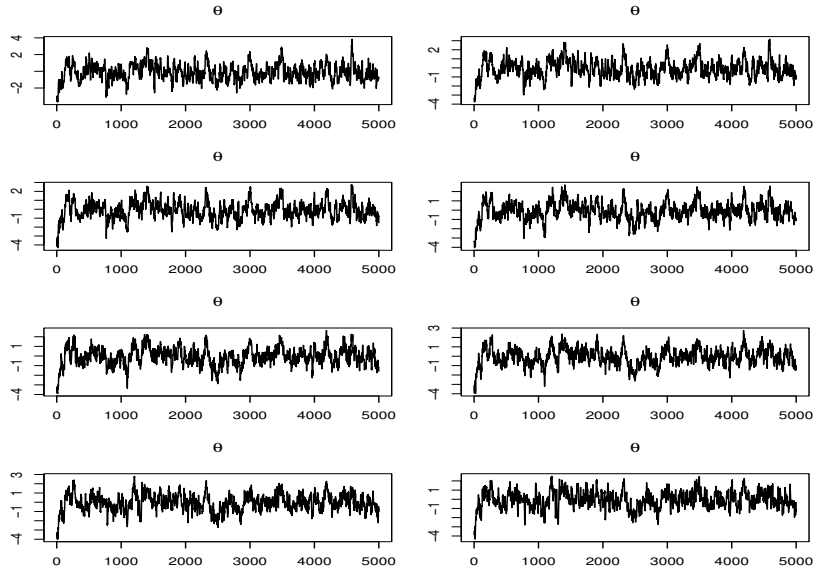

Figure 8: Sample path of the MCMC output from Equation (6) by using the HY algorithm with $\rho = 0.90$. The eight panels correspond to eight parameters of θ.

8 Monte Carlo Error Variance

The two algorithms can be compared by obtaining an estimate of the Monte Carlo error variance from the sample MCMC outputs. To get an estimate of this we run each of the algorithms 5 times with the initial values -3.5, 3.5, -5, 5, and 6, respectively. We throw away 500 iterations as 'burn-in'. From each run we get Monte Carlo estimates of the eight parameters which are actually trying to estimate mean of θ. Let $\psi = E(\theta)$ be the true mean of θ which we assume to be zero and $\hat{\psi}_i$'s are the Monte Carlo estimates of ψ. An estimate of the Monte Carlo error variance can be computed by the quantity:

$$\frac{1}{p}\sum_{j=1}^{p}\left(\hat{\psi}_j - \psi\right)^2, \tag{7}$$

where $p = 5$. The Monte Carlo error variance estimates for each of the eight parameters obtained by using Equation (7) are given in Table 4.

Table 4: Monte Carlo error variance estimates obtained by (7) for $\rho = 0.70$ and $\rho = 0.90$ by using both algorithms.

	$\rho = 0.70$		$\rho = 0.90$	
Parameter	M-H	HY	M-H	HY
θ_1	0.0025	0.0002	0.0071	0.0060
θ_2	0.0021	0.0004	0.0070	0.0062
θ_3	0.0018	0.0008	0.0075	0.0066
θ_4	0.0013	0.0009	0.0064	0.0055
θ_5	0.0012	0.0008	0.0062	0.0030
θ_6	0.0022	0.0004	0.0054	0.0019
θ_7	0.0028	0.0007	0.0055	0.0015
θ_8	0.0020	0.0007	0.0029	0.0011

From Table 4 we see that when $\rho = 0.70$ estimates of error variances are much smaller for HY algorithm than those from the random walk M-H algorithm. When $\rho = 0.90$, though the error variances increase, still the estimates from HY algorithm are much smaller than those obtained from the M-H algorithm.

9 Discussion

The hybrid (HY) algorithm appears to provide consistent results over the random walk Metropolis Hastings algorithm in respect of convergence and efficiency. Assuming two different types of target distributions we compare the two algorithms under consideration. The first type is posterior distributions from two different logistic regression

models and the second type is the multivariate normal distribution with exponential correlation between the pair of elements. In each of the above examples we observed that HY algorithm is more likely to be efficient than the M-H algorithm. The significant issue to notice is that in implementing the random walk M-H algorithm for component-wise updating, setting up of correct proposal variance or jump sizes for the parameters is a difficult task, especially when the dimension of the target distribution is very high. On the other hand, implementation of HY algorithm is not very difficult and we need to set only one proposal variance for all of the parameters. Though choice of proposal variance affects both algorithms, hybrid algorithm, in fact, is less sensitive to this specification. In applications, we require extra human effort to understand and program the HY algorithm and extra machine time required is negligible.

In the end, it may be concluded, at least from the examples analyzed and discussed in this paper, that for practical purposes use of the hybrid algorithm for complicated and high-dimensional target distributions rather than the random walk Metropolis Hastings algorithm will be a good choice for satisfactory and improved results.

Appendix

Derivatives of Model (4)

Rewriting Equation (5) we have,

$$\log P(\beta,\phi|y) = K + \sum_{i=1}^{n}\Big[y_i g_i(\beta,\phi) - \log(1+e^{g_i(\beta,\phi)})\Big] - \frac{1}{2c^2}\phi^2, \tag{8}$$

where

$$g_i(\beta,\phi) = \beta_0 + \left(\sum_{j=1}^{p}\beta_j^{e^{\phi}}X_{ij}\right)^{e^{-\phi}}. \tag{9}$$

Differentiating Equation (8) with respect to β_j, $j = 0,1,2,\cdots,p$ and ϕ we get the following $(p+2)$ equations:

$$\frac{\delta \log P(\beta,\phi|y)}{\delta\beta_j} = \sum_{i=1}^{n}\left[y_i - \left(1-\frac{1}{1+e^{g_i(\beta,\phi)}}\right)\right]\frac{\delta}{\delta\beta_j}g_i(\beta,\phi), \tag{10}$$

$$\frac{\delta \log P(\beta,\phi|y)}{\delta\phi} = \sum_{i=1}^{n}\left[y_i - \left(1-\frac{1}{1+e^{g_i(\beta,\phi)}}\right)\right]\frac{\delta}{\delta\phi}g_i(\beta,\phi) - \frac{\phi}{c^2}. \tag{11}$$

To get the $(p+1)$ equations of (10) we need to differentiate $\frac{\delta}{\delta\beta_j}g_i(\beta,\phi)$ with respect to β_j, $j = 0,1,2,\cdots,p$. The derivatives are as follows:

$$\begin{aligned}
\frac{\delta g_i(\beta,\phi)}{\delta\beta_0} &= 1, \\
\frac{\delta g_i(\beta,\phi)}{\delta\beta_1} &= \left(\sum_{j=1}^{p}\beta_j^{e^\phi}X_{ij}\right)^{e^{-\phi}-1}\beta_1^{e^\phi-1}X_{i1}, \\
\frac{\delta g_i(\beta,\phi)}{\delta\beta_2} &= \left(\sum_{j=1}^{p}\beta_j^{e^\phi}X_{ij}\right)^{e^{-\phi}-1}\beta_2^{e^\phi-1}X_{i2}, \\
&\vdots \\
\frac{\delta g_i(\beta,\phi)}{\delta\beta_p} &= \left(\sum_{j=1}^{p}\beta_j^{e^\phi}X_{ij}\right)^{e^{-\phi}-1}\beta_p^{e^\phi-1}X_{ip}.
\end{aligned}$$

Furthermore,

$$\frac{\delta g_i(\beta,\phi)}{\delta\phi} = \left(\sum_{j=1}^{p}\beta_j^{e^\phi}X_{ij}\right)^{e^{-\phi}}\left[\frac{\sum_{j=1}^{p}\left(\beta_j^{e^\phi}\log(\beta_j)X_{ij}\right)}{\left(\sum_{j=1}^{p}\beta_j^{e^\phi}X_{ij}\right)} - e^{-\phi}\log\left(\sum_{j=1}^{p}\beta_j^{e^\phi}X_{ij}\right)\right].$$

References

[1] Chib, Siddartha and Greenberg, Edward (1995). Understanding the Metropolis-Hastings algorithm. *The American Statistician*, Vol. 49, No.4, 327-335.

[2] Duane, S., Kennedy, A.D., Pendleton, B.J. and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195, 216-222.

[3] Gelman, A., Carlin, John B., Stern, Hal S. and Rubin, Donald B. (1995). *Bayesian Data Analysis*. Chapman & Hall, London.

[4] Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 6, 721-741.

[5] Gustafson, Paul (1998). A guided walk Metropolis algorithm. *Statistics and computing*, 8, 357-364.

[6] Gustafson, P., MacNab, Y.C., and Wen, S. (2002). On the value of derivative evaluations and random walk suppression in Markov chain Monte Carlo algorithms. To appear.

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

[8] Hosmer, D.W. and Lemeshow, S. (2000). *Applied Logistic Regression*. John Wiley & Sons, Inc.

[9] Kazi, Azad M.R. (2003). Non-additive effects in Logistic Regression. Unpublished M.Sc. thesis at UBC.

[10] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics* 21, 1087-1092.

[11] Neal, R.M. (1993). Bayesian learning via stochastic dynamics, in C.L. Giles, S.J. Hanson, and J.D. Cowan (eds.). *Advances in Neural Information Processing Systems* 5, 475-482, San Mateo, California. Morgan Kaufmann.

BLANK PAGE