

A note on some algorithms for the Gibbs posterior¹

Kun Chen^a, Wenxin Jiang^{b2}, Martin A. Tanner^b

^a*Department of Statistics and Actuarial Science, University of Iowa, Iowa City, IA 52242, USA*

^b*Department of Statistics, Northwestern University, Evanston, IL 60208, USA*

MCS: 62-04

July 7, 2009

Abstract

Jiang and Tanner (2008) consider a method of classification using the Gibbs posterior which is directly constructed from the empirical classification errors. They propose an algorithm to sample from the Gibbs posterior which utilizes a smoothed approximation of the empirical classification error, via a Gibbs sampler with augmented latent variables. In this paper, we note some drawbacks of this algorithm and propose an alternative method for sampling from the Gibbs posterior, based on the Metropolis algorithm. Numerical performance of the algorithms are examined and compared via simulated data. We find that the Metropolis algorithm produces good classification results at an improved speed of computation.

1 Introduction

1.1 Gibbs posterior

The problem of interest here is to predict y , a $\{0, 1\}$ response, based on x , a vector of predictors of dimension $\dim(x) = K$. We have $D^n = (y^{(i)}, x^{(i)})_1^n$, the observed data with *sample size* n , typically assumed to form n iid (independent and identically distributed) copies of (y, x) . We are interested in predicting y by a linear classification rule $I[x^T \beta > 0]$ indexed by a parameter b , and try to minimize the classification risk $R(\beta) = P^*\{y \neq I(x^T \beta > 0)\}$. For this purpose, one can use a *Gibbs posterior* (see, e.g., Zhang 2006, Jiang and Tanner 2008) over $\beta \in \Omega$ for some parameter space $\Omega \subset \Re^K$:

¹Technical Report 09-01, Department of Statistics, Northwestern University. The Octave programs for the newly proposed Metropolis algorithm are downloadable at

<http://newton.stats.northwestern.edu/~jiang/gibbscompute/metro5.zip>

An R package is also available upon request.

²Corresponding author.

$$\omega(d\beta|D^n) = w(d\beta|D^n)\pi(d\beta) = e^{-n\psi R_n(\beta)}\pi(d\beta) / \int_{\beta \in \Omega} e^{-n\psi R_n(\beta)}\pi(d\beta),$$

where π is a prior over $\beta \in \Omega$, and $\psi > 0$ is a constant (‘the inverse temperature’), which will be taken to be 1 unless otherwise noted.

Here R_n is a sample version of R depending on data D^n . Examples include:

- (i) $R_n = n^{-1} \sum_{i=1}^n I[y^{(i)} \neq A_i] = -\psi^{-1} n^{-1} \sum_{i=1}^n \log\{A_i e^{\psi(y^{(i)}-1)} + (1 - A_i) e^{-\psi y^{(i)}}\}$, where $A_i = I[(x^{(i)})^T \beta > 0]$;
- (ii) $R_n = -\psi^{-1} n^{-1} \sum_{i=1}^n \log\{\Phi_i e^{\psi(y^{(i)}-1)} + (1 - \Phi_i) e^{-\psi y^{(i)}}\}$, where $\Phi_i = \Phi(\sigma^{-1}(x^{(i)})^T \beta)$, Φ is the standard normal cumulative density function, and σ is a scaling factor.

Choices (ii) and (i) are close when $\sigma \rightarrow 0$ but Choice (ii) makes R_n smooth in β . Jiang and Tanner (2008, Section 7) proposed the use of (ii) since R_n in (ii) is related to a mixture model and can be used to simulate the posterior by the Gibbs sampler with augmented data, so that all conditional distributions are standard.

1.2 Variable selection and parameterization

Jiang and Tanner (2008, Section 4.2) propose a parameterization of β so as to allow a subset of x components to be used for classification with variable selection. For this purpose, the parameter vector $\beta = (\beta_j)_1^K$ is replaced by the following parameters: $(\gamma, \beta_1, \tilde{\beta}_\gamma)$, where β_1 is the first component of β and can be standardized as $\beta_1 \in \{-1, +1\}$, following Horowitz (1992). The integer vector $\gamma = (\gamma_j)_1^K$ is the ‘model’ indicator with $\gamma_j = I[\beta_j \neq 0]$, noting which components of β are nonzero (therefore $\gamma_1 = 1$). The vector $\tilde{\beta}_\gamma$ includes real valued components $(\beta_j)_{j>1, \gamma_j=1}$. The Gibbs posterior is induced by a prior π on $\beta \in \Omega$, which could be equivalently specified by placing a prior on the parameters $(\gamma, \beta_1, \tilde{\beta}_\gamma)$. A Gibbs posterior is then obtained as $\omega(d\beta|D^n) \propto e^{-n\psi R_n(\beta)}\pi(d\beta)$. In the following section we will consider a normal-binary prior for $(\gamma, \beta_1, \tilde{\beta}_\gamma)$.

1.3 A prior specification (normal-binary)

For the prior π , suppose conditional on γ , β_1 is independent of $\tilde{\beta}_\gamma$ (the subset of $(\beta_j)_2^K$ with $\gamma_j = 1$), $\beta_1|\gamma = \pm 1$ with probability 0.5 each, and $\tilde{\beta}_\gamma|\gamma \sim N(0, V_\gamma)$, according to the prior π .

The following remarks specify the prior on γ (the ‘model’ indicators). We impose $x_2 = 1$ to correspond to the intercept β_2 , which is always included in the model, so that $\gamma_2 = 1$ always. Also $\gamma_1 = 1$, which corresponds to always including the signed coefficient $\beta_1 \in \{-1, 1\}$ due to the standardization of Horowitz (1992). For $(\gamma_j)_{j=3}^K$, we assume that they are iid binary with selection probability λ and size restriction \bar{r} . Conceptually, one first generates $\check{\gamma} = \check{\gamma}_1^K$, where $\check{\gamma}_{1,2} = 1$, and $\check{\gamma}_3^K$ are iid binary with selection probability λ . Then set $\gamma = \check{\gamma}$ only when $\sum_{j=1}^K \check{\gamma}_j \leq \bar{r}$.

1.4 The Gibbs sampler algorithm of Jiang and Tanner (2008, Section 7)

Consider the smoothed sample risk function R_n in (ii). It is noted that

$$e^{-n\psi R_n} \propto \prod_{i=1}^n \{ \Phi_i p_1^{y^{(i)}} (1 - p_1)^{1-y^{(i)}} + (1 - \Phi_i) p_0^{y^{(i)}} (1 - p_0)^{1-y^{(i)}} \},$$

where $\Phi_i = \Phi(\sigma^{-1}(x^{(i)})^T \beta)$, Φ is the standard normal cumulative density function, and σ is a scaling factor, $p_0 = 1/(1 + e^\psi)$ and $p_1 = e^\psi/(1 + e^\psi)$.

This can be recognized as the likelihood for a mixture of two binary models with mixing probability Φ_i , suggesting a data augmentation method incorporating latent variables $Z = (Z^{(i)})_1^n$, where $Z^{(i)}$ are independent $N((x^{(i)})^T \beta, \sigma)$, so that $y^{(i)}|Z^{(i)}$ are independent $Bin(1, p_{I[Z^{(i)} > 0]})$. The Gibbs sampler can be used to obtain the joint distribution of (Z, γ, β) , where all full conditional distributions are standard. We integrate over β_γ and use the distribution $\gamma|Z$ instead of $\gamma|Z, \beta_\gamma$ in the Gibbs sampler.

Define $\beta^T = (\beta_1, \tilde{\beta}^T)$, $\tilde{\gamma} = (\beta_1, \gamma_3, \dots, \gamma_K)$, and let $\tilde{\beta}_\gamma$ include $\tilde{\beta}_j$'s ($j = 2, 3, \dots, K$) with $\gamma_j = 1$. (Note that $\gamma_1 = \gamma_2 = 1$.) Consider the following MCMC algorithm starting from any initial position. For $t = 1, 2, \dots$:

(Step 1). Sample $Z^t | \tilde{\beta}_\gamma^{t-1}, \tilde{\gamma}^{t-1}$;

(Step 2). Sample $\tilde{\gamma}^t | Z^t$;

(Step 3). Sample $\tilde{\beta}_\gamma^t | \tilde{\gamma}^t, Z^t$.

The details of these three steps are described in Jiang and Tanner (2008, Section 7). They note that all the conditional distributions are standard.

1.5 Drawbacks of the Jiang and Tanner (2008, Section 7) algorithm.

Despite the attractiveness of the above algorithm regarding the standard conditional distributions, in several simulation experiments, we find (see Section 3) that this algorithm may not do well in minimizing the classification risk. One undesirable feature is that Step 2 involves updating the model indexes γ_3^K component by component. This is very inefficient when K is large and can lead to slow convergence.

A more fundamental weakness is revealed when we closely examine Step 3 of the Jiang and Tanner (2008, Section 7) algorithm, which, when omitting the time index t , is as follows:

Step 3: Simulate $\tilde{\beta}_\gamma | \beta_1, \gamma, Z \sim N \left\{ (\sigma^2 V_\gamma^{-1} + \tilde{X}_\gamma^T \tilde{X}_\gamma)^{-1} \tilde{X}_\gamma^T Z(\beta_1), \sigma^2 (\sigma^2 V_\gamma^{-1} + \tilde{X}_\gamma^T \tilde{X}_\gamma)^{-1} \right\}$.

Here \tilde{X}_γ represents the design matrix of $[x_j^{(i)}]$ with $i \in \{1, \dots, n\}$ and j includes all indexes $j > 1$ such that $\gamma_j = 1$, $Z(\beta_1) = (Z^{(1)}(\beta_1), \dots, Z^{(n)}(\beta_1))^T$ where $Z^{(i)}(\beta_1) = Z^{(i)} - x_1^{(i)} \beta_1$. Although this is a standard distribution, we notice that the standard deviation is of order $O(\sigma)$ for small smoothing parameter σ . Therefore the update size in $\tilde{\beta}_\gamma$ will be small in this case, resulting in slow convergence. On the other hand, the smoothing parameter σ is required to be small to have good classification performance, since the smoothing risk (ii) will be close to the unsmoothed classification error (i). Therefore, there may be situations when no choice of the smoothing parameter can simultaneously satisfy the goals of the classification accuracy and speedy computation. In particular, this algorithm will not be able to handle the Gibbs posterior using the unsmoothed risk (i) (corresponding to $\sigma = 0$).

2 Metropolis Algorithm

In order to simulate from the Gibbs posterior with any amount of smoothing in the risk (including the unsmoothed case (i)), we propose a Metropolis algorithm which will not be slowed by a small σ or even $\sigma = 0$. This algorithm is also more efficient than the component-wise updates of the model indexes for large K , since here the computation of each iteration will no longer scale with K .

For variable selection purposes, the Metropolis algorithm can have a ‘between’ step to propose deletion / addition, or swapping of variables as described in Brown, Vannucci and

Fearn (2002) in the classical Bayesian context. We modify the algorithm of Brown, Vannucci and Fearn (2002) for the Gibbs posterior and add a ‘within’ step to update the parameter values when the model indicators are fixed. The reason is that the nonzero “regression coefficient” parameters $\tilde{\beta}_\gamma$ cannot be integrated away analytically in general.

In the Metropolis algorithm below for each $j = 3, \dots, K$, denote $q(\beta_j)$ as the density of $N(0, \sigma_{add}^2)$, which is used as the proposal density for a move to a possibly nonzero new location β_j from the previous location 0. Let $\pi(\beta|D^n)$ represent the density of the target Gibbs posterior. To allow moves among all possible models and parameters, each iteration of the algorithm is a composite transition ($\beta \rightarrow \beta' \rightarrow \beta^*$) that combines BETWEEN steps that propose changes between different models, as well as WITHIN steps that propose changes of parameters within a fixed model. These steps are then given as:

BETWEEN steps I and II: These update β to β' with model indexes changing.

I: (add/delete): Randomly choose an index $j \in \{3, \dots, K\}$. (Note that $\gamma_1 = \gamma_2 = 1$.)

I(i) (delete) If $\gamma_j = 1$, propose $\gamma'_j = 0$ and propose a move from β_j to $\beta'_j = 0$, with all remaining components of β unchanged. This proposal is accepted with probability

$$\min \left\{ 1, \frac{\pi(\beta'|D^n)q(\beta_j)}{\pi(\beta|D^n)} \right\}.$$

I(ii) (add) If $\gamma_j = 0$, propose $\gamma'_j = 1$ and propose a move from $\beta_j = 0$ to $\beta'_j \sim N(0, \sigma_{add}^2)$, with all remaining components of β unchanged. This proposal is accepted with probability

$$\min \left\{ 1, \frac{\pi(\beta'|D^n)}{\pi(\beta|D^n)q(\beta'_j)} \right\}.$$

II: (swap): Let $I_0 = \{k : \gamma_k = 0, k \in \{3, \dots, K\}\}$ and $I_1 = \{k : \gamma_k = 1, k \in \{3, \dots, K\}\}$.

When I_0 and I_1 are both non-empty, randomly choose an index k from I_0 and randomly choose an index l from I_1 .

Propose $\gamma'_k = 1$ and $\gamma'_l = 0$, and propose a move from $\beta_k = 0$ to $\beta'_k \sim N(0, \sigma_{add}^2)$, as well as a move from β_l to $\beta'_l = 0$. This proposal is accepted with probability

$$\min \left\{ 1, \frac{\pi(\beta'|D^n)q(\beta_l)}{\pi(\beta|D^n)q(\beta'_k)} \right\}.$$

WITHIN step III: These update β' to β^* with model indexes fixed and with the nonzero values of β' 's changed.

III: (within): Propose a move from β'_1 to $\beta_1^* \sim \text{Bin}(1, 0.5)$, as well as a move from $\tilde{\beta}'_\gamma$ to $\tilde{\beta}_\gamma^* \sim N(\tilde{\beta}'_\gamma, \sigma_q^2 I_{(\sum_{j=2}^K \gamma_j) \times (\sum_{j=2}^K \gamma_j)})$, accepted with probability

$$\min \left\{ 1, \frac{\pi(\beta^* | D^n)}{\pi(\beta' | D^n)} \right\}.$$

The steps I, II, III are then cycled in the iterations. Alternatively, I or II can be randomly chosen with probability 0.5 each, and followed by III in each iteration. This second option is used unless otherwise noted.

In Section 3, we also consider the situation when β_1 is not standardized to be ± 1 and when γ_1 is not constrained to be 1. The algorithm can be obtained from above by changing the index set from $\{3, \dots, K\}$ to $\{1, 3, \dots, K\}$ (noting that the index 2 corresponds to the ‘intercept’ term and is always included) and by incorporating β_1 to the continuous components $\tilde{\beta}_\gamma$ when they are being updated.

3 Simulations

3.1 Models for simulation

For each simulation, a “training set” includes $(x^{(i)}, y^{(i)})_{i=1}^n$, which are iid realizations of (x, y) . We will use $\dim(x) = K = 50$, $n = 30$. An iid realization of “testing set” is also generated with size $n_{tst} = 200$ for each simulation. We consider the following two models for the Data Generation Process (DGP).

1. *Simulation model 1 of DGP: “mvm”.*

$P[y = 0] = 0.5 = P[y = 1]$. $P[x_2 = 1 | y = 0] = 1 = P[x_2 = 1 | y = 1]$ so that x_2 corresponds to the intercept term. For $j \neq 2$,

$$(x_j)_{j \neq 2} | y = 0 \sim N((\mu_j^-)_{j \neq 2}, s^2 I_{K-1}),$$

$$(x_j)_{j \neq 2} | y = 1 \sim N((\mu_j^+)_{j \neq 2}, s^2 I_{K-1}),$$

where $\mu_j^\pm = 0$ for $j \in \{4, 5, \dots, K\}$ and $\mu_j^\pm = \pm \delta$ for $j \in \{1, 3\}$. We choose $\delta = 0.5$ and $s^2 = 0.09$.

2. *Simulation model 2 of DGP: “fivedot”.*

For each $j \in \{4, 5, \dots, K\}$, x_j are Uniform $[-1, 1]$. $P[x_2 = 1] = 1$ so that x_2 corresponds to the intercept term. For $j \in \{1, 3\}$, $P[(x_1, x_3) = (0, 0)] = 0.75$, $P[(x_1, x_3) = (s_1, s_2)] = 1/16$ for any $s_1 \in \{-1, 1\}$ and $s_2 \in \{-1, 1\}$. $y = I[(x_1, x_3) \neq (0, 0)]$. We name this model “fivedot” since the design points of (x_1, x_3) form a set of five points.

3.2 Algorithms to be compared

We consider the following algorithms:

- (A). “Lee.et.al.”: This is a Gibbs sampler algorithm of Lee et. al. (2003), for sampling from the likelihood-based posterior under the probit regression model. Although this is not sampling from a Gibbs posterior that is directly constructed from the classification error, we still include it for comparison purposes, since this method is commonly used for classification.
- (B). “Sec7”: This is the Gibbs sampler algorithm of Jiang and Tanner (2008, Section 7), for sampling from the Gibbs posterior based on a smoothed sample classification error. (The smoothing parameter is σ which is the standard deviation of an augmented normal variable. “Sec7.2” uses $\sigma = 0.2$ and “Sec7.02” uses $\sigma = 0.02$). In the simulations with the Sec7 algorithms, we cycle through Steps 2, 3, 1 and use a random ordering when updating the model indexes in Step 2.
- (C). “Metropolis”: This is the Metropolis algorithm to sample from a Gibbs posterior constructed from the original sample classification error without smoothing ($\sigma = 0$). (We use $\sigma_{add} = 1$ and $\sigma_q = 1$ for the proposal densities in the BETWEEN and WITHIN steps, respectively.)

3.3 Measures of classification performance

Jiang and Tanner (2008) studied the mean classification performance when classification rules (parameterized by β) are obtained randomly from the Gibbs posterior. Given one training data set $D = (x^{(i)}, y^{(i)})_{i=1}^n$, when a section of the Markov chain $\{\beta^t(D)\}_{tmin}^{tmax}$ (after a burn-in period) is simulated from a posterior distribution, the mean performance is approximated by the average over these iterations. For any algorithm described above, based on one training

data set D , the training error averaged over iterations $tmin + 1$ to $tmax$ is then defined as:

$$AV.TR.ERR(D, D)_{tmin, tmax} = (tmax - tmin)^{-1} \sum_{t=tmin+1}^{tmax} n^{-1} \sum_{i=1}^n |y^{(i)} - I(x^{(i)T} \beta^t(D) > 0)|.$$

The corresponding average testing error is obtained by the performance on an independent data set $D^* = (x^{*(i)}, y^{*(i)})_{i=1}^{n_{tst}}$,

$$AV.TST.ERR(D, D^*)_{tmin, tmax} = (tmax - tmin)^{-1} \sum_{t=tmin+1}^{tmax} n_{tst}^{-1} \sum_{i=1}^{n_{tst}} |y^{*(i)} - I(x^{*(i)T} \beta^t(D) > 0)|.$$

We use $tmin = 1500$ and $tmax = 2000$ unless otherwise noted.

When simulations are repeated $nrep$ times (we will use $nrep = 50$), for each simulation $s = 1, \dots, nrep$, a training set $D(s)$ and a testing set $D^*(s)$ will be generated independently. The average performance (in training and testing errors) over these $nrep$ simulations will then be, respectively,

$$\begin{aligned} AV.TR.ERR_{tmin, tmax}^{1, nrep} &= (nrep)^{-1} \sum_{s=1}^{nrep} AV.TR.ERR(D(s), D(s))_{tmin, tmax}, \text{ and} \\ AV.TST.ERR_{tmin, tmax}^{1, nrep} &= (nrep)^{-1} \sum_{s=1}^{nrep} AV.TST.ERR(D(s), D^*(s))_{tmin, tmax}. \end{aligned}$$

It is such criteria that are compared for the different algorithms in our study.

3.4 Results

1. (Computing time.) We used Octave (which is the same as MATLAB) on a linux PC machine (Pentium 4HT, 3.2 GHz, 512 MB RAM), and ran 2000 iterations for all the methods. It took per unit time about 7 min (for each of the 50 simulations) for the Sec7 methods, and per unit time about 5 min for the Lee.et.al. method. The Metropolis method takes per unit time about 2 min when steps I,II, III are cycled in the iterations. (The time will decrease when I or II is randomly chosen in each iteration.)
2. (When β_1 is normalized.)

We ran 50 simulations and averaging over the repeated simulations, the results are shown in Table 1. In Table 1, Sec7 performs very poorly for the mvn model. We found that it is related to the β_1 . When $\beta_1 = -1$, the classification result is poor for “mvn”, but when σ is small, it is very difficult for Sec7 algorithms to move β_1 from -1 to $+1$.

(Note that this would not affect the fivedot model which is symmetric in classification performance regarding whether $\beta_1 = 1$ or -1 is used.) By examining a detailed description of Step 2 in Jiang and Tanner (2008, Section 7), we found that the probability of such a move is exponentially small, roughly of order $e^{-(\beta_1^2/\sigma^2)*(positive\ const)}$. This may be resolved either by standardizing $|\beta_1|$ to be smaller than 1 (e.g., of order σ), or use unrestricted β_1 . The latter approach is appealing since it allows β_1 to be entered symmetrically as other coefficients and is more parallel to Lee.et.al.’s approach. Although the classification rule is invariant to a scale change of all β_j ’s, the coefficients will not be too large due to the use of normal priors.

Table 1: Average performance over iterations 1501 to 2000, averaged over 50 simulations. (β_1 normalized to $\{-1, +1\}$ in all methods except Lee.et.al.).

‘‘mvn’’	Lee.et.al.	Sec7.2	Sec7.02	Metropolis
training error	0.035364	0.250579	0.345784	0.044019
test error	0.066643	0.336689	0.386133	0.086691
‘‘fivedot’’:	Lee.et.al.	Sec7.2	Sec7.02	Metropolis
training error	0.21617	0.13141	0.16233	0.10168
test error	0.29310	0.15528	0.16708	0.15045

Note: In the Metropolis algorithm here, steps I, II, III are cycled in each iteration. Also, we applied a size restriction 3 to the Lee.et.al. method which does not normalize β_1 , and a size restriction 4 for all other methods which do impose normalization $\beta_1 \in \{-1, 1\}$.

3. (When β_1 is unrestricted.)

When we use the approach without normalizing β_1 , we obtain the results in Table 2. Comparing these results to those in Table 1, for the mvn model, we notice that the performance of Sec7 algorithms without standardizing β_1 improves dramatically. (However, the Lee.et.al. method still performs the best.) Comparing to Table 1 for the fivedot model, the performance without standardization is slightly worse. This is a case

where linear classification is not the optimal rule, where the Lee.et.al. method using the likelihood-based posterior performs the worst.

Table 2: Average performance over iterations 1501 to 2000, averaged over 50 simulations. (β_1 unrestricted)

‘‘mvn’’	Lee.et.al.	Sec7.2	Sec7.02	Metropolis
mean training error	0.040283	0.063457	0.068791	0.045165
mean test error	0.073354	0.107584	0.100987	0.097986
‘‘fivedot’’	Lee.et.al.	Sec7.2	Sec7.02	Metropolis
mean training error	0.20327	0.15933	0.21289	0.12384
mean test error	0.29908	0.20279	0.22966	0.18175

Note: In the Metropolis algorithm here, Steps I or II are randomly chosen with probability 0.5 each, and followed by III in each iteration. Also, we applied a size restriction 4 to the Lee.et.al. method, same as what is used for other methods.

4. (Choice of parameters.)

We used the choice $\psi = 1$ and $\sigma \in \{0.2, 0.02\}$. The other parameter choices are $V_\gamma = c(\tilde{X}_\gamma^T \tilde{X}_\gamma + 0.000001 * I)^{-1}$, $c = n$; $\lambda = 0.05$; size restriction is $\bar{r} = 4$; in these simulations. The term $0.000001 * I$ was added to avoid singular matrix inversion. The total number of iterations is 2000; first 1500 are run-in, where the performance of the last 500 outcomes are averaged. We always include the intercept term. In the Gibbs posterior-based methods (for Sec7 algorithm and the Metropolis) we applied a size restriction $\bar{r} = 4$. For the likelihood-based posterior (with Lee.et.al method) all β_j components enter symmetrically and we applied a size restriction (taken to be 3 in Table 1 and 4 in Table 2), which is a favorable modification for this method since here x_1 , x_2 and x_3 are relevant variables in data generation, while the original Lee et al. method does not have a size restriction.

5. (Slowness of Sec7 algorithm.) In Table 2, for data generated under the ‘‘fivedot’’ model,

the performance of algorithm Sec7.02 (with $\sigma = 0.02$) is worse than Sec7.2 (with $\sigma = 0.2$) when both are run for 2000 iterations with last 1/4 (i.e. 500 iterations) used in evaluation. This may be because the algorithm Sec7.02 updates much more slowly due to the small σ . We suspect that running Sec7.02 longer will improve the performance. To confirm this, we ran 50 more simulations under the “fivedot” model, with the number of iterations being 20000. (Each simulation now takes about 70 minutes.) In Figure 1, the average performance (training error and test error) over 50 simulations are plotted, which are averaged over 40 time periods, each with 500-iterations: Iterations 1-500, Iterations 501-1000, ..., Iterations 19501-20000. We note that in this case indeed there is a slow improvement over time in classification performance.

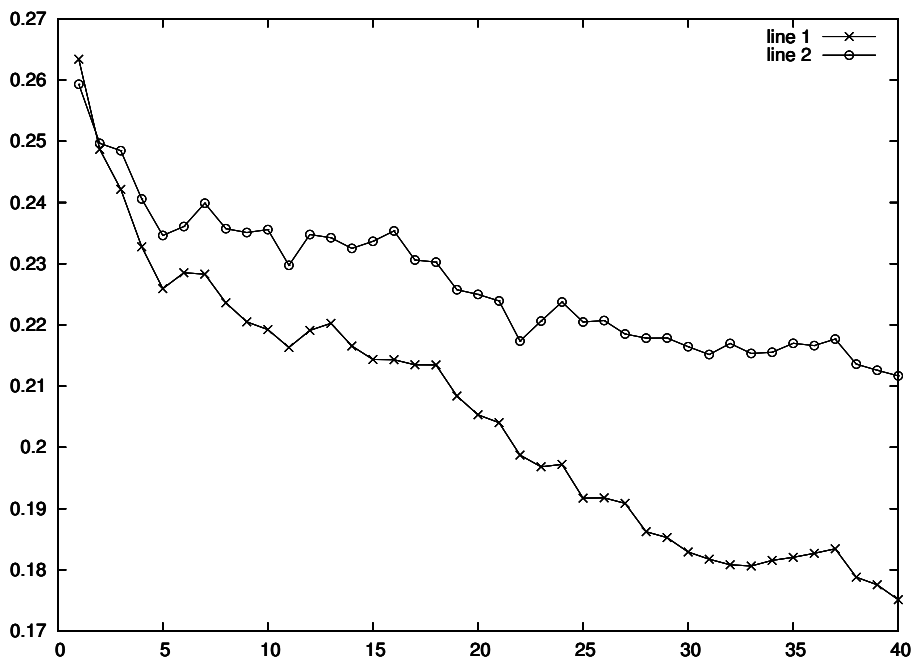


Figure 1: Mean performance over 50 simulations from the ‘fivedot’ model, averaged over 40 time periods each with 500 iterations. Crosses are for training errors and circles are for testing errors.

- (Choice on the inverse temperature ψ .) Regarding the choice of ψ , up to this point we used $\psi = 1$. Now we run the analysis for $\psi = (1/8, 1/4, 1/2, 1, 2, 4, 8, 16)$ using the Metropolis method for 2000 and 4000 iterations, where the average performances of last

1/4 of iterations are used, and averaged over 50 simulations. The average test errors are reported below in Table 3, where it seems that any choice $\psi \geq 1$ works well.

Table 3: Average test error over 50 simulations using Metropolis. (β_1 unrestricted)

model='mvn'								
psi	1/8	1/4	1/2	1	2	4	8	16
average over								
(1501-2000)	0.487	0.454	0.293	0.106	0.089	0.074	0.088	0.081
(3000-4000)	0.485	0.452	0.267	0.107	0.078	0.075	0.083	0.075
model='fivedot'								
psi	1/8	1/4	1/2	1	2	4	8	16
average over								
(1501-2000)	0.427	0.368	0.284	0.174	0.155	0.156	0.179	0.146
(3000-4000)	0.427	0.372	0.287	0.185	0.146	0.154	0.185	0.144

Note: In the Metropolis algorithm here, Steps I or II are randomly chosen with probability 0.5 each, and followed by III in each iteration.

4 Discussion

Jiang and Tanner (2008, Section 7) proposed an algorithm for simulating from the Gibbs posterior. In this paper, we show that this algorithm may perform poorly even after a reasonably long computation time, due to a number of reasons. Although removing the normalization of a signed regression coefficient may improve the performance, we propose an alternative Metropolis algorithm for simulating from the Gibbs posterior, which generally performs well and converges much faster. The proposed Metropolis algorithm is compared to the Sec7 algorithms (based on Jiang and Tanner 2008) as well as to the Lee.et.al. algorithm (based on Lee et al. 2003) when applied to some simulated data.

When data are generated under the “fivedot” model, the Lee.et.al. algorithm performs

the worst, while all other algorithms have smaller testing errors. All these methods are designed to generate linear classification rules. The performance is different because Lee.et.al. method uses a likelihood based posterior which requires the correct model specification. Here, however, the linear classification rules are ‘misspecified’ in the sense that it will not generate the best possible Bayes rule. In this case, the Gibbs posterior based on sample classification errors (whether smoothed or not) performs better.

When data are generated under the “mvn” model, the Lee.et.al. algorithm performs very well. The reason is that their method uses a likelihood based on probit regression of $y|x$, which is very close to the true model of logistic regression (as implied by the mixture of two normal distributions of $x|y$). The performance is even better than that of the Gibbs posterior using the Metropolis algorithm, which directly uses the empirical classification error to construct the posterior and does not require the correct specification of the likelihood. This shows that while the likelihood-based posterior is model dependent, when the model is not very much misspecified, its performance can still be very good.

Taking into account these experiments, we see that the Metropolis is generally preferable to the Sec7 methods, and works much faster than all other methods (especially when $K = \dim(x)$ is high). It can also work much better than the Lee.et.al. method when there is model misspecification and is not much worse when the model is correctly specified.

References

- Brown, P., Vannucci, M. and Fearn (2002). Bayesian model averaging with selection of regressors. *Journal of Royal Statistical Society B*, 64, 519-536.
- Horowitz, J. L., (1992). A smoothed maximum score estimator for the binary response model. *Econometrica* 60, 505-531.
- Jiang, W. and Tanner, M. A. (2008). Gibbs posterior for variable selection in high dimensional classification and data mining. *Annals of Statistics*. 36, 2207-2231.
- Lee, K. E., Sha, N., Dougherty, E. R., Vannucci, M. and Mallick, B. K. (2003). Gene selection: a Bayesian variable selection approach. *Bioinformatics* 19, 90-97.
- Zhang, T. (2006). Information Theoretical Upper and Lower Bounds for Statistical Estimation. *IEEE Transaction on Information Theory* 52, 1307- 1321.