Advanced MCMC Final Project

Christian Paul Ryan Morgan

October 2024

1 Model Selection

Design and implement an algorithm that returns samples of $(S, \beta(S))$. Explicitly describe the cross-model moves (i.e., those that move between $(S, \beta(S))$ and $(S', \beta(S'))$ where S, S' are distinct subsets of $\{1, 2, \ldots, p\}$).

In this problem, we have p covariates, $X_1, \ldots, X_p$, and an outcome Y. We have 2^p possible regression models, one for each subset S of covariates, and each has its own parameter $\beta^{(S)}$. We wish to perform inference on $(S, \beta^{(S)})$.

We specify a uniform prior on S, so that $p(S) = 2^{-p}$ for all subsets.

1. Design and implement an algorithm that returns samples of $(S, \beta^{(S)})$.

Let's first generate a Uniform random variable R from [0,1] to determine what type of move we would like.

Now, we describe the model moves. There are 4 possible types of moves.

The first possible move is a "stay" move. In this move, we let $(S, \beta) \to (S, \beta')$. We will perform a standard Metropolis-Hasting move here, and let $\beta' \sim N_{|S|}(0, I_{|S|})$. We perform this move if $R \in [0, .25)$.

The second possible move is a cross model move, called "add". Here, $(S, \beta) \to (S', \beta')$, where $|S'| = |S| + 1$. Let $U \sim Unif[0, 1]$. Then let $h(\beta_1, U) = (\beta_1, \beta_2)$. We perform this move if $R \in [.25, .5)$.

The third possible move is "delete". Let $(S, \beta) \to (S', \beta')$, where $|S'| = |S| - 1$. Let $h(\beta_1, \beta_2) = (\beta_1, U)$. We perform this move if $R \in [0.5, .75)$.

The last type of move is "swap". Again, $(S, \beta) \to (S', \beta')$, with $|S'| = |S|$. We let $h(\beta^S) = (\beta^{S'})$. We perform this move if $R \in [.75, 1]$.

In the following figures, we see that our simulated posterior probabilities lead to successful model selection for both 5 and 10 covariates; the true non-zero coefficients in the data have high posterior probabilities under simulation.

```
# n = 500, p = 5
data <- gen_data(500, 5)
sim1 <- run_sim(y = data$y, data_mat = data$data_mat, m = 50000)
s_probs <- get_sim_post_prob(sim1)
#posterior prob
s_probs %>%
  mutate(prob = n / 50000)
```

```
## # A tibble: 7 x 7
## # Groups:   X1, X2, X3, X4, X5 [7]
##   X1    X2    X3    X4    X5        n    prob
##   <chr> <chr> <chr> <chr> <chr> <int>   <dbl>
## 1 FALSE FALSE TRUE  TRUE  TRUE  36416 0.728
## 2 TRUE  FALSE TRUE  TRUE  TRUE   6111 0.122
## 3 FALSE TRUE  TRUE  TRUE  TRUE   4949 0.0990
## 4 TRUE  TRUE  TRUE  TRUE  TRUE   2512 0.0502
## 5 FALSE FALSE FALSE TRUE  TRUE      8 0.00016
## 6 TRUE  FALSE FALSE FALSE TRUE      3 0.00006
## 7 FALSE FALSE FALSE FALSE TRUE      1 0.00002
```

```
# true S (non zero)
sort(data$coef_index)
```

```
## [1] 3 4 5
```

Figure 1: Markdown Comparison of True vs. Selected Coefficients (P=5)

```
# try again for p = 10
data10 <- gen_data(1000, 10)
sim10 <- run_sim(y = data10$y, data_mat = data10$data_mat, m = 50000)
s_probs10 <- get_sim_post_prob(sim10)
#posterior prob
s_probs10 %>%
  mutate(prob = n / 50000)
```

```
## # A tibble: 82 x 12
## # Groups:   X1, X2, X3, X4, X5, X6, X7, X8, X9, X10 [82]
##    X1    X2    X3    X4    X5    X6    X7    X8    X9    X10       n   prob
##    <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <int>  <dbl>
##  1 FALSE FALSE TRUE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE 41218 0.824
##  2 FALSE FALSE TRUE  FALSE FALSE FALSE FALSE FALSE TRUE  FALSE  1742 0.0348
##  3 FALSE FALSE TRUE  FALSE TRUE  FALSE FALSE FALSE FALSE FALSE  1159 0.0232
##  4 FALSE FALSE TRUE  FALSE FALSE TRUE  FALSE FALSE FALSE FALSE   819 0.0164
##  5 FALSE FALSE TRUE  FALSE FALSE FALSE TRUE  FALSE FALSE FALSE   756 0.0151
##  6 FALSE FALSE TRUE  FALSE FALSE FALSE FALSE FALSE FALSE TRUE    662 0.0132
##  7 TRUE  FALSE TRUE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE   596 0.0119
##  8 FALSE FALSE TRUE  FALSE FALSE FALSE FALSE TRUE  FALSE FALSE   547 0.0109
##  9 FALSE TRUE  TRUE  FALSE FALSE FALSE FALSE FALSE FALSE FALSE   539 0.0108
## 10 FALSE FALSE TRUE  TRUE  FALSE FALSE FALSE FALSE FALSE FALSE   521 0.0104
## # i 72 more rows
```

```
# true S (non zero)
sort(data10$coef_index)
```

```
## [1] 3
```

Figure 2: Markdown Comparison of True vs. Selected Coefficients (P=10)

2 MALA for Standard Normals

How sample size grows as MCMC iterations increases
Examining the bias of estimating the first and second moments of the distribution, which are known in advance

Parameters: $\lambda, d, \theta \in \mathbb{R}^d$ and

$$\begin{aligned}
&\theta_{k+1} = \theta_k + \frac{1}{2}\lambda^2 \nabla log(\pi(\theta_k)) + \epsilon_k \\
&\epsilon \sim \mathcal{N}(0, \lambda^2 I_d) \\
&\theta_0 \sim \mathcal{N}(0, I_d)
\end{aligned}$$

To derive the sampler,

$$\begin{aligned}
&\pi(\theta) = \frac{1}{c} exp(\frac{-1}{2}\|\theta\|)_2^2 \\
&log(\pi(\theta)) = -log(c) - \frac{1}{2}\|\theta\|_2^2 \\
&\nabla log(\pi(\theta)) = -\theta \\
&\nabla^2 log(\pi(\theta)) = -I
\end{aligned} \tag{1}$$

$$\begin{aligned}
\theta_{k+1} &= \theta_k + \frac{\lambda^2}{2}(-\theta_k) + \epsilon_k \\
&= \theta_k - \frac{\lambda^2}{2}\theta_k + \epsilon_k \\
&= (1 - \frac{\lambda^2}{2})\theta_k + \epsilon_k
\end{aligned}$$

We tested our the performance of our MALA sampler on a grid of values of dimension $d = 1, 10, 100$ and step sizes $\lambda = 0.25, 0.5, 1, 2$. For each experiment, we used a burn-in period of 1000 iterations. We ran experiments where we kept 1000 to 10000 iterations, incremented by 1000. We evaluate the performance of our sampler using effective sample size (ESS) and the bias of the true mean and variance of the distribution.

We find that our MALA sampler is sensitive to both d and λ. Figure 3 provides a visualization of our results. When $d = 1$, larger step sizes tend to lead to higher ESS and reduced bias. However, as d grows, smaller step sizes tend to lead to better sampling diagnostics. In general, our sampler struggles to fully explore the sample space in high dimensions, even with favorable hyperparameter settings.

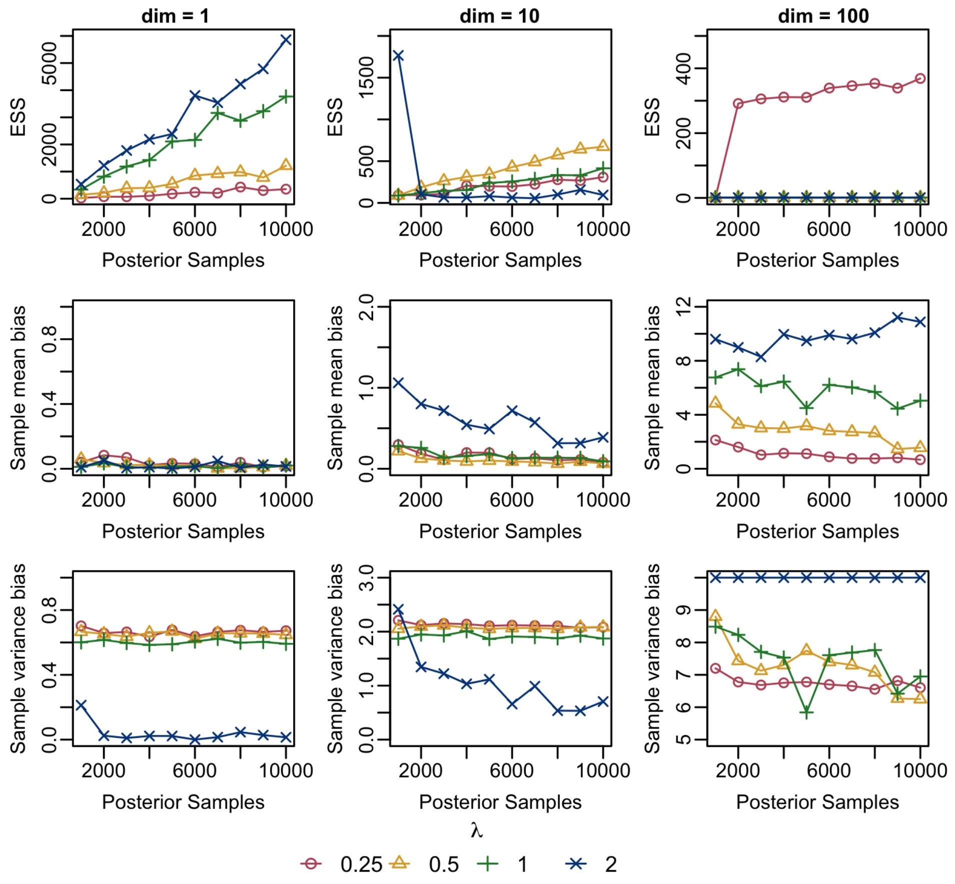

Figure 3: Diagnostics of standard normal MALA sampler experiments.

3 MALA for Bayesian Logistic Regression

3.1 Derivations

We first derive the MALA iteration for Bayesian Logistic Regression. To do this, we derive two quantities: the gradient of the log of the target distribution, and the acceptance ratio when used in the metropolis-hasting framework.

Gradient Derivation. The target distribution in our example is the posterior distribution of $\beta \in \mathbb{R}^p$, conditional on observations $y_1, ..., y_n \in \mathbb{R}$ following the model

$$y_i|\beta \sim \text{Bern}([1 + \exp(-x_i^\intercal \beta)]^{-1}) \tag{2}$$

$$\beta \sim \mathcal{N}(0, I). \tag{3}$$

The resulting log posterior likelihood is (up to a constant term independent of β)

$$\log(p(\beta|y_1,...,y_n)) = \sum_{i=1}^{n} \log(p(y_i|\beta)) + \log(p(\beta)) \tag{4}$$

$$= \left(\sum_{i=1}^{n} -y_i \log(1+\exp(-x_i^\intercal \beta)) + (1-y_i)\log\left(\frac{\exp(-x_i^\intercal\beta)}{1+\exp(-x_i^\intercal\beta)}\right)\right) - .5||\beta||_2^2 \tag{5}$$

$$= \sum_{i=1}^{n} -\log\left(1+\exp(-x_i^\intercal\beta)\right) + (y_i-1)(x_i^\intercal\beta) - .5||\beta||_2^2. \tag{6}$$

$$\tag{7}$$

Taking the gradient of equation 4 results in

$$\nabla_\beta \log(p(\beta|y_1,...,y_n)) = \sum_{i=1}^{n} \frac{1}{1+\exp(x_i^\intercal\beta)} x_i + (y_i-1)x_i. \tag{8}$$

Acceptance Ratio Derivation. To fully specify MALA, we must calculate the acceptance ratio; for β and a proposed β'

$$\alpha_3 = \min\left\{1, \frac{p(\beta'|y_1,...,y_n)q(\beta|\beta')}{p(\beta|y_1,...,y_n)q(\beta'|\beta)}\right\} = \min\left\{1, \frac{p(y_1,...,y_n|\beta')}{p(y_1,...,y_n|\beta)}\right\}. \tag{9}$$

For numerical stability, we compute $\log(\alpha_3)$ in our algorithm.

3.2 Results

Having derived the gradient term and acceptance ratio in equations 4 and 9, we now present our numerical results. Our experiment parameters are summarized below.

1. We range the number parameters, p, in the set $\{1, 10, 100\}$.
2. We range the number step size, λ, in the set $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$.
3. For each p, $n = 1000$ points are samples from the model, and 20000 MALA steps were generated with 5000 discarded as a burn in. For HMC, 4 chains were used, each taking 2000 iterations.

We implement the code for MALA in julia. The results of the experiment for MALA are in Table 1, and the results for HMC are in 2.

Table 1: Results for MALA. Going from left to right, the first table shows effective sample size, time (in seconds), and posterior bias ($||\beta_* - \hat{\beta}||_2^2$).

Results MALA	ESS			Time			Posterior Bias		
λ/p	$p=1$	$p=10$	$p=100$	$p=1$	$p=10$	$p=100$	$p=1$	$p=10$	$p=100$
$\lambda=10^{-4}$	5.54	24.77	295.56	3.2	5.10	40.29	0.16	9.0	116.76
$\lambda=10^{-3}$	5.52	41.83	324.29	1.9	5.12	42.15	0.02	3.75	87.1162
$\lambda=10^{-2}$	65.16	73.3	300.62	1.9	5.11	43.8	0.0003	0.25	8.16
$\lambda=10^{-1}$	4832.35	2020.81	710.28	1.9	5.43	43.9	0.0005	0.23	8.11

Table 2: Results for HMS. Going from left to right, the first table shows effective sample size, time (in seconds), and posterior bias ($||\beta_* - \hat{\beta}||_2^2$).

Results HMC	ESS			Time			Posterior Bias		
	$p = 1$	$p = 10$	$p = 100$	$p = 1$	$p = 10$	$p = 100$	$p = 1$	$p = 10$	$p = 100$
	4000	4000	4000	22.741	48.092	177.646	0.0004	0.23	144.79