Unit 3: Examples of Metropolis–Hastings-type Algorithms

3.1 Examples of Metropolis–Hastings Algorithms

In the last unit, we constructed Metropolis–Hastings algorithms targeting a probability distribution π defined on the space $\mathcal{X}$, and we found the optimal acceptance probability function according to Peskun ordering. We summarize it in Algorithm 3.1 and discuss a few common choices of the proposal scheme. Note that to implement Algorithm 3.1, we only need to be able to evaluate $\pi(x)$ up to a normalizing constant, which makes it useful for application to Bayesian statistics. Recall that we always assume the densities $q(x, y) = Q(x, \mathrm{d}y)/\mu(\mathrm{d}y)$ and $\pi(x) = \pi(\mathrm{d}x)/\mu(\mathrm{d}x)$ exist.

Algorithm 3.1 (Metropolis–Hastings algorithm)**.** Initialize the sampler at some X_0 (which can be deterministic or stochastic). For $t = 1, 2, \ldots,$

(i) Sample Y from the distribution $Q(X_{t-1}, \cdot)$.

(ii) Calculate the acceptance probability $\alpha(X_{t-1}, Y)$ where

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

(iii) Generate $U \sim \mathrm{Unif}([0, 1])$. If $U \leq \alpha(X_{t-1}, Y)$, set $X_t = Y$; otherwise, set $X_t = X_{t-1}$.

Example 3.1. When $Q(x, \cdot)$ is a fixed probability distribution independent of x, Algorithm 3.1 is known as the independence Metropolis–Hastings sampler. Its theoretical properties have been well studied in the literature [10, 16, 2], but it is rarely used in practice.

Example 3.2. Let $\mathcal{X} \subset \mathbb{R}^d$ for some $d \geq 1$. Pick some symmetric density function $\tilde{q}(x)$ (i.e., $\tilde{q}(x) = \tilde{q}(-x)$ for every x), and define $Q(x, \mathrm{d}y) = \tilde{q}(y - x)\mu(\mathrm{d}y)$. In this case, $\alpha(x, y)$ is simplified to $1 \wedge (\pi(y)/\pi(x))$, and Algorithm 3.1 is often known as the random walk Metropolis (RWM) algorithm. For example, if $\mathcal{X}$ is continuous, we can let $Q(x, \cdot)$ be a normal distribution with mean 0 and variance ϕ for some fixed $\phi > 0$. Such choices are very popular in practice, since the resulting Metropolis–Hastings algorithm is very straightforward to implement.

Example 3.3. Let $\mathcal{X} = \mathbb{R}^d$ for some $d \geq 1$ and μ be the Lebesgue measure. Let $Q(x, \cdot)$ be

$$N\left(x + \frac{\phi}{2}\nabla \log \pi(x),\ \phi I\right),$$

where $\phi > 0$ is some fixed constant. The resulting algorithm is known as Metropolis-adjusted Langevin algorithm (MALA). We point out two major differences between MALA and RWM. First, the proposal distribution of MALA depends on π, while that of RWM usually does not. Involving $\nabla \log \pi(x)$ in the proposal is often helpful, because this is the direction where $\log \pi$ increases fastest. Second, the proposal of MALA is usually not symmetric, i.e., $q(x, y) \neq q(y, x)$. So to calculate $\alpha(x, y)$, we need to evaluate both $\nabla \log \pi(x)$ and $\nabla \log \pi(y)$. MALA is very popular in practice, and its convergence properties and optimal choice of ϕ have also been extensively studied [13, 4, 17, 15]. It is a special case of a more sophisticated and also very popular algorithm known as Hamiltonian Monte Carlo [12]. When π has highly correlated coordinates, MALA can be quite inefficient. One solution is to change the proposal distribution $Q(x, \cdot)$ to

$$N\left(x + \frac{\phi}{2}\Sigma\,\nabla \log \pi(x),\ \phi\Sigma\right),$$

where Σ is the covariance matrix (or an estimate of it) under π. A further generalization of this proposal yields the Riemann manifold MALA [8].

Example 3.4. Our next example is the random-scan Gibbs sampler. Assume that $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_p$ is a product space, and let $\pi(x_i \,|\, x_{-i})$ denote the conditional distribution of the i-th coordinate given the other $p-1$ coordinates. The proposal distribution $Q(x, \cdot)$ encodes the following procedure: first, we draw i from the uniform distribution on $\{1, 2, \dots, p\}$; second, we modify x_i by resampling it from the conditional distribution $\pi(x_i \,|\, x_{-i})$ (the other coordinates remain unchanged). Denote the proposed state by $y = (x_1, \dots, x_{i-1}, y_i, x_{i+1}, \dots, x_p)$, and the proposal density of such y can be calculated by $q(x, y) = \pi(y_i \,|\, x_{-i})/p$.[1] Now let's consider the acceptance probability. Since $\pi(x) = \pi(x_{-i})\pi(x_i \,|\, x_{-i})$, we have

$$\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)} = \frac{\pi(y)\pi(x_i \,|\, x_{-i})/p}{\pi(x)\pi(y_i \,|\, x_{-i})/p} = \frac{\pi(x_{-i})}{\pi(x_{-i})} = 1.$$

Hence, the acceptance probability is always 1.

Example 3.5. We can of course combine the Gibbs update with other proposals. Let's assume $\mathcal{X} = \mathbb{R}^2$ and π is continuous. Suppose that we can sample from $\pi(x_1 \,|\, x_2)$ but not $\pi(x_2 \,|\, x_1)$. Then, we can use the following proposal distribution at $x = (x_1, x_2)$:

(a) With probability 1/2, we propose $x' = (x_1', x_2)$ where x_1' is drawn from $\pi(x_1 \,|\, x_2)$.

(b) With probability 1/2, we propose $x' = (x_1, x_2')$ where x_2' is drawn from $N(x_2, \sigma^2)$.

So we can write down the proposal density as

$$q((x_1, x_2), (x_1', x_2')) = \begin{cases} \frac{1}{2}\pi(x_1' \,|\, x_2), & \text{if } x_2 = x_2', \\ \frac{1}{2\sqrt{2\pi\sigma^2}} e^{-(x_2' - x_2)/2\sigma^2}, & \text{if } x_1 = x_1'. \end{cases}$$

(Since we assume π is continuous, we do not need to worry about the case $x_1 = x_1', x_2 = x_2'$.) It is easy to check that the first type of proposal is always accepted, while the second type is accepted with probability $1 \wedge \pi(x')/\pi(x)$. Such a scheme is sometimes referred to as Metropolis-within-Gibbs sampling.

[1] Note that if $\mathcal{X}_i = \mathbb{R}$, the distribution $Q(x, \cdot)$ is degenerate and cannot have a density with respect to the Lebesgue measure on $\mathbb{R}^p$. But we do not need to worry about this, and a rigorous justification for $q(x, y)$ will be given in Unit 5.

3.2 Combining Metropolis–Hastings Schemes

The following simple lemma shows that we can easily devise new MCMC algorithms by "combining" multiple Metropolis–Hastings algorithms.

Lemma 3.1. *Let P_1 and P_2 be two transition kernels with stationary distribution π. For any $a \in (0,1)$, both P_1P_2 and $aP_1 + (1-a)P_2$ are transition kernels that have π as a stationary distribution, where the two kernels are defined by*

$$
\begin{aligned}
(P_1P_2)(x, B) &= \int_{\mathcal{X}} P_1(x, \mathrm{d}y) P_2(y, B), \\
(aP_1 + (1-a)P_2)(x, B) &= aP_1(x, B) + (1-a)P_2(x, B),
\end{aligned}
$$

for all $x \in \mathcal{X}$, $B \in \mathcal{B}(\mathcal{X})$.

Proof. It is easy to check that both P_1P_2 and $aP_1 + (1-a)P_2$ are indeed transition kernels. Since

$$
\begin{aligned}
(\pi(P_1P_2))(B) = \int_{\mathcal{X}} (P_1P_2)(x, B)\pi(\mathrm{d}x) &= \int_{\mathcal{X}^2} P_1(x, \mathrm{d}y) P_2(y, B)\pi(\mathrm{d}x) \\
&= \int_{\mathcal{X}} P_2(y, B)\pi(\mathrm{d}y) = \pi(B),
\end{aligned}
$$

P_1P_2 is invarinat with respect to π. The proof for $aP_1 + (1-a)P_2$ is easy. □

Remark 3.1. Note that even if P_1, P_2 are both reversible, P_1P_2 is usually not reversible.

We can interpret P_1, P_2 in Lemma 3.1 as two Metropolis–Hastings algorithms targeting π equipped with different proposals, which we denote by Q_1, Q_2. By Lemma 3.1, we can also simulate Markov chains P_1P_2 or $aP_1 + (1-a)P_2$ to generate samples from π. This yields the following two algorithms. The extension to three or more Metropolis–Hastings kernels is straightforward.

Algorithm 3.2. For $t = 1, 2, \dots,$

(i) Perform one Metropolis–Hastings step with proposal Q_1 and generate $X' \sim P_1(X_{t-1}, \cdot)$.

(ii) Perform one Metropolis–Hastings step with proposal Q_2 and generate $X_t \sim P_2(X', \cdot)$.

Algorithm 3.3. For $t = 1, 2, \dots,$

(i) Generate $W \sim \mathrm{Unif}([0,1])$. If $W \leq a$, set $k = 1$; otherwise, set $k = 2$.

(ii) Sample Y from the distribution $Q_k(X_{t-1}, \cdot)$.

(iii) Calculate the acceptance probability $\alpha_k(X_{t-1}, Y)$ where

$$
\alpha_k(x, y) = \min\left\{1, \frac{\pi(y) q_k(y, x)}{\pi(x) q_k(x, y)}\right\}.
$$

(iv) Generate $U \sim \text{Unif}([0,1])$. If $U \leq \alpha_k(X_{t-1}, Y)$, set $X_t = Y$; otherwise, set $X_t = X_{t-1}$.

There is another way to make use of multiple proposal schemes. We can simply define

$$Q = aQ_1 + (1-a)Q_2,$$

and implement the resulting Metropolis–Hastings algorithm. This will be more efficient than Algorithm 3.3 according to Peskun ordering; see the exercise below. However, when combining a large number of proposal schemes, calculating the acceptance probability (which requires evaluating the proposal probabilities of all proposal schemes involved) can be time-consuming.

Exercise 3.1. Let $Q_1, Q_2, \dots, Q_m$ be transition kernels with densities $q_1, q_2, \dots, q_m$, and let $a_1, a_2, \dots, a_m$ be non-negative constants that sum up to 1. Define $Q = \sum_{i=1}^m a_i Q_i$; denote its density by q. For $x \neq y$, define

$$\begin{aligned} \bar{p}(x,y) &= q(x,y) \min\left\{1, \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}\right\}, \\ p(x,y) &= \sum_{i=1}^m a_i q_i(x,y) \min\left\{1, \frac{\pi(y)q_i(y,x)}{\pi(x)q_i(x,y)}\right\}. \end{aligned}$$

Show that $\bar{p}(x,y) \geq p(x,y)$.

3.3 Multiple-try Metropolis Algorithm

On continuous spaces, we have seen in Example 3.3 that one can use gradient information to design efficient proposals. What if the gradient information is not available, e.g. when $\mathcal{X}$ is discrete or $\nabla \log \pi(x)$ is extremely difficult to calculate? In such scenarios, we can use an algorithm called multiple-try Metropolis (MTM) algorithm [11], whose dynamics is, to some extent, similar to that of MALA.

Fix a proposal kernel Q, which is usually chosen to be a random walk. Introduce a weighting function $w(x,y) > 0$, which represents our preference for proposing y at x. It needs to satisfy

$$\pi(x)q(x,y)w(x,y) = \pi(y)q(y,x)w(y,x). \tag{1}$$

This may remind you of the condition we imposed on the acceptance probability function when constructing the Metropolis–Hastings algorithm, but note that w does not need to be bounded, and we will only need to be able to evaluate $w(x,y)$ up to a normalizing constant. Some examples of w include

$$w(x,y) = \pi(y)q(y,x), \quad w(x,y) = \sqrt{\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}}, \quad w(x,y) = \min\left\{1, \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}\right\}.$$

MTM proceeds as follows.

Algorithm 3.4 (Multiple-try Metropolis)**.** Fix a positive integer $N \geq 1$. For $t = 1, 2, \dots,$

(i) Sample $Y_1, \dots, Y_N$ independently from the distribution $Q(X_{t-1}, \cdot)$.

(ii) Calculate $w(X_{t-1}, Y_i)$ for $i = 1, \dots, N$, and draw Y^* from $\{Y_1, \dots, Y_N\}$ with probability proportional to $w(X_{t-1}, Y_i)$.

(iii) Sample $Z_1, \dots, Z_{N-1}$ independently from the distribution $Q(Y^*, \cdot)$, and calculate the acceptance probability

$$\alpha = \min \left\{ 1, \ \frac{w(X_{t-1}, Y_1) + \cdots + w(X_{t-1}, Y_N)}{w(Y^*, Z_1) + \cdots + w(Y^*, Z_{N-1}) + w(Y^*, X_{t-1})} \right\}.$$

(iv) Generate $U \sim \mathrm{Unif}([0,1])$. If $U \leq \alpha$, set $X_t = Y^*$; otherwise, set $X_t = X_{t-1}$.

Note that in the above algorithm, the exact probability of proposing Y^* at X_{t-1} is not calculated, which would require a summation over all possible choices of $(Y_1, \dots, Y_n)$. The acceptance probability depends on not only X_{t-1}, Y^* but also $Y_1, \dots, Y_N, Z_1, \dots, Z_{N-1}$. This is similar to Algorithm 3.3, where the acceptance probability depends on which Q_k is used, and we do not need to calculate the marginal proposal probability with k integrated out. Now let's prove that the MTM algorithm is indeed invariant with respect to π.

Theorem 3.1. *Let P denote the transition kernel of Algorithm 3.4. Then P is reversible with respect to π.*

Proof. We check the detailed balance condition $\pi(\mathrm{d}x)P(x, \mathrm{d}y) = \pi(\mathrm{d}y)P(y, \mathrm{d}x)$ for any $x \neq y$. Observe that given $Y_1 = y_1, \dots, Y_N = y_N, Z_1 = z_1, \dots, Z_{N-1} = z_{N-1}$, the probability of moving from $X_{t-1} = x$ to $X_t = y$ is

$$\begin{aligned} &\frac{w(x,y) \sum_{j=1}^N \mathbb{1}(y = y_j)}{\sum_{i=1}^N w(x, y_i)} \min \left\{ 1, \ \frac{\sum_{i=1}^N w(x, y_i)}{w(y, x) + \sum_{i=1}^{N-1} w(y, z_i)} \right\} \\ = &\sum_{j=1}^N \frac{w(x, y_j) \mathbb{1}(y = y_j)}{\max\left\{ \sum_{i=1}^N w(x, y_i), \ w(y_j, x) + \sum_{i=1}^{N-1} w(y_j, z_i) \right\}}. \end{aligned}$$

By integrating over $Y_1, \dots, Y_N, Z_1, \dots, Z_{N-1}$, we get

$$P(x, B) = \sum_{j=1}^N \int \frac{w(x, y_j) \mathbb{1}_B(y_j) \left\{ \prod_{i=1}^N Q(x, \mathrm{d}y_i) \right\} \left\{ \prod_{i=1}^{N-1} Q(y_j, \mathrm{d}z_i) \right\}}{\max\left\{ \sum_{i=1}^N w(x, y_i), \ w(y_j, x) + \sum_{i=1}^{N-1} w(y_j, z_i) \right\}}.$$

Since $Y_1, \dots, Y_N$ are i.i.d. (and thus exchangeable), the N terms involved in the above summation are equal. Hence, by considering the case where y_N is selected, we get

$$P(x, B) = N \int \frac{w(x, y) \mathbb{1}_B(y) Q(x, \mathrm{d}y) \left\{ \prod_{i=1}^{N-1} Q(x, \mathrm{d}y_i) \right\} \left\{ \prod_{i=1}^{N-1} Q(y, \mathrm{d}z_i) \right\}}{\max\left\{ w(x, y) + \sum_{i=1}^{N-1} w(x, y_i), \ w(y, x) + \sum_{i=1}^{N-1} w(y, z_i) \right\}}.$$

This shows that

$$\pi(\mathrm{d}x)P(x,\mathrm{d}y) = N \int \frac{\pi(\mathrm{d}x)w(x,y)Q(x,\mathrm{d}y)\left\{\prod_{i=1}^{N-1} Q(x,\mathrm{d}y_i)\right\}\left\{\prod_{i=1}^{N-1} Q(y,\mathrm{d}z_i)\right\}}{\max\left\{w(x,y)+\sum_{i=1}^{N-1} w(x,y_i),\ w(y,x)+\sum_{i=1}^{N-1} w(y,z_i)\right\}}.$$

By (1), this is symmetric in (x, y), which concludes the proof. □

Remark 3.2. MTM can be further generalized by considering exchangeable but dependent candidates; see [5]. The optimal choice of w largely depends on the problem, but it is usually preferable to construct $w(x, y)$ as a function of $\pi(y)q(y, x)/\pi(x)q(x, y)$ [3, 6].

3.4 Spike-and-slab Variable Selection

To illustrate the use of Metropolis–Hastings-type algorithms, let's consider Bayesian spike-and-slab variable selection. The data consists of an $n \times p$ design matrix, denoted by L, and a response vector y. We assume that

$$y = L\beta + \epsilon, \quad \epsilon \sim N(0, \tau^{-1}I),$$

and most entires of β are zero. Let $\gamma \in \{0,1\}^p$ be a function of β such that $\gamma_i = 1$ if and only if $\beta_i \neq 0$. Our goal is to learn γ (and sometimes also β) from the data. Let $|\gamma| = \sum_{i=1}^p \gamma_i$, and let L_γ denote the submatrix of L with columns indexed by γ. Let $p(\tau, \gamma, \beta)$ denote our prior distribution on the parameters. We assume $p(\tau, \gamma, \beta) = p(\beta \,|\, \tau, \gamma)p(\gamma)p(\tau)$ (i.e., γ and τ are independent a priori), and we consider the following standard choice of prior on (τ, β):

$$\begin{aligned} \tau &\sim \mathrm{Gamma}(\kappa_1/2, \kappa_2/2), \\ \beta \,|\, \tau, \gamma &\sim N(0, \tau^{-1}V_\gamma), \end{aligned} \tag{2}$$

where V_γ is an $|\gamma| \times |\gamma|$ positive definite matrix that may depend on L_γ and other hyperparameters. (Note that here we use the shape-rate paratermization of the Gamma distribution.) For example, we can use $V_\gamma = \sigma^2 I$ or $V_\gamma = g(L_\gamma^\top L_\gamma)^{-1}$.

3.4.1 Sampling from $p(\gamma \,|\, y)$

A routine calculation yields the following formula for the marginal likelihood of γ:

$$p(y \,|\, \gamma) \propto \frac{1}{\sqrt{\det(I + L_\gamma^\top L_\gamma V_\gamma)}} \left\{1 - \frac{y^\top L_\gamma (L_\gamma^\top L_\gamma + V_\gamma^{-1})^{-1} L_\gamma^\top y}{y^\top y + \kappa_2}\right\}^{-(n+\kappa_1)/2}. \tag{3}$$

So we can construct MCMC algorithms on $\{0,1\}^p$ targeting the posterior distribution $p(\gamma \,|\, y) \propto p(\gamma)p(y \,|\, \gamma)$ (we assume that the prior $p(\gamma)$ is easy to evaluate).

Metropolis–Hastings sampler. For Metropolis–Hastings algorithms, all we need is to specify the proposal distribution. Here is a popular choice for $Q(\gamma, \cdot)$:

(a) With probability a_{add}, we sample $j \in \{i \colon \gamma_i = 0\}$ change γ_j to 1.

(b) With probability a_{del}, we sample $k \in \{i \colon \gamma_i = 1\}$ change γ_k to 0.

(c) With probability $1 - a_{\text{add}} - a_{\text{del}}$, we sample $j \in \{i \colon \gamma_i = 1\}$ and $k \in \{i \colon \gamma_i = 0\}$ and change γ_j to 0 and γ_k to 1 simultaneously.

The resulting Metropolis–Hastings algorithm is often known as the add-delete-swap sampler. The meaning of "add-delete-swap" is obvious: when we propose changing some γ_j from 0 to 1 (resp. from 1 to 0), we are adding (resp. deleting) a predictor our regression model; when we change γ_j from 0 to 1 and γ_k from 1 to 0, we are swapping one predictor with another. Note that the coordinates j, k can be chosen randomly with equal probability or from some pre-specified distribution. For example, we can first calculate the correlation between L_i and y for each column i and use this to devise the scheme of sampling j, k. A simpler construction is to merge cases (i) and (ii): with probability a, we sample j from $\{1, 2, \dots, p\}$ with equal probability and set $\gamma_j = 1 - \gamma_j$. The convergence rate of this add-delete-swap sampler has been studied in [18].

The main reason why swap moves are often used is that predictors can be highly correlated (an extreme case is that two columns of L are identical). It is entirely possible that the collinearity in L has more complex patterns, and we have to simultaneously modify more than 2 coordinates of γ to move to another good model. In Exercise (3.2), we describe a sampler similar to Algorithm 3.3 that allows us to combine multiple add/delete moves to propose new values of γ.

Random-scan Gibbs sampler. Though we do not know the normalizing constant of $p(\gamma \mid y)$, we can exactly calculate the conditional posterior distribution of γ_i given γ_{-i}, since

$$p(\gamma_i \mid y, \gamma_{-i}) = \frac{p(\gamma_i \mid y, \gamma_{-i})}{p(\gamma_i = 1 \mid y, \gamma_{-i}) + p(\gamma_i = 0 \mid y, \gamma_{-i})},$$

and all terms on the right-hand side can be calculated up to a normalizing constant. However, this is less efficient than the Metropolis–Hastings sampler with only add/delete proposals. To see this, suppose that we have selected the coordinate j and the current value of γ_j is 0. Let $\gamma' = (\gamma_1, \dots, \gamma_{j-1}, 1, \gamma_{j+1}, \dots, \gamma_p)$ (that is, γ' is obtained from γ by adding predictor j). The Gibbs update means that with probability $p(\gamma_j = 1 \mid y, \gamma_{-j})$, we move from γ to γ', and with probability $p(\gamma_j = 0 \mid y, \gamma_{-j})$, we stay at γ. So this is essentially an acceptance-rejection step where we propose γ' and accept it with probability

$$\frac{p(\gamma_j = 1 \mid y, \gamma_{-j})}{p(\gamma_j = 0 \mid y, \gamma_{-j}) + p(\gamma_j = 1 \mid y, \gamma_{-j})} = \frac{p(\gamma' \mid y)}{p(\gamma \mid y) + p(\gamma' \mid y)}.$$

As we have seen in Unit 2, according to Peskun ordering, this is less efficient than accepting γ' with probability $\min\{1, p(\gamma' \mid y)/p(\gamma \mid y)\}$.

Exercise 3.2. Let Q be a proposal kernel such that $Q(\gamma, \mathrm{nb}(\gamma)) = 1$, where

$$\mathrm{nb}(\gamma) = \{\gamma' \colon |\gamma' - \gamma| = 1\}.$$

That is, Q defines how to add or delete one predictor from γ. Let N be a fixed positive integer. Consider an MCMC algorithm which updates the current state γ_0 as follows.

(i) Sample k from the uniform distribution on $\{1, 2, \dots, N\}$.

(ii) Draw $\gamma_i \sim Q(\gamma_{i-1}, \cdot)$ for $i = 1, 2, \dots, k$.

(iii) Let γ_k be our proposal, and we calculate the acceptance probability

$$\alpha = \min\left\{1, \frac{p(\gamma_k \,|\, y) \prod_{i=1}^k q(\gamma_i, \gamma_{i-1})}{p(\gamma_0 \,|\, y) \prod_{i=1}^k q(\gamma_{i-1}, \gamma_i)}\right\}.$$

(iv) With probability α, we move to γ_k; with probability $1 - \alpha$, we stay at γ_0.

Show that this algorithm is reversible with respect to $p(\gamma \,|\, y)$.

3.4.2 Sampling from $p(\beta, \tau \,|\, y)$

Once we have an MCMC sampler targeting $p(\gamma \,|\, y)$, we can generate samples of τ and β by sampling from $p(\tau \,|\, y, \gamma)$ and $p(\beta \,|\, y, \gamma, \tau)$. But let's consider an alternative approach that directly targets $p(\beta, \tau \,|\, y)$ without explicitly sampling γ. To be able to evaluate $p(\beta)$, we let $V_\gamma = \sigma^2 I$ in (2), and we use

$$p(\gamma) = \rho^{|\gamma|}(1 - \rho)^{p - |\gamma|}$$

where $\rho \in (0, 1)$ is another hyperparameter. In this case, we can integrate out γ and get

$$\beta \,|\, \tau \sim (1 - \rho)\delta_0 + \rho N(0, \tau^{-1}\sigma^2).$$

So we can evaluate the posterior $p(\beta, \tau \,|\, y) \propto p(\beta \,|\, \tau)p(\tau)p(y \,|\, \beta, \tau)$ up to a normalizing constant. Metropolis–Hastings algorithms can be constructed that target this posterior distribution. Note that when proposing a new value for β_j, we should use a mixture of δ_0 and some continuous distribution.

Random-scan Gibbs sampler. The Gibbs sampler is likely to be more efficient than Metropolis–Hastings samplers in this case. Observe that the conditional posterior distribution $p(\tau \,|\, y, \beta)$ is still a Gamma distribution, which is used to update τ. The conditional posterior distribution of β_j given β_{-j} and τ is still a mixture of δ_0 and normal distribution, which is also easy to sample from. The expression of this conditional posterior distribution is left as an exercise.

Exercise 3.3. Find the expression for $p(\beta_j \,|\, y, \beta_{-j}, \tau)$. Note that this is the density with respect to the dominating measure $\delta_0 + \lambda$, where λ denotes the Lebesgue measure.

3.4.3 Numrical Implementation

For large data sets, an efficient implementation of the above MCMC algorithms can be extremely important. For example, if $|\gamma|$ tends to be large (say, a few hundred), then how to calculate $\hat{\beta}_\gamma := (L_\gamma^\top L_\gamma + V_\gamma^{-1})^{-1} L_\gamma^\top y$ and $\det(I + L_\gamma^\top L_\gamma V_\gamma)$ in (3) is crucial. In particular, directly calculating the inverse is unnecessary and can be quite inefficient.

Assume that $V_\gamma = \sigma^2 I$. A popular and efficient approach is to calculate the Cholesky decomposition of $R_\gamma^\top R_\gamma = L_\gamma^\top L_\gamma + \sigma^{-2} I$, where R is an upper-triangular matrix. Once R_γ is given, $\hat{\beta}_\gamma$ can be obtained by forward/backward substitution (note that all you need is to solve a linear system), and $\det(L_\gamma^\top L_\gamma + \sigma^{-2} I) = \det(R_\gamma)^2$ (the determinant of a triangular matrix is just the product of diagonals). For the add-delete-swap sampler, this scheme can be implemented very efficiently, since R_γ can be obtained by updating so that the complexity of each iteration is $O(|\gamma|^2)$ [14, 7]. If σ is unknown and also needs to be sampled, this scheme is not feasible, but an efficient solution, still based on Cholesky updating, is given in [19]. Other factors can also affect the implementation (e.g., how large n and p are). For an excellent reference on numerical linear algebra, see, e.g., [9].

Other than the spike-and-slab prior discussed in this note, another popular approach to Bayesian variable selection is to use the so-called horseshoe prior, and its numerical implementation has been studied in [1].

References

[1] Anirban Bhattacharya, Antik Chakraborty, and Bani K Mallick. Fast sampling with gaussian scale mixture priors in high-dimensional regression. *Biometrika*, page asw042, 2016.

[2] Austin Brown and Galin L Jones. Exact convergence analysis for Metropolis–Hastings independence samplers in Wasserstein distances. *Journal of Applied Probability*, 61(1):33–54, 2024.

[3] Hyunwoong Chang, Changwoo J Lee, Zhao Tang Luo, Huiyan Sang, and Quan Zhou. Rapidly mixing multiple-try Metropolis algorithms for model selection problems. In *Advances in Neural Information Processing Systems*, 2022.

[4] Sinho Chewi, Chen Lu, Kwangjun Ahn, Xiang Cheng, Thibaut Le Gouic, and Philippe Rigollet. Optimal dimension dependence of the Metropolis-adjusted Langevin algorithm. In *Conference on Learning Theory*, pages 1260–1300. PMLR, 2021.

[5] Radu V Craiu and Christiane Lemieux. Acceleration of the multiple-try Metropolis algorithm using antithetic and stratified sampling. *Statistics and computing*, 17:109–120, 2007.

[6] Philippe Gagnon, Florian Maire, and Giacomo Zanella. Improving multiple-try Metropolis with local balancing. *arXiv preprint arXiv:2211.11613*, 2022.

[7] Edward I George and Robert E McCulloch. Approaches for Bayesian variable selection. *Statistica Sinica*, pages 339–373, 1997.

[8] Mark Girolami and Ben Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[9] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[10] Jun S Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and computing*, 6:113–119, 1996.

[11] Jun S Liu, Faming Liang, and Wing Hung Wong. The multiple-try method and local optimization in Metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, 2000.

[12] Radford M Neal. MCMC using Hamiltonian dynamics. *arXiv preprint arXiv:1206.1901*, 2012.

[13] Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.

[14] Michael Smith and Robert Kohn. Nonparametric regression using Bayesian variable selection. *Journal of Econometrics*, 75(2):317–343, 1996.

[15] Rong Tang and Yun Yang. On the computational complexity of Metropolis-adjusted Langevin algorithms for Bayesian posterior sampling. *Journal of Machine Learning Research*, 25(157):1–79, 2024.

[16] Guanyang Wang. Exact convergence analysis of the independent Metropolis-Hastings algorithms. *Bernoulli*, 28(3):2012–2033, 2022.

[17] Keru Wu, Scott Schmidler, and Yuansi Chen. Minimax mixing time of the Metropolis-adjusted Langevin algorithm for log-concave sampling. *Journal of Machine Learning Research*, 23(270):1–63, 2022.

[18] Yun Yang, Martin J Wainwright, and Michael I Jordan. On the computational complexity of high-dimensional Bayesian variable selection. *The Annals of Statistics*, 44(6):2497–2532, 2016.

[19] Quan Zhou and Yongtao Guan. Fast model-fitting of Bayesian variable selection regression using the iterative complex factorization algorithm. *Bayesian Analysis*, 14(2):573, 2019.