Indian Institute of Technology Kanpur

Department of Mathematics and Statistics

Multiple Try Bernoulli Factory MCMC

Project Report for MTH393

Author:
Dwija Kakkad

Supervisor:
Prof. Dootika Vats

May 26, 2024

Abstract

Most of the standard MCMC algorithms require explicit evaluations of acceptance ratios. When the target distribution is intractable, these methods cannot be used. One way to overcome this is to use Bernoulli factories. However, these require an upper bound on the target distribution and become inefficient when a tight bound is not available. Multiple Try Metropolis is a commonly used MCMC method which involves proposing multiple values at each step and accepting the optimal proposal in terms of a specific weight function. We combine Multiple Try Metropolis with Bernoulli factories to give a novel method to sample from intractable posteriors.

Contents

1 Introduction

Bayesian statistics deals with inference based on the Bayes' theorem. Bayesian inference involves combining a prior hypothesis with the likelihood of observing the data given the prior hypothesis to obtain a posterior distribution on the parameters of interest. Unobserved quantities of interest can then be computed as expectations under this posterior distribution. Let $\pi(x)$ denote the posterior distribution,

$$\pi(x) \propto p(\theta)L(x|\theta)$$

where p is the prior density and L is the likelihood. Let $h(x)$ be the quantity of interest. We want to estimate

$$\theta_\pi = \int_{\mathcal{X}} h(x)\pi(x)dx.$$

The standard Monte-Carlo estimator for this quantity is

$$\hat{\theta}_\pi = \frac{\sum_{i=1}^{n} h(X_i)}{n} \text{ where } X_i \overset{iid}{\sim} \pi \text{ for } i = 1, \dots n$$

Sometimes, i.i.d. sampling is not possible from the posterior distribution. In these cases, one may use various methods such as Importance Sampling or Markov Chain Monte Carlo (MCMC). MCMC algorithms involve construction of stationary Markov Chains which have the target distribution as their equilibrium distribution. By using a Markov Chain of sufficient length, we can obtain samples from the target distribution and use these to estimate any quantity of interest (Brooks et al., 2011). Various MCMC algorithms have been discussed in the literature, the most popular being the Metropolis Hastings (MH) algorithm.

Given a target distribution π defined on a support $\mathcal{X}$, the Metropolis Hastings algorithm (Metropolis et al., 1953; Hastings, 1970) proposes a method to construct an ergodic Markov Chain which is stationary with respect to π. We need to choose an appropriate starting value X_0 and a proposal distribution, say q. The MH update is then given by,

The acceptance ratio α_{MH}, known as the Metropolis Hastings ratio requires π and q to be explicitly available to us. Sometimes, we do not have access to the posterior

Algorithm 1 Metropolis Hastings

1: **Input**: Current state x_t
2: **Output**: Next state x_{t+1}
3: Draw $y \sim q(\cdot|x_t)$
4: Accept y with probability $\alpha_{MH}(x, y) = \min\left(1, \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}\right)$
5: If y is rejected, set $x_{t+1} = x_t$

distribution π. This may be due to intractable priors or intractable likelihoods. It may be the case that the posterior is given in the form of an intractable integral, i.e. the posterior is of the form

$$\pi(x) = \int g(x, \eta) d\eta.$$

It also may be the case that the proposal distribution is intractable. In these cases, where we cannot explicitly evaluate the acceptance ratios, we can use Bernoulli factories to accept proposals with a particular ratio (Gonçalves et al., 2017; Vats et al., 2021; Huber, 2016). We use the Barker's acceptance ratio

$$\alpha_B(x, y) = \frac{\pi(y)q(y|x)}{\pi(y)q(y|x) + \pi(x)q(x|y)},$$

instead of α_{MH}. Barker's algorithm, which uses this ratio to accept a proposal, is less commonly used than the MH algorithm due to Peskun's result (Peskun and Peskun, 1973), which establishes that MH is always more efficient than Barkers's algorithm. However, it has been shown that Barker's algorithm is not too worse than the MH algorithm in terms of their asymptotic variance. (Gonçalves et al., 2017)
Multiple Try Metropolis algorithms involve proposing multiple values at each step, and comparing them by certain weights as shown in Liu et al. (2000). This allows exploration of a larger portion of the parameter space and decreases the correlation among the states (Robert et al., 2018) of the generated Markov chain, especially in the case of higher dimensions or multi-modal target distributions (Martino, 2018). The method is computationally demanding, but this can be improved if one evaluates the k weights of the proposals and then the $k-1$ weights of the auxiliary variables using parallel computing as discussed in Laloy and Vrugt (2012). In this report, we extend the two-coin algorithm for multiple proposals and propose a Multiple Try

algorithm using Bernoulli factories to accept or reject a proposal, using a modified acceptance ratio.
Section 2 discusses basic Markov chain theory, Section 3 discusses the two-coin algorithm, a portkey version of the two-coin algorithm and its extension for multiple proposals. Section 4 discusses the standard Multiple Try Metropolis algorithm. Section 5 contains the proposed algorithm using Bernoulli factories in Multiple Try Metropolis. Section 6 contains the results from implementation of three versions of MTM with Bernoulli factories on a Gamma mixture of Weibulls.

2 Markov Chain Theory

Let $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$ be a measurable space. We will assume $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{B}(\mathcal{X})$ is the Borel $\sigma-$field of $\mathcal{X}$.

Definition 1. *Markov Chain: A Markov chain is a discrete time stochastic process denoted by a sequence of random variables $X_1, X_2, \dots$ taking values in an arbitrary state-space. The sequence $\{X_1, X_2, \dots\}$ is such that the distribution of X_{n+1} given $X_1, \dots X_n$ is the same as the distribution of X_{n+1} given X_n.*

Definition 2. *Stationary Markov Chain: A Markov chain is said to be stationary if the conditional distribution of X_{n+1} given X_n is the same for all n.*

Definition 3. *Transition Probability: For a general state space S, the transition probabilities are specified by defining a Markov transition kernel. A Markov transition kernel is a map $P : \mathcal{X} \times \mathcal{B}(\mathcal{X}) \to [0, 1]$ such that, for all $A \in \mathcal{B}(\mathcal{X})$, $P(\cdot, A)$ is a measurable function on $\mathcal{X}$ and for all $x \in \mathcal{X}$, $P(x, \cdot)$ is a probability measure on $\mathcal{B}(\mathcal{X})$.*

Definition 4. *$\pi-$reversibility: Let π be a probability density. The Markov transition kernel P is said to be reversible with respect to π if*

$$\int \pi(x)P(x, B) = \int_B \pi(dx) = \pi(B),$$

i.e. $\pi P = \pi$. This implies that π is invariant for P. An equivalent condition for π reversibility is the detailed balanced condition. We can show that if a Markov chain

satisfies the detailed balance condition given by

$$\pi(x)A(x,y) = \pi(y)A(y,x)$$

where $A(x,y)$ is the probability of moving from x to y, the Markov chain is $\pi-$ reversible.

It can be shown that the Metropolis Hastings algorithm is π reversible, (Roberts and Rosenthal, 2004; Robert and Casella, 2000). Giving a similar proof,

Proof. We need to show that $\pi(x)A(x,y) = \pi(y)A(y,x)$, where $A(x,y)$ is as defined above.

$$\begin{aligned}
\pi(x)A(x,y) &= \pi(x) \times q(y|x) \times \alpha_{MH}(x,y) \\
&= \pi(x)q(y|x) \times \min\left(1, \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}\right) \\
&= \min(\pi(y)q(y|x), \pi(x)q(x|y)),
\end{aligned}$$

which is symmetric with respect to x and y. So, the Metropolis Hastings transition kernel satisfies detailed balance. ■

3 Bernoulli Factories

In the Metropolis Hastings algorithm and the Barker's algorithm, when the acceptance ratio is explicitly available to us, a $\mathcal{U}(0,1)$ draw is used to accept a proposal. However, we cannot use this when the ratio is intractable. In these cases, we use a Bern(α_B) draw and accept the proposal if we draw 1.

3.1 The two-coin algorithm

The two-coin algorithm simulates events with probability $\alpha_B(x,y)$. Let $\pi(x)q(x|y) = c_x p_x$ and $\pi(y)q(y|x) = c_y p_y$. We re-write $\alpha_B(x,y)$ as

$$\alpha_B(x,y) = \frac{c_y p_y}{c_x p_x + c_y p_y}.$$

One way to arrive at c_x and p_x is to find a c_x satisfying

$$\pi(x)q(x|y) \leq c_x, \text{ and setting } p_x = \pi(x)q(x|y)c_x^{-1}.$$

We use the same method to evaluate c_y and p_y. The two coin algorithm (Algorithm 2) then outputs 1 with probability proportional to $\alpha_B(x, y)$. The efficiency of the algorithm depends on the value of c. Note that a uniform bound over the entire support is not required, c_x and c_y can be local bounds. As given in Gonçalves et al. (2017), the number of loops needed until the algorithm stops is distributed as $\text{Geom}((c_x p_x + c_y p_y)/(c_x + c_y))$, and so the expected number of loops is

$$\frac{c_x + c_y}{c_x p_x + c_y p_y} = \frac{c_x + c_y}{\pi(x)q(y|x) + \pi(y)q(x|y)}$$

Note that the mean execution time of the algorithm depends on the number of loops, which is directly proportional to the upper bounds c_x and c_y. If the bounds are loose, then the algorithm will be less efficient.

Algorithm 2 Two coin algorithm

1: Draw $C_1 \sim \text{Bern}\left(\frac{c_y}{c_x+c_y}\right)$
2: **if** $C_1 = 1$ **then**
3: Draw $C_2 \sim \text{Bern}(p_y)$
4: **if** $C_2 = 1$ **then**
5: output 1
6: **else**
7: go to Step 1
8: **end if**
9: **end if**
10: **if** $C_1 = 0$ **then**
11: Draw $C_2 \sim \text{Bern}(p_x)$
12: **if** $C_2 = 1$ **then**
13: output 0
14: **else**
15: go to Step 1
16: **end if**
17: **end if**

Note that we do not have the exact value of p_x. To generate a Bern(p_x) draw, we need an unbiased estimator of p_x, say $\hat{p}_x$ and draw from Bern($\hat{p}_x$). We can use a Monte-Carlo estimator or an Importance Sampling estimator to do this. For eg, if the target is given to us as $\int g(\eta, x)d\eta$, we choose a proposal distribution having density h. We then draw M from this proposal, and draw a Bernoulli random variable

$$C|M \sim \text{Bern}\left(\frac{g(M,x)}{c_x h(M)}\right).$$

Proposition 1. *This Bernoulli draw gives us 1 with probability p_x*

Proof. We need to show that $E_B(C) = p_x$, where B denotes the Bernoulli distribution.

$$\begin{aligned}
E_B(C) &= E_H(E_B(C|M)) \\
&= E_H\left(\frac{g(M,x)}{c_x h(M)}\right) \\
&= \frac{\int g(M,x)}{c_x} dM \\
&= \frac{\pi(x)}{c_x} \\
&= p_x.
\end{aligned}$$

■

3.2 The Portkey method

When the bound on $\pi(x)$ is not very tight, we get a small value of p_x which leads to a large number of loops in a Bernoulli factory, which decreases the efficiency of the two-coin algorithm. The Portkey method (Vats et al., 2021) fixes this problem by effectively providing an upper bound on the number of loops. The algorithm introduces a Bernoulli draw at the beginning of each outer loop which allows a rejection with probability $1 - \beta$. If we set β close to 1, we can avoid a large number of loops when the value of p_x or p_y is very low. Setting $\beta = 1$ gives us the original two-coin algorithm. As given in Vats et al. (2021), the number of loops needed until

the algorithm stops is distributed as $\text{Geom}(s_b)$, where

$$s_b = (1-\beta) + \beta \cdot \frac{c_y p_y + c_x p_x}{c_y + c_x}$$

So the expected number of loops of this algorithm will be

$$\left((1-\beta) + \beta \cdot \frac{c_y p_y + c_x p_x}{c_y + c_x}\right)^{-1}$$

We typically choose β to be close to 1. So, $(1-\beta)$ will be close to 0. We can see that the expected number of loops of the Portkey algorithm (Algorithm 3) is less than that of the two-coin algorithm (Algorithm 2).

Algorithm 3 Portkey algorithm

Draw $S \sim \text{Bern}(\beta)$
if $S = 0$ **then**
    output 0
else
    Draw $C_1 \sim \text{Bern}\left(\frac{c_y}{c_x + c_y}\right)$
    if $C_1 = 1$ **then**
        Draw $C_2 \sim \text{Bern}(p_y)$
        if $C_2 = 1$ **then**
            output 1
        else
            go to Step 1
        end if
    end if
    if $C_1 = 0$ **then**
        Draw $C_2 \sim \text{Bern}(p_x)$
        if $C_2 = 1$ **then**
            output 0
        else
            go to Step 1
        end if
    end if
end if

Proposition 2. *The Portkey algorithm accepts a proposal with an acceptance ratio of*

$$r_p = \frac{\pi(y)}{\pi(x) + \pi(y) + (\beta^{-1} - 1)(c_x + c_y)}.$$

Here, β is as defined in 3.

Proof. The paper Vats et al. (2021) gives us a proof of this. Here, we provide an alternate proof for the same proposition. Let Y be the random variable indicating if we accept the proposal or not. i.e., we accept the proposal if $Y = 1$ and reject otherwise.

$$\begin{aligned}
\Pr(Y = 1) &= \Pr(S = 1)[\Pr(C1 = 1, C2 = 1) \\
&\quad + \Pr(C1 = 1, C2 = 0) \Pr(Y = 1) \\
&\quad + \Pr(C1 = 0, C2 = 0) \Pr(Y = 1)] \\
&= \beta \left[\frac{c_y p_y}{c_y + c_x} + Pr(Y = 1) \left(\frac{c_y(1 - p_y) + c_x(1 - p_x)}{c_x + c_y} \right) \right] \\
&= \frac{\beta c_y p_y}{c_x + c_y - \beta(c_x + c_y - c_x p_x - c_y p_y)} \\
&= \frac{\pi(y)}{\pi(x) + \pi(y) + (\beta^{-1} - 1)(c_x + c_y)}
\end{aligned}$$

■

4 Multiple Try Metropolis

In the standard Metropolis Hastings algorithm for a target π, we draw from a proposal distribution and accept it with probability α_{MH}. Let T be the Markov transition kernel for the MH algorithm. The multiple try Metropolis algorithm (MTM) enables us to propose multiple points from $T(\mathbf{x}, \cdot)$ at each step. In Liu et al. (2000), a weight function $w(\mathbf{x,y})$ is defined as:

$$w(\mathbf{x,y}) = \pi(\mathbf{x}) T(\mathbf{x,y}) \lambda(\mathbf{x,y}),$$

where π is the target distribution, T is the Markov transition kernel and λ is a non-negative symmetric function in $\mathbf{x}$ and $\mathbf{y}$. The only condition to be satisfied by λ is that $\lambda(\mathbf{x},\mathbf{y}) > 0$ whenever $T(\mathbf{x},\mathbf{y}) > 0$.

Algorithm 4 Multiple try Metropolis

1: **Input:** current state $\mathbf{x}_t = \mathbf{x}$
2: **Output:** new state $\mathbf{x}_{t+1}$
3: Draw k i.i.d. proposals $\mathbf{y}_1, \ldots \mathbf{y}_k$ from $T(\mathbf{x}, \cdot)$.
4: Compute $w(\mathbf{y}_j, x)$ for $j = i, \ldots k$.
5: Select $\mathbf{Y} = \mathbf{y}$ among the set $\{\mathbf{y}_1 \ldots \mathbf{y}_k\}$ with probability proportional to $w(\mathbf{y}_j, \mathbf{x}), j = i, \ldots k$.
6: Draw $\mathbf{x}_1^*, \ldots \mathbf{x}_{k-1}^* \sim T(\mathbf{y}, \cdot)$ and let $\mathbf{x}_k^* = \mathbf{x}$.
7: Let

$$\alpha_{MTM} = \min\left\{1, \frac{w(\mathbf{y}_1, \mathbf{x}) + \cdots + w(\mathbf{y}_k, \mathbf{x})}{w(\mathbf{x}_1^*, \mathbf{y}) + \cdots + w(\mathbf{x}_k^*, \mathbf{y})}\right\}$$

8: $u \sim \mathcal{U}(0, 1)$
9: **if** $u \leq \alpha_{MTM}$ **then**
10: $\quad \mathbf{x}_{t+1} = \mathbf{y}$
11: **else**
12: $\quad \mathbf{x}_{t+1} = \mathbf{x}$
13: **end if**

The above MTM algorithm gives us a $\pi-$invariant Markov chain. A proof of the same is given in Liu et al. (2000). Note that the k trial proposals $\mathbf{y}_1, \ldots, \mathbf{y}_k$ need not be independent draws. i.e. if the transition distribution depends on some other random variable $\mathbf{e}$, then we can generate $\mathbf{e}$ once and then using that we can generate $\mathbf{y}_1 \cdots \mathbf{y}_k$.

Proposition 3. *When $k = 1$, α_{MTM} becomes the Metropolis-Hastings acceptance ratio and the MTM algorithm becomes the regular MH algorithm.*

Proof. When $k = 1$, we do not have to draw $\mathbf{x}_1^*, \ldots, \mathbf{x}_{k-1}^*$ and $\mathbf{x}_k = \mathbf{x}$. So,

$$\begin{aligned}
\alpha_{MTM} &= \min\left\{1, \frac{w(\mathbf{y}, \mathbf{x})}{w(\mathbf{x}, \mathbf{y})}\right\} \\
&= \min\left\{1, \frac{\pi(\mathbf{y})T(\mathbf{y}, \mathbf{x})}{\pi(\mathbf{x})T(\mathbf{x}, \mathbf{y})}\right\} \qquad \text{as } \lambda \text{ is symmetric for } \mathbf{x} \text{ and } \mathbf{y} \\
&= \alpha_{MH}(\mathbf{x}, \mathbf{y})
\end{aligned}$$

■

Proposition 4. *If we choose* $\lambda(\boldsymbol{x}, \boldsymbol{y}) = \{T(\boldsymbol{x}, \boldsymbol{y})T(\boldsymbol{y}, \boldsymbol{x})\}^{-1}$, $\alpha_{MH} \to 1$ *and the MTM sampler converges to a Gibbs sampler as* $k \to \infty$, .

Proof. The proposals $\mathbf{y}_1 \ldots \mathbf{y}_k$ are drawn from $T(\mathbf{x}, \cdot)$., and a new $\mathbf{y}$ is chosen from this distribution, i.e. $\mathbf{y} \sim T(\mathbf{x}, \cdot)$. The kernel $T(\mathbf{x}, \cdot)$ is essentially the proposal density conditioned on $\mathbf{x}$, so we can rewrite $T(\mathbf{x}, \cdot)$ as $q(\cdot|\mathbf{x})$, where q is the proposal density.
Taking $\lambda(\mathbf{x}, \mathbf{y}) = \{T(\mathbf{x}, \mathbf{y})T(\mathbf{y}, \mathbf{x})\}^{-1}$, the weights associated become equal to the importance sampling weights (as $w(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{x})T(\mathbf{x}, \mathbf{y})\lambda(\mathbf{x}, \mathbf{y})$)
By the Weak Law of Large Numbers,

$$\begin{aligned}
\lim_{k\to\infty} \frac{\sum_{i=1}^{k} w(\mathbf{y}_i, \mathbf{x})}{k} &= E_{q(\cdot|\mathbf{x})}(w(\mathbf{y}, \mathbf{x})) \\
&= \int_{A_x} \pi(\mathbf{y})q(\mathbf{x}|\mathbf{y})\lambda(\mathbf{x}, \mathbf{y})q(\mathbf{y}|\mathbf{x})d\mathbf{y} \\
&= \int_{A_x} \pi(\mathbf{y})d\mathbf{y}
\end{aligned}$$

where $\mathbf{y}$ is drawn from $q(\cdot|\mathbf{x})$ and $A_x = \{\mathbf{y} : T(\mathbf{x}, \mathbf{y}) > 0\}$

Also,

$$\begin{aligned}\lim_{k\to\infty}\frac{\sum_{i=1}^{k} w(\mathbf{x}_i^*,\mathbf{y})}{k} &= E_{q(\cdot|\mathbf{y})}(w(\mathbf{x*},\mathbf{y}) \\ &= \int_{B_x} \pi(\mathbf{x}^*)q(\mathbf{y}|\mathbf{x}^*)\lambda(\mathbf{y},\mathbf{x})q(\mathbf{x}^*|\mathbf{y})d\mathbf{x}^* \\ &= \int_{B_x} \pi(\mathbf{x}^*)d\mathbf{x}^*\end{aligned}$$

where $\mathbf{x}^*$ is drawn from $q(\cdot|\mathbf{y})$ and $B_x = \{\mathbf{x}^* : T(\mathbf{y},\mathbf{x}^*) > 0\}$. So,

$$\begin{aligned}\lim_{k\to\infty}\alpha_{MTM} &= \frac{\int_{A_x}\pi(\mathbf{y})d\mathbf{y}}{\int_{B_x}\pi(\mathbf{x}^*)d\mathbf{x}^*} \\ &= 1\end{aligned}$$

Since we are accepting $\mathbf{y}$ with a probability of 1 and $\mathbf{y}$ is a draw from $q(\cdot|\mathbf{x})$, this is a Gibbs sampling update. ■

We can change λ according to our target. Different choices of λ give us different variations of MTM.

1. When $\lambda(\mathbf{x},\mathbf{y}) \equiv 1$, the algorithm corresponds to the simple Multiple Try Metropolis algorithm(Liu et al., 2000).

2. If we take $\lambda(\mathbf{x},\mathbf{y}) = \{T(\mathbf{x},\mathbf{y})T(\mathbf{y},\mathbf{x})\}^{-1}$, the weights correspond to the importance weights, this is a popular choice for λ.

3. **Orientational Bias Monte Carlo:** If we take

$$\lambda = \left(\frac{T(\mathbf{x},\mathbf{y}) + T(\mathbf{y},\mathbf{x})}{2}\right)^{-1}$$

when T is symmetric, the algorithm becomes the Orientational Bias Monte Carlo scheme given in Frenkel and Smit (1996) and the function w becomes directly proportional to the target density π. So, in step 5, we pick a $\mathbf{y}$ with

probability proportional to $\pi(\mathbf{y}_j)$, and α_{MTM} in step 7 becomes:

$$\alpha_{MTM} = min\left\{1, \frac{\pi(\mathbf{y}_1) + \cdots + \pi(\mathbf{y}_k)}{\pi(\mathbf{x}_1^*) + \cdots + \pi(\mathbf{x}_k^*)}\right\}.$$

4. **Independent Multiple Try Metropolis:**The general MTM algorithm requires $2k-1$ draws from the transition distribution. If the proposal pdf is independent of the previous state, i.e.: $T(\mathbf{x}, \cdot)$ is equal to the distribution of $\mathbf{y}$ (if q is the proposal pdf, then $q(y|x) = q(y)$), we don't need to sample the $k-1$ draws in step 6. Let $\mathbf{y}_j$ be the selected draw in step 5. In step 6, we simply set $\mathbf{x}_1^* = \mathbf{y}_1 \ldots \mathbf{x}_{j-1}^* = \mathbf{y}_{j-1}, \mathbf{x}_j^* = \mathbf{y}_{j+1} \ldots \mathbf{x}_{k-1}^* = \mathbf{y}_k, \mathbf{x}_k^* = \mathbf{x}$. The chain will still remain ergodic and we can rewrite α_{MTM} accordingly. (Martino, 2018)Another method to avoid $2k-1$ draws is to reuse the samples in the previous iteration.

5 Multiple Try Metropolis using Bernoulli Factories

Let $\pi(\mathbf{x})$ be the target we want to sample from. To use a Multiple Try Metropolis algorithm to sample from π, we use the acceptance ratio

$$\alpha_{MTM} = \min\left\{1, \frac{w(\mathbf{y}, \mathbf{x}) + \sum_{j=1}^{k-1} w(\mathbf{y}_j, \mathbf{x})}{w(\mathbf{x}, \mathbf{y}) + \sum_{j=1}^{k-1} w(\mathbf{x}_j^*, \mathbf{y})}\right\},$$

where w, $\mathbf{y}_j$ and $\mathbf{x}_j^*$ are as defined in 4. When $\pi(\mathbf{x})$ is not available to us directly, it becomes difficult to use this acceptance ratio. We use a modified acceptance ratio

$$\alpha_m = \frac{w(\mathbf{y}, \mathbf{x}) + \sum_{j=1}^{k-1} w(\mathbf{y}_j, \mathbf{x})}{w(\mathbf{y}, \mathbf{x}) + \sum_{j=1}^{k-1} w(\mathbf{y}_j, \mathbf{x}) + w(\mathbf{x}, \mathbf{y}) + \sum_{j=1}^{k-1} w(\mathbf{x}_j^*, \mathbf{y})}$$

and use Bernoulli factories to accept or reject a proposal.

Theorem 1. *Any acceptance ratio $s(t)$ satisfying the condition: $s(t) = ts(1/t)$ yields*

a $\pi-$ invariant Markov chain, where

$$t = \frac{w(\boldsymbol{y}, \boldsymbol{x}) + \sum_{j=1}^{k-1} w(\boldsymbol{y}_j, \boldsymbol{x})}{w(\boldsymbol{x}, \boldsymbol{y}) + \sum_{j=1}^{k-1} w(\boldsymbol{x}_j^*, \boldsymbol{y})}$$

Proof. Let q be the proposal density. We need to prove that the Markov chain satisfies detailed balance, i.e. $\pi(\mathbf{x})A(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{y})A(\mathbf{x}, \mathbf{y})$, where $A(\mathbf{x}, \mathbf{y})$ is the transition probability of moving from $\mathbf{x}$ to $\mathbf{y}$. Let I indicate which of $\mathbf{y}_j$ is chosen. Since the acts of choosing $\mathbf{y}_j$ given $\mathbf{x}$ are disjoint,

$$\begin{aligned}
\pi(\mathbf{x})A(\mathbf{x}, \mathbf{y}) &= \pi(\mathbf{x})P\left[\bigcup_{j=1}^{k}\{(\mathbf{y}_j = \mathbf{y})\bigcap(I = j)\}|\mathbf{x}\right] \\
&= k\pi(\mathbf{x})P[(\mathbf{y}_k = \mathbf{y}) \cap (I = k)|\mathbf{x}] \\
&= k\pi(\mathbf{x}) \int \cdots \int q(\mathbf{y}|\mathbf{x})q(\mathbf{y}_1|\mathbf{x}) \ldots q(\mathbf{y}_{k-1}|\mathbf{x}) \\
&\times \frac{w(\mathbf{y}, \mathbf{x})}{w(\mathbf{y}, \mathbf{x}) + \sum_{j=1}^{k-1} w(\mathbf{y}_j, \mathbf{x})} \times s(t) \\
&\times q(\mathbf{x}_1^*|\mathbf{y}) \ldots q(\mathbf{x}_{k-1}^*|\mathbf{y}) d\mathbf{y}_1 \ldots d\mathbf{y}_{k-1} d\mathbf{x}_1^* \ldots d\mathbf{x}_{k-1}^* \\
&= k\pi(\mathbf{x}) \int \cdots \int q(\mathbf{y}|\mathbf{x})q(\mathbf{y}_1|\mathbf{x}) \ldots q(\mathbf{y}_{k-1}|\mathbf{x}) \\
&\times \frac{w(\mathbf{y}, \mathbf{x})}{w(\mathbf{x}, \mathbf{y}) + \sum_{j=1}^{k-1} w(\mathbf{x}_j^*, \mathbf{y})} \times s(\frac{1}{t}) \\
&\times q(\mathbf{x}_1^*|\mathbf{y}) \ldots q(\mathbf{x}_{k-1}^*|\mathbf{y}) d\mathbf{y}_1 \ldots d\mathbf{y}_{k-1} d\mathbf{x}_1^* \ldots d\mathbf{x}_{k-1}^*
\end{aligned}$$

The term inside the integral is symmetric with respect to $\mathbf{x}$ and $\mathbf{y}$. This holds because

$$\frac{1}{W_y} \times s\left(\frac{W_y}{W_x}\right) = \frac{1}{W_x} \times s\left(\frac{W_x}{W_y}\right)$$

where

$$W_y = w(\mathbf{y}, \mathbf{x}) + \sum_{j=1}^{k-1} w(\mathbf{y}_j, \mathbf{x})$$

and

$$W_x = w(\mathbf{x}, \mathbf{y}) + \sum_{j=1}^{k-1} w(\mathbf{x}_j^*, \mathbf{x}).$$

For the terms outside,

$$k\pi(\mathbf{x})q(\mathbf{y}|\mathbf{x})w(\mathbf{y}, \mathbf{x}) = k\pi(\mathbf{x})\pi(\mathbf{y})\lambda(\mathbf{x}, \mathbf{y})$$

Since λ is symmetric, this term is also symmetric with respect to $\mathbf{x}$ and $\mathbf{y}$. And so, the detailed balanced condition holds. ■

5.1 Bernoulli factories for multiple proposals

The Multiple Try Metropolis algorithm involves a step where we choose a proposal out of k proposals with probability proportional to the weight function. If we cannot evaluate the posterior density, then we cannot explicitly evaluate the weights as well. So we use another Bernoulli factory to sample a proposal with probability proportional to w.
Let $c_{y_1}, \dots c_{y_j} \dots c_{y_k}$ be the upper bounds for $w(y_1, x), \dots w(y_k, x)$ respectively and S be $\sum_{j-1}^{k} c_{y_j}$. Then, $p_{y_j} = w(y_j, x)c_{y_j}^{-1}$

Proposition 5. *Algorithm 5 chooses a proposal y_j with probability proportional to $w(y_j, x)$.*

Proof. The probability that the random variable Y takes the value y_j,

$$\begin{aligned}
\Pr(Y = y_j) &= \Pr(C1 = j)\Pr(C2 = 1|C1 = j) + \sum_{i=1}^{k} \Pr(C1 = i)\Pr(C2 = 0|C1 = i)\Pr(Y = y_j) \\
&= \frac{c_{y_j} p_{y_j}}{S - \sum_{i=1}^{k} c_{y_i}(1 - p_{y_i})} \\
&= \frac{w(y_j, x)}{\sum_{i=1}^{k} c_{y_i} p_{y_i}} \\
&= \frac{w(y_j, x)}{\sum_{i=1}^{k} w(y_i, x)}
\end{aligned}$$

■

Algorithm 5 Bernoulli Factory for multiple proposals

Draw $C_1 \sim \text{Categorical}\left(\frac{c_{y_1}}{S}, \dots \frac{c_{y_k}}{S}\right)$
if $C_1 = 1$ **then**
    Draw $C_2 \sim \text{Bern}(p_{y_1})$
    if $C_2 = 1$ **then**
        output 1
    else
        go to Step 1
    end if
    ...
else if $C_1 = j$ **then**
    Draw $C_2 \sim \text{Bern}(p_{y_j})$
    if $C_2 = 1$ **then**
        output j
    else
        go to Step 1
    end if
    ...
else if $C_1 = k$ **then**
    Draw $C_2 \sim \text{Bern}(p_{y_k})$
    if $C_2 = 1$ **then**
        output k
    else
        go to Step 1
    end if
end if

We can calculate the expected efficiency of this algorithm as well. Clearly, the number of loops needed until the algorithm stops follows a geometric distribution. The probability that the algorithm stops is given by

$$\begin{aligned} p &= \frac{c_{y_1}}{S} p_{y_1} + \ldots \frac{c_{y_k}}{S} p_{y_k} \\ &= \frac{\sum_{i=1}^{k} c_{y_i} p_{y_i}}{\sum_{i=1}^{k} c_{y_i}} \end{aligned}$$

So, the number of loops till the algorithm stops follows a Geom(p) distribution, where p is as given above and the expected number of loops is then

$$p^{-1} = \left(\frac{\sum_{i=1}^{k} c_{y_i} p_{y_i}}{\sum_{i=1}^{k} c_{y_i}} \right)^{-1}$$

5.2 Two-coin algorithm using Telescopic Product

When we use the two-coin algorithm for Multiple Try Metropolis, the bound we get on W_y can be quite loose as W_y itself is a sum of weight functions. i.e., the acceptance ratio for the MTM algorithm is

$$\frac{\sum_{j=1}^{k} w(\mathbf{y}_j, \mathbf{x})}{\sum_{j=1}^{k} w(\mathbf{y}_j, \mathbf{x}) + \sum_{j=1}^{k} w(\mathbf{x}_j^*, \mathbf{y})}.$$

Let c_{y_i} be a bound on $w(\mathbf{y}_i, \mathbf{x})$. Then, we can take the bound on W_y to be

$$C_Y = k * \max_{i \in 1, \ldots k} \{c_{y_i}\}.$$

Similarly let $c_{x_i^*}$ be the bound on $w(\mathbf{x}_i^*, \mathbf{y})$. The bound on W_x will become

$$C_X = k * \max_{i \in 1, \ldots k} \{c_{x_i^*}\}.$$

In this case, even if any one of the bounds on the weights is extremely large, the algorithm becomes extremely slow as the number of loops increases drastically. For a reasonable number of proposals, the problem is often that out of all the proposed

values of y, the bound on the weight of one of them is quite large as compared to the other bounds. To fix such cases, we can use the following modification to the two-coin algorithm. Note that our goal is to accept $\mathbf{y}$ with probability $\frac{W_y}{W_y+W_x}$ Once we get the bounds $c_{x_i^*}$, we arrange the bounds in a non-decreasing order. So, C_X becomes $k * c_{x_k^*}$. We then rewrite the acceptance function as

$$\alpha_{MTM} = \frac{W_y}{W_y + \sum_{i=1}^{k-1} w(\mathbf{x}_i^*, \mathbf{y})} \times \frac{W_y + \sum_{i=1}^{k-1} w(\mathbf{x}_i^*, \mathbf{y})}{W_x + W_y}$$

Let

$$A_1 = \frac{W_y}{W_y + \sum_{i=1}^{k-1} w(\mathbf{x}_i^*, \mathbf{y})}$$

and

$$A_2 = \frac{W_y + \sum_{i=1}^{k-1} w(\mathbf{x}_i^*, \mathbf{y})}{W_x + W_y}.$$

We can use the standard two-coin algorithm to simulate Bernoulli coins with parameters A_1 and A_2 (in steps 1 and 5 of Algorithm 6). The main advantage of this method is that in some cases we will not require the largest bound on $w(\mathbf{x}_i^*, \mathbf{y})$. Another improvement to this would be to compare C_y and C_x. If $C_x > C_y$, we will proceed exactly as shown above and if $C_y > C_x$, we will then use the ratio

$$r'_g = \frac{W_x}{W_y + W_x}$$

to generate a Bernoulli coin and reject the proposal with that probability. Note that we make use of the fact that $C1$ and $C2$ are independent, and that even if one of them is 0, the two-coin algorithm outputs 0.

Proposition 6. *The expected number of loops in Algorithm 6 is*

Proof. Let BF_1 denote the first Bernoulli factory, i.e. the one where we draw Bern(A_1) and let BF_2 denote the second Bernoulli factory, the one where we draw Bern(A_2). The expected number of loops in Algorithm 6 can be given as the sum of the expected number of loops in BF_1, say E_1 and the expected number of loops of

Algorithm 6 Bernoulli Factory using telescoping product

1: $C_1 \sim Bern(A_1)$
2: **if** $C_1 = 0$ **then**
3: return 0
4: **else if** $C_1 = 1$ **then**
5: $C_2 \sim Bern(A_2)$
6: return C_2
7: **end if**

BF_2, say E_2 given $C1 = 1$. That is,

$$\begin{aligned} E_l &= E_1 + E_2 \cdot \Pr(C1 = 1) \\ &= E_1 + E_2 \cdot A_1 \end{aligned}$$

Using the same notation as above, let C_X and C_Y be the bounds on W_x and W_y respectively, where C_X and C_Y are as defined above. Let c_{max} denote the largest bound among $c_{x_1^*}, \ldots c_{x_k^*}$. We can calculate E_1 and E_2 using the formula mentioned in section 3. So, we get

$$E_1 = \frac{C_Y + (k-1) * c_{max}}{W_y + \sum_{i=1}^{k-1} w(x_i^*, y)}$$

and

$$E_2 = \frac{2 * \max\{C_Y, (k-1) * c_{max}\} + c_{max}}{W_x + W_y}.$$

The expected number of loops then becomes

$$E_l = \frac{C_Y + (k-1) * c_{max}}{W_y + \sum_{i=1}^{k-1} w(x_i^*, y)} + \frac{2 * \max\{C_Y, (k-1) * c_{max}\} + c_{max}}{W_x + W_y} \cdot \frac{W_y}{W_y + \sum_{i=1}^{k-1} w(x_i^*, y)}$$

■

5.3 Portkey Algorithm for Multiple Try Metropolis

In Multiple Try Metropolis, we use the generalized acceptance ratio

$$\frac{W_y}{W_x + W_y}$$

where W_x and W_y are as defined in 5. We can use the Portkey algorithm to accept or reject the proposed trial, modifying the acceptance rate to

$$\frac{W_y}{W_y + W_x + (\beta^{-1} - 1)(c_x + c_y)}.$$

Here, c_x and c_y are bounds on W_x and W_y respectively.

Theorem 2. *The Markov chain obtained using Multiple Try Metropolis and the Portkey algorithm is π-invariant.*

Proof. We proved the Markov chain to be invariant when the acceptance probability is of the form $s(t)$ where $s(t) = ts(1/t)2$. Here, the acceptance ratio for the Portkey algorithm does not satisfy that condition and so, a separate proof is required.
Let $a(\mathbf{x}, \mathbf{y}) = (\beta^{-1} - 1)(c_x + c_y)$. Observe that $a(\mathbf{x}, \mathbf{y})$ is symmetric with respect to

$\mathbf{x}$ and $\mathbf{y}$.

$$\begin{aligned}
\pi(\mathbf{x})A(\mathbf{x},\mathbf{y}) &= \pi(\mathbf{x})P\left[\bigcup_{j=1}^{k}\{(\mathbf{y}_j=\mathbf{y})\bigcap(I=j)\}|\mathbf{x}\right] \\
&= k\pi(\mathbf{x})P[(\mathbf{y}_k=\mathbf{y})\cap(I=k)|\mathbf{x}] \\
&= k\pi(\mathbf{x})\int\cdots\int q(\mathbf{y}|\mathbf{x})q(\mathbf{y}_1|\mathbf{x})\ldots q(\mathbf{y}_{k-1}|\mathbf{x}) \\
&\times \frac{w(\mathbf{y},\mathbf{x})}{w(\mathbf{y},\mathbf{x})+\sum_{j=1}^{k-1}w(\mathbf{y}_j,\mathbf{x})}\times r_g \\
&\times q(\mathbf{x}_1^*|\mathbf{y})\ldots q(\mathbf{x}_{k-1}^*|\mathbf{y})d\mathbf{y}_1\ldots d\mathbf{y}_{k-1}d\mathbf{x}_1^*\ldots d\mathbf{x}_{k-1}^* \\
&= k\pi(\mathbf{x})q(\mathbf{y}|\mathbf{x})w(\mathbf{y},\mathbf{x})\int\cdots\int q(\mathbf{y}_1|\mathbf{x})\ldots q(\mathbf{y}_{k-1}|\mathbf{x}) \\
&\times \frac{1}{W_y}\times\frac{W_y}{W_x+W_y+a(\mathbf{x},\mathbf{y})} \\
&\times q(\mathbf{x}_1^*|\mathbf{y})\ldots q(\mathbf{x}_{k-1}^*|\mathbf{y})d\mathbf{y}_1\ldots d\mathbf{y}_{k-1}d\mathbf{x}_1^*\ldots d\mathbf{x}_{k-1}^* \\
&= k\pi(\mathbf{x})\pi(\mathbf{y})\lambda(\mathbf{x},\mathbf{y})\int\cdots\int q(\mathbf{y}_1|\mathbf{x})\ldots q(\mathbf{y}_{k-1}|\mathbf{x}) \\
&\times \frac{1}{W_x+W_y+a(\mathbf{x},\mathbf{y})} \\
&\times q(\mathbf{x}_1^*|\mathbf{y})\ldots q(\mathbf{x}_{k-1}^*|\mathbf{y})d\mathbf{y}_1\ldots d\mathbf{y}_{k-1}d\mathbf{x}_1^*\ldots d\mathbf{x}_{k-1}^*
\end{aligned}$$

This is symmetric with respect to $\mathbf{x}$ and $\mathbf{y}$, and so the algorithm satisfies the detailed balance condition. ■

5.4 Multiple Try Bernoulli Factory MCMC

Finally, here is an algorithm which uses Multiple Try Bernoulli Factories to sample from target densities where the acceptance ratios are intractable. Recall that we often wish to sample from intractable target distributions. A specific case is when the target g is given as $g(x)=\int g(x,\eta)d\eta$. In these cases, we use Bernoulli factories to accept or reject a proposal. The normal two-coin algorithm used with Barker's algorithm is often inefficient if the bound on the target is loose. Using Multiple

Try Metropolis with Bernoulli factories can be more efficient as several proposals are available at each step and this may prevent the Markov chain from getting stuck at one state. Note that we use two Bernoulli factories in this algorithm, one to choose a proposal y depending on a weight function and one to accept or reject y with a certain probability.

Algorithm 7 Multiple Try Bernoulli Factory MCMC

1: **Input:** Current state $\mathbf{x}_t = \mathbf{x}$
2: **Output:** New state $\mathbf{x}_{t+1}$
3: Draw k i.i.d. proposals $\mathbf{y}_1, \dots \mathbf{y}_k$ from $q(\cdot|\mathbf{x})$
4: Using Algorithm 5 select $\mathbf{Y} = \mathbf{y}$ from $\{\mathbf{y}_1, \dots \mathbf{y}_k\}$ with probability proportional to $w(\mathbf{y}_j, \mathbf{x})$, where $j = \{1, \dots k\}$
5: Draw $\mathbf{x}_1^*, \dots \mathbf{x}_k^*$ from $q(\cdot|\mathbf{y})$, and set $\mathbf{x}_k^* = \mathbf{x}$.
6: Using Algorithm 2 or Algorithm 6 draw a Bernoulli coin C with parameter

$$\alpha_m = \frac{w(\mathbf{y}, \mathbf{x}) + \sum_{j=1}^{k-1} w(\mathbf{y}_j, \mathbf{x})}{w(\mathbf{y}, \mathbf{x}) + \sum_{j=1}^{k-1} w(\mathbf{y}_j, \mathbf{x}) + w(\mathbf{x}, \mathbf{y}) + \sum_{j=1}^{k-1} w(\mathbf{x}_j^*, \mathbf{y})}$$

7: Or using Algorithm 3 draw a Bernoulli coin C with parameter

$$\frac{w(\mathbf{y}, \mathbf{x}) + \sum_{j=1}^{k-1} w(\mathbf{y}_j, \mathbf{x})}{w(\mathbf{y}, \mathbf{x}) + \sum_{j=1}^{k-1} w(\mathbf{y}_j, \mathbf{x}) + w(\mathbf{x}, \mathbf{y}) + \sum_{j=1}^{k-1} w(\mathbf{x}_j^*, \mathbf{y}) + (\beta^{-1} - 1)(C_X + C_Y)},$$

where C_X and C_Y are as defined in 5.2.
8: **if** $C = 1$ **then**
9: **return** $\mathbf{x}_{t+1} = \mathbf{y}$
10: **else if** $C = 0$ **then**
11: **return** $\mathbf{x}_{t+1} = \mathbf{x}_t$
12: **end if**

Using the Bernoulli factory with telescopic product given in Algorithm 6 can be computationally more efficient as it stops as soon as one proposal is accepted. The Portkey version will significantly decrease the number of loops, but we cannot extend that to the Bernoulli factory with multiple proposals because it changes the acceptance ratio. Depending on the target given and the bounds available to us, we can choose which version of the Bernoulli factory to use.

6 Example

6.1 Gamma mixture of Weibulls

Consider a target π of a Gamma mixture of Weibulls, i.e.

$$\pi(x) = \int \pi(x|\lambda)\nu(\lambda)d\lambda,$$

where ν is the p.d.f. of a $\Gamma(10, 100)$ distribution. $\pi(x|\lambda)$ is the p.d.f. of a Weibull distribution having a fixed shape parameter k and scale λ.We use Multiple Try MCMC using Bernoulli factories to sample from this. Let m be the number of proposals. Averaged results from 20 replications are given, implementing the following three versions of Multiple Try Bernoulli Factory MCMC:

1. Using the categorical Bernoulli factory for choosing y and the normal two-coin algorithm for accepting y.

2. Using the categorical Bernoulli factory for choosing y and the two-coin algorithm with telescopic product for accepting y.

3. Using the categorical Bernoulli factory for choosing y and the Portkey algorithm for accepting y.

The Markov chain is of length $1e4$ for all algorithms.

m	5	10	100
Avg loops categorical	15310.71	117.2706	47.30897
Avg loops bernoulli	190.6966	457.8088	3838.265
Max loops categorical	3035623327	13675089	411648
Max loops bernoulli	2781672	21374296	54869132

Table 1: Categorical Bernoulli and the normal two-coin algorithm

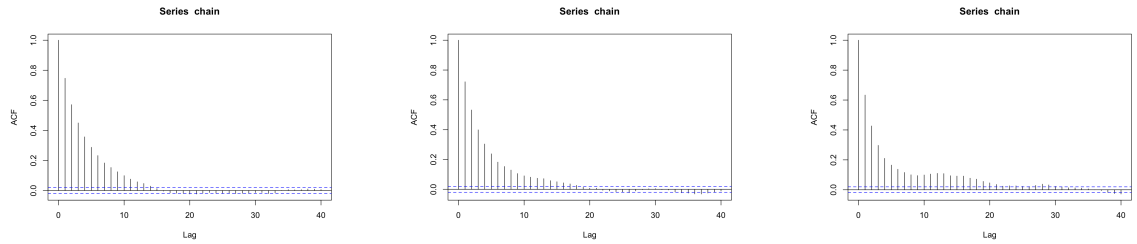
We compare the ACF plots of the Markov chains obtained from the three variations of the algorithms for different values of m.
Observe that the average loops of the categorical Bernoulli factory decreases with an increase in the number of proposals. Also, the average loops in the Portkey

m	5	10	100
Avg loops categorical	4751.27319	78.92179	50.81533
Avg loops bernoulli	60.82526	32.8379	77.82781
Max loops categorical	27484092	1619503	629765
Max loops bernoulli	742	926	976

Table 2: Categorical Bernoulli and the Portkey algorithm with $\beta = 0.99$
x

m	5	10	100
Avg loops categorical	8186.492	49.06611	50.75465
Avg loops bernoulli	350.8424	2115.254	5468.922
Max loops categorical	1465235736	1202788	1149513
Max loops bernoulli	10728075	247129580	35421630

Table 3: Categorical Bernoulli and the two-coin algorithm using telescopic product



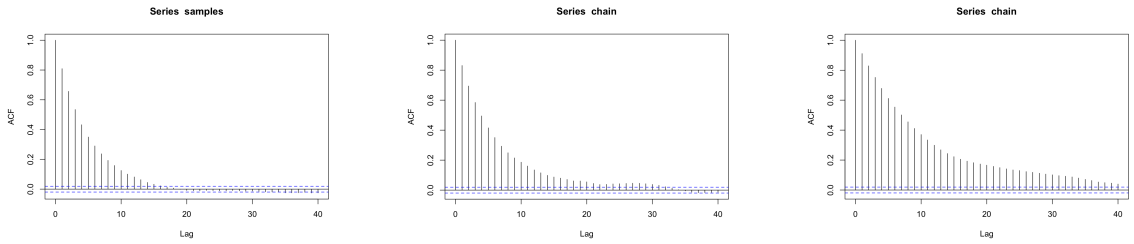
(a) $m = 5$ (b) $m = 10$ (c) $m = 100$

Figure 1: ACF plots for the categorical and two-coin Bernoulli factory

algorithm are significantly less than the average loops in the two-coin algorithm, but the Markov chain converges slower as seen in the ACF plots.

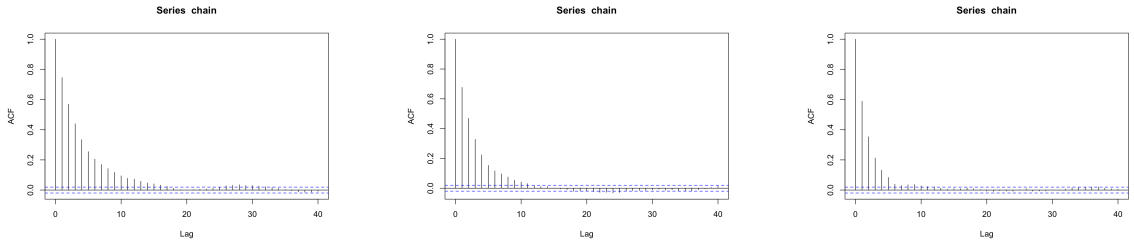
7 Future Work

It is evident that the two-coin algorithm is inefficient even when multiple proposals are used. The Portkey version decreases the number of loops, but that involves modifying the acceptance ratio. Huber introduced a logistic Bernoulli factory (Huber,



(a) $m = 5$ (b) $m = 10$ (c) $m = 100$

Figure 2: ACF plots for the categorical and Portkey Bernoulli factory, using $\beta = 0.99$



(a) $m = 5$ (b) $m = 10$ (c) $m = 100$

Figure 3: ACF plots for the categorical and two-coin Bernoulli factory using telescopic product

2016) to accept a proposal with probability $r/(1+r)$, which we believe is more efficient than the two-coin algorithm. If a suitable generalization can be found where Huber's Bernoulli factory can be used to accept a proposal with probability $r1/(r1+r2)$, that would yield an algorithm more efficient than the one which uses two-coin.
The example given in Section 6 discusses the case where the target distribution is intractable, but this algorithm can also be used in problems where the proposal distribution is intractable. Because the choice of the proposal depends on the user, we may get a tighter bound on the proposal than the target leading to increased efficiency.

References

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov Chain Monte Carlo*. CRC press.

Frenkel, D. and Smit, B. (1996). *Understanding molecular simulation : from algorithms to applications. 2nd ed*, volume 50.

Gonçalves, F. B., Łatuszyński, K., and Roberts, G. O. (2017). Barker's algorithm for Bayesian inference with intractable likelihoods.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109.

Huber, M. (2016). Optimal linear Bernoulli factories for small mean problems.

Laloy, E. and Vrugt, J. (2012). High-dimensional posterior exploration of hydrologic model using multiple-try dream(zs) and high-performance computing. *Water Resources Research*, 50.

Liu, J., Liang, F., and Wong, W. (2000). The multiple-try method and local optimization in Metropolis sampling. *Journal of the American Statistical Association*, 95:121–134.

Martino, L. (2018). A review of multiple try MCMC algorithms for signal processing. *Digital Signal Processing*, 75:134–152.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

Peskun, B. P. H. and Peskun, P. H. (1973). Optimum Monte-Carlo sampling using Markov chains. *Biometrika*, 60:607–612.

Robert, C. and Casella, G. (2000). Monte carlo statistical method. *Technometrics*, 42.

Robert, C. P., Elvira, V., Tawn, N., and Wu, C. (2018). Accelerating mcmc algorithms.

Roberts, G. O. and Rosenthal, J. S. (2004). General state space markov chains and mcmc algorithms. *Probability Surveys*, 1(none).

Vats, D., Gonçalves, F., Łatuszyński, K., and Roberts, G. O. (2021). Efficient Bernoulli factory MCMC for intractable posteriors.