Nested MCMC and Tempering MCMC Comparison

Sitong Liu

February 20, 2022

1 Introduction

The Cut model is an alternative to Bayesian inference designed to remove unwanted feedback from poorly specified modules (Carmona and Nicholls, 2020). Nested MCMC and tempering MCMC are two widely-used methods to simulate samples from the cut posterior. This report aims to compare the two algorithms in terms of efficiency and precision using the Biased data.

2 Data and Model

In this section, we introduce the Biased data that we use to analyze in this report.

The Biased data is a simple synthetic example where the "misspecification" comes from the poorly chosen prior. Suppose we have two datasets informing an unknown parameter φ. The first is a "reliable" small sample $Z = (Z_1, \dots, Z_m)$, $Z_i \sim N(\varphi, \sigma_z^2)$, iid for $i = 1, \dots, m$ distribution, with σ_z unknown; the second is a larger sample $Y = (Y_1, \dots, Y_n)$, $Y_i \sim N(\varphi + \theta, \sigma_y^2)$, iid for $i = 1, \dots, n$, with σ_y unknown. The "bias" θ is unknown.

This model was used by Liu et al. (2009), Jacob et al. (2017), and Carmona and Nicholls (2020). As assigned by Carmona and Nicholls (2020), we set m = 25, n = 50, the true generative parameters $\varphi^* = 0$, $\theta^* = 1$, and $\sigma_z = 2$, $\sigma_y = 1$; assign a constant prior for φ, and a conjugate prior $N(0, \sigma_\theta^2)$ for θ, where $\sigma_\theta = 0.33$. Note that σ_θ is assigned a different value from Carmona and Nicholls (2020) to generate a more extreme result.

3 Cut Model

The cut model defined for the Biased data introduced in Section 2 is $p_{cut}(\varphi, \theta | Z, Y) = p(\varphi|Z)p(\theta|Y, \varphi)$, where

$$p(\varphi|Z) = \frac{p(Z|\varphi)p(\varphi)}{p(Z)}, \quad p(Z) = \int p(Z|\varphi)p(\varphi)d\varphi;$$

$$p(\theta|Y,\varphi) = \frac{p(Y|\theta,\varphi)p(\theta|\varphi)}{p(Y|\varphi)}, \quad p(Y|\varphi) = \int p(Y|\theta,\varphi)p(\theta|\varphi)d\theta.$$

In this example, since it involves Gaussian distribution, both posteriors for φ and θ can be written in closed forms:

$$\begin{aligned}
p(\varphi|Z) &\propto p(Z|\varphi)p(\varphi) \\
&\propto \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma_z} \exp\left(-\frac{1}{2\sigma_z^2}(z_i-\varphi)^2\right) \\
&\propto \exp\left(-\frac{(\varphi-\sum_{i=1}^{m} z_i/m)^2}{2\sigma_z^2/m}\right);
\end{aligned}$$

$$\begin{aligned}
p(\theta|Y,\varphi) &\propto p(Y|\theta,\varphi)p(\theta|\varphi) \\
&\propto \exp\left(-\frac{1}{2\frac{\sigma_y^2}{n}}(\bar{y}-(\theta+\varphi))^2 + \frac{1}{2\sigma_\theta^2}\theta^2\right) \\
&\propto \exp\left(-\frac{\sigma_\theta^2(\bar{y}-\varphi)^2+\sigma_\theta^2\theta^2-2\sigma_\theta^2(\bar{y}-\varphi)\theta+\frac{\sigma_y^2}{n}\sigma^2}{2\frac{\sigma_y^2}{n}\sigma_\theta^2}\right) \\
&\propto \exp\left(-\frac{(\theta^2-2\mu_{\theta|Y,\varphi}\theta+\mu_{\theta|Y,\varphi}^2)+\mu_{\theta|Y,\varphi}(\bar{y}-\varphi)-\mu_{\theta|Y,\varphi}^2}{2\sigma_{\theta|Y,\varphi}^2}\right) \\
&\propto \exp\left(-\frac{(\theta-\mu_{\theta|Y,\varphi})^2}{2\sigma_{\theta|Y,\varphi}^2}\right);
\end{aligned}$$

where $\mu_{\theta|Y,\varphi} = \frac{n\sigma_\theta^2(\bar{y}-\varphi)}{n\sigma_\theta^2+\sigma_y^2}$, $\quad \sigma_{\theta|Y,\varphi}^2 = \frac{\sigma_y^2\sigma_\theta^2}{\sigma_y^2+n\sigma_\theta^2}$.

So, the Cut model is

$$p_{cut}(\varphi,\theta|Z,Y) = p(\varphi|Z)p(\theta|Y,\varphi); \tag{1}$$

$$p(\varphi|Z) = N(\varphi;\bar{Z},\frac{\sigma_z^2}{m}), \quad \varphi\in\Re; \tag{2}$$

$$p(\theta|Y,\varphi) = N(\theta;\mu_{\theta|Y,\varphi},\sigma_{\theta|Y,\varphi}^2), \quad \theta\in\Re; \tag{3}$$

where $\bar{Z} = m^{-1}\sum_{i=1}^{m} Z_i$.

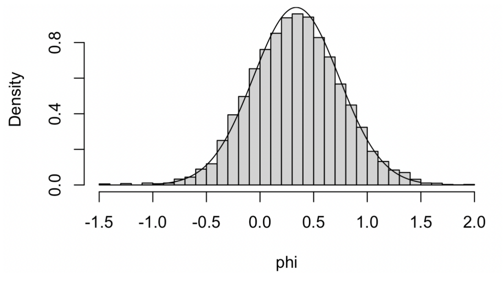
4 Parameter Update

In this section, we introduce the approach to update φ, θ targeting the cut posterior, refering to Eq. 1. It is worth noting that both φ and θ can be sampled directly using the explicit formular (Eq. 2 and Eq. 3).

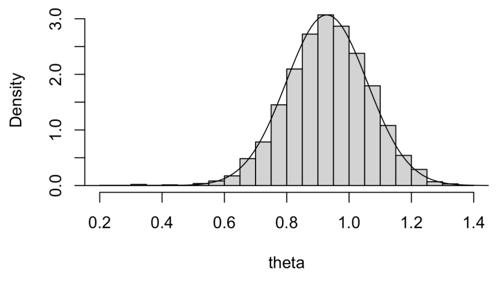
4.1 Update φ

To update φ parameter, we use a random walk Metropolis algorithm. The random walk proposal is $\phi^{(t)} = \varphi^{(t-1)} + W$ (ϕ refers to the proposal state and φ refers to the Markov chain state), where $W \sim g$, $\quad g \sim N(w;0,\sigma_w^2)$ and σ_w^2 is determined empirically. Hence, we have $q(\phi^{(t)}|\varphi^{(t-1)}) = g(\phi^{(t)}-\varphi^{(t-1)}) = g(-\phi^{(t)}+\varphi^{(t-1)}) = q(\varphi^{(t-1)}|\phi^{(t)})$. Since φ has a flat prior, the Metropolis-Hastings rejection ratio α depends only on the likelihood ratio between the current and proposed values, i.e. $\frac{p(Z;\phi^{(t)})}{p(Z|\varphi^{(t-1)})}$.

To check the validity of φ MCMC update, we plot MCMC φ histogram with the exact density (Eq. 2) superposed. The exact density correctly characterises the histogram, so the MCMC update is valid. The result is displayed in Fig. 1(a).



(a)



(b)

Figure 1: Validity check of MCMC update. (a) plot MCMC φ histogram with the exact density superposed; (b) plot MCMC θ histogram with the exact density superposed.

4.2 Update θ

To update θ parameter, we also use a random walk Metropolis algorithm. The random walk proposal is $\Theta^{(t)} = \theta^{(t-1)} + W$, ($\Theta$ refers to the proposal state and θ refers to the Markov chain state), where $W \sim g, \quad g \sim N(w; 0, \sigma_w^2)$ and σ_w^2 is determined empirically. Hence, we have $q(\Theta^{(t)}|\theta^{(t-1)}) = g(\Theta^{(t)} - \theta^{(t-1)}) = g(-\Theta^{(t)} + \theta^{(t-1)}) = q(\theta^{(t-1)}|\Theta^{(t)})$. The Metropolis-Hastings rejection ratio $\alpha = \frac{p(\Theta^{(t)}|Y,\varphi)}{p(\theta^{(t-1)}|Y,\varphi)}$.

To check the validity of θ MCMC update, we plot MCMC θ histogram with the exact density (Eq. 3) superposed (with $\varphi = 0$ fixed). The exact density correctly characterises the histogram, so the MCMC update is valid. The result is displayed in Fig. 1(b).

5 Nested MCMC

In this section, we introduce how to sample (φ, θ) targeting the cut posterior (Eq. 1), using nested MCMC algorithm.

5.1 Algorithm

The basic procedure of nested MCMC is shown in *Algorithm 1*: sample N_1 draws from $p(\varphi|Z)$; for each sampled value of φ, run a side-chain targeting $p(\theta|Y, \varphi)$ for N_2 steps, where N_2 is large enough to avoid initialisation bias; keep only the last sampled value in this side-chain. The resulting joint samples (φ, θ) are approximately distributed according to the cut posterior.

5.2 Parameters and Tuning Criteria

There are four parameters to specify in a nested MCMC algorithm:

- N_1: run length of the main chain;
- N_2: run length of the side-chain;

Algorithm 1 Nested MCMC Targeting the Cut Posterior (Eq. 1)

```
1: for $i = 1, \ldots, N_1$ do
2:     Sample $(\varphi^{(i)}) \sim p(\varphi|Z)$ using any standard sampler
3: end for
4: Let $\{\varphi^{(i)}\}_{i=1}^{N_1}$ be samples after burn-in and thinning
5: for $i = 1, \ldots, N_1$ do
6:     for $j = 1, \ldots, N_2$ do
7:         Sample $(\theta^{(i,j)}) \sim p(\theta|Y, \varphi^{(i)})$, using any standard sampler
8:         (initialize the jth side-chain with the final state of the j-1th side-chain)
9:     end for
10:    Let $\theta^{(i)} = \theta^{(i,N_2)}$ (final state)
11: end for
12: return $\{(\theta^{(i)}, \varphi^{(i)})\}_{i=1}^{N_1}$
```

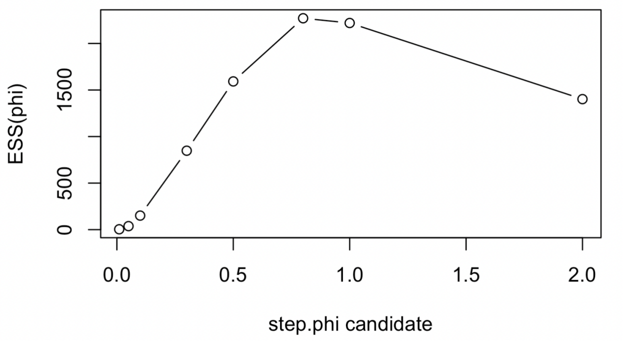


Figure 2: Effective sample size of φ with different step.phi values.

- step.phi: step size to update φ using a random walk Metropolis algorithm;
- step.theta: step size to update θ using a random walk Metropolis algorithm.

These parameters are tuned based on the following criteria:

- N_1 should be set sufficiently large to ensure that the Markov chain converges to the targeting distribution;
- N_2 is chosen as the smallest length of side-chain that ensures convergence;
- The optimal step.phi is the one that maximizes the effective sample size of sampled φ;
- step.theta is chosen to make the optimal ntemper=8 (this generates a clear distinction between nested MCMC and tempering MCMC, details will be given later).

Here we display the parameter tuning result.

N_1 is set to be 10000 throughout the experiments.

We choose eight step.phi candidates: (0.01, 0.05, 0.1, 0.3, 0.5, 0.8, 1, 2), and calculate the effective sample size of φ in each sampled Markov chain. step.phi = 0.8 gives the biggest effective sample size, so we set step.phi = 0.8 to update φ. The result is shown in Fig. 2.

step.theta is set to be 0.15, which is a reasonable choice for both nested MCMC and tempering MCMC.

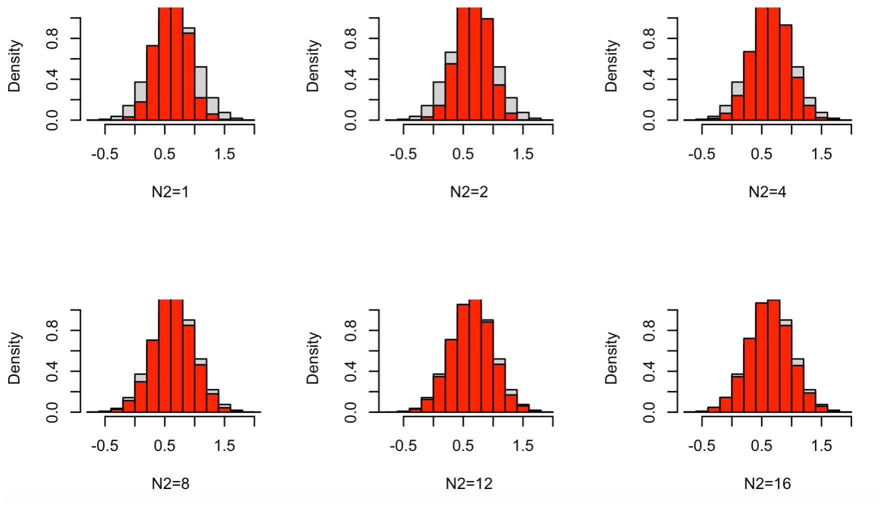


Figure 3: Comparison of MCMC θ distribution with different N_2 values. The red histogram corresponds to N_2 we are interested in, i.e. $N_2 = 1, 2, 4, 8, 12, 16$, and the grey histogram is generated by the true distribution.

After fixing the optimal step.phi and step.theta value, we select the smallest N_2 that ensures convergence of the side-chain. The idea is to start with a relatively small N_2, and compare the result (histogram of the sampled value) with that generated by the true distribution (Eq. 3), to see whether they are the same. If so, the smaller N_2 is acceptable. Then, repeat the procedure, try with a smaller N_2, and stop until the results are not the same. We try with $N_2 = (1, 2, 4, 8, 12, 16)$, and it turns out that $N_2 = 8$ yields a θ distribution close to truth, while $N_2 = 8$ is not very big. Hence, it can be concluded that $N_2 = 8$ is approximately the optimal side-chain length. The result is displayed in Fig. 3.

With the optimal parameter choice, i.e. $N_1 = 10000$, $N_2 = 8$, step.phi $= 0.8$, step.theta $= 0.15$, Fig. 4 shows some sample output.

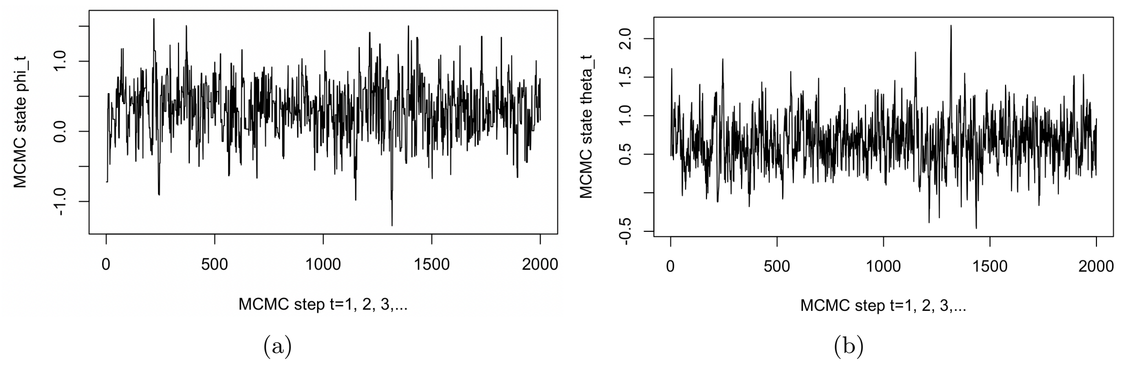


Figure 4: Nested MCMC targeting the cut posterior (Eq. 1) with the optimal parameters: (a) MCMC trace of φ^t plotted against t = 1,...,2000; (b) MCMC trace of θ^t plotted against t = 1,...,2000.

6 Tempering MCMC

In this section, we introduce how to sample (φ, θ) targeting the cut posterior (Eq. 1), using tempering MCMC algorithm.

6.1 Algorithm

The basic procedure of tempering MCMC is shown in *Algorithm 2*: instead of moving directly from φ^{t-1} to φ^t, we move along a linear path $\varphi(c) = c\varphi^t + (1-c)\varphi^{t-1}$ in a sequence $c^1, \ldots, c^m$, with $c^i = i/m$. At each step we draw a new sample of θ but keep only the sample at the last step m, which becomes θ^t.

Algorithm 2 Tempering MCMC Targeting the Cut Posterior (Eq. 1)

```
1: for $t = 1, \ldots, T$ do
2:     Sample $(\varphi^{(t)}) \sim p(\varphi|Z)$ using any standard sampler
3:     for $m = 1, \ldots, M$ do
4:         Let $(\varphi^{(t,m)}) = ((M-j) \cdot \varphi^{(t-1)} + m \cdot \varphi^{(t)})/M$
5:         Sample $(\theta^{(t,m)}) \sim p(\theta|Y, (\varphi^{(t,m)}))$ using any standard sampler
6:     end for
7:     Let $\theta^{(t)} = \theta^{(t,M)}$ (final state)
8: end for
9: return $\{(\theta^{(t)}, \varphi^{(t)})\}_{t=1}^{T}$
```

6.2 Parameters and Tuning Criteria

There are four parameters to specify in a tempering MCMC algorithm:

- niter: iteration number of the main loop (corresponding to T in *Algorithm 2*);
- ntemper: step number of inner loop in each main loop iteration (corresponding to M in *Algorithm 2*);
- step.phi: step size to update φ using a random walk Metropolis algorithm;
- step.theta: step size to update θ using a random walk Metropolis algorithm.

These parameters are tuned based on the following criteria:

- niter, step.phi and step.theta have the same tuning criteria as those in nested MCMC, i.e niter is sufficiently large to guarantee convergence of the Markov chain, step.phi maximizes the effective sample size of φ, and step.phi is chosen to make the optimal ntemper = 10;
- ntemper is determined empirically and should ensure convergence.

Here we display the parameter tuning result.

niter = 10000 throughout the experiments. step.phi = 0.8 in accordance with nested MCMC. Fig. 5 shows the result of applying the tempering MCMC with step.theta = 0.15 with the number of steps, i.e. ntemper, ranging from 1 to 16. It turns out that ntemper=8 yields a θ distribution close to truth. Hence, it can be concluded that ntemper=8 is approximately the optimal side-chain length.

With the optimal parameter choice, i.e. niter = 10000, ntemper = 8, step.phi = 0.8, step.theta = 0.15, Fig. 6 shows some sample output.

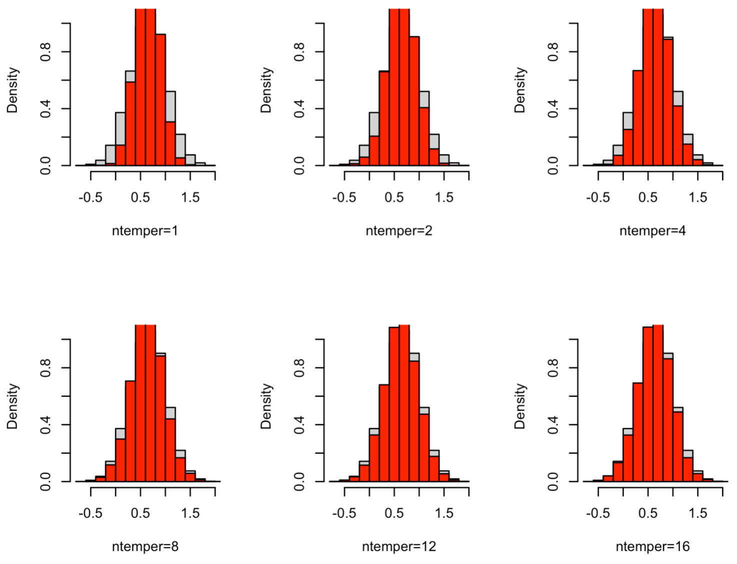


Figure 5: Comparison of MCMC θ distribution with different ntemper values. The red histogram correspons to ntemper we are interested in, i.e ntemper = 1, 2, 4, 8, 12, 16, and the grey histogram is generated by the true distibution.

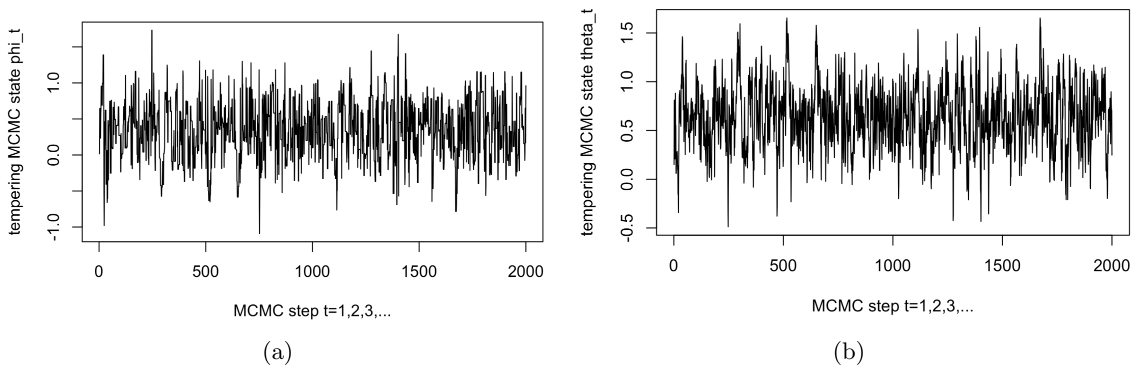


Figure 6: Tempering MCMC targeting the cut posterior (Eq. 1) with the optimal parameters: (a) MCMC trace of φ^t plotted against t = 1,...,2000; (b) MCMC trace of θ^t plotted against t = 1,...,2000.

7 Analysis

In this section, we compare nested MCMC and tempering MCMC in terms of efficiency and precision. Since both algorithms are essentially the same in sampling φ, we will only focus on θ samples.

7.1 Efficiency

We introduce Effective Sample Size (ESS) to evaluate efficiency, which is a measure of the precision gain afforded by our N_1 correlated samples:

$$var(\bar{f_n}) = \frac{var(f(X))}{ESS},$$

where $\bar{f}_n = N_1^{-1} \sum_t f(X_t)$.

There are two basic rules of comparing efficiency between two MCMC algorithms, statistical efficiency and computational efficiency. The former one is compared in terms of ESS/n (n = N_1 or niter in this setting), which is the effective independent samples per MCMC sample. In practice, it is common that computationally expensive methods with a high statistical efficiency per sample are slow to compute. Therefore, we often compare MCMC algorithms based on computational efficiency, which is defined as

$$\rho = \frac{ESS}{S}, \tag{4}$$

where S is the time in CPU seconds to make n steps of the MCMC. Here, we use computational efficiency. Note that higher computational efficiency implies better efficiency.

7.2 Precision

We use two methods to measure precision: energy distance and Kullback-Leibler divergence.

7.2.1 Energy Distance

Energy distance is a statistical distance between probability distributions. Suppose $U = (U_1, \ldots, U_n)$ and $V = (V_1, \ldots, V_m)$ are samples from two different distributions that we want to compare, the energy score is:

$$ES(U,V) = \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\|U_i - V_j\|_2 - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|U_i - U_j\|_2 - \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\|V_i - V_j\|_2.$$

To compare how precise the MCMC samples of θ is compared to the true distribution, we set

$$Y = (\theta^{(1)}, \ldots, \theta^{(T)}),$$

where $\theta^{(t)} \sim N(\theta; \mu_{\theta|Y,\varphi}, \sigma^2_{\theta|Y,\varphi}), \quad t = 1, \ldots, T$;

$$Z = (\tilde{\theta}^{(1)}, \ldots, \tilde{\theta}^{(T)}),$$

where $\tilde{\theta}^{(t)}$ are samples generated by nested MCMC algorithm, $t = 1, \ldots, T$.

Note that smaller energy distance indicates better precision, i.e. closer to the true distribution.

7.2.2 Kullback-Leibler Divergence

The Kullback-Leibler divergence (KL divergence), i.e. $D_{KL}(P||Q)$, is a measure of how one probability distribution Q is different from a second, reference probability distribution P. For distributions P and Q of a continuous random variables,

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) log(\frac{p(x)}{q(x)})dx,$$

where p, q denote the probability densities of P and Q.

In this example, p is the exact cut posterior distribution, and q is the distribution of MCMC samples. Since we only have MCMC samples from q (not the explicit expression for q), it might be difficult to calculate KL-divergence directly. Therefore, we adopt a kind of contrastive learning approach to estimate KL divergence.

Let $p(\theta) = \pi^{cut}(\theta|Y,\varphi)$ be the exact cut posterior, refering to Eq. 3. To be precise, $\pi^{cut}(\theta|Y,\varphi) = \int_{-\infty}^{\infty} p(\varphi|Z)p(\theta|Y,\varphi)d\varphi$, sample $\theta \sim \pi^{cut}(\cdot|Y,Z)$ by sampling $(\varphi,\theta) \sim p(\varphi|Z)p(\theta|Y,\varphi)$ and ignore φ.

Let $q(\theta) = \pi^{nested}(\theta|Y,Z)$ be the distribution of θ drawn using nested MCMC (or equivalently, $q(\theta) = \pi^{temper}(\theta|Y,Z)$ be the distribution of θ drawn using tempering MCMC). Note than we can only sample from these distributions but cannot evaluate them.

Let $\theta^{(t)} \sim p(\theta)$, $t = 1,\ldots,T$; $\quad \theta^{(t)} \sim q(\theta)$, $t = T+1,\ldots,2T$.

Let $c_t = 1$, $t = 1,\ldots,T$; $\quad c_t = 0$, $t = T+1,\ldots,2T$, i.e. C is an indicator with $c_t = 1$ indicating samples from true distribution and $c_t = 0$ indicating MCMC samples.

Then we have $C \sim \text{Bernoulli}(d(\theta))$, where $C = (c_t)_{t=1}^{2T}$ and $d(\theta)$ is a function of θ, representing the probability of obtaining a sample from $p(\theta)$ in a Bernoulli trial.

Try to fit C with logistic regression:

$$d(\theta) = \frac{e^{\eta(\theta)}}{1+e^{\eta(\theta)}},$$

where $\eta(\theta) = \alpha + \beta\theta$.

By maximizing log-likelihood, we can get $\hat{\beta}$:

$$\begin{aligned}\hat{\beta} &= \text{argmax } l(\beta;\theta,C) \\ &= \text{argmax } \{\frac{1}{T}\sum_{t=1}^{T} log(d(\theta_t;\beta)) + \frac{1}{T}\sum_{t=T+1}^{2T} log(1-d(\theta_t;\beta))\}.\end{aligned}$$

It has been shown by Gutmann and Hyvärinen (2012) that $d(\theta;\hat{\beta})$ is approximately equal to $\frac{p(\theta)}{p(\theta)+q(\theta)}$. Thus, $\frac{p(\theta)}{q(\theta)} \approx \frac{d(\theta;\hat{\beta})}{1-d(\theta;\hat{\beta})}$.

Therefore, we can estimate KL divergence by:

$$D_{KL}(P||Q) = E_{\theta\sim p}(log(\frac{p(\theta)}{q(\theta)})) \tag{5}$$

$$\approx \frac{1}{T}\sum_{t=1}^{T} log(\frac{d(\theta_t)}{1-d(\theta_t)}). \tag{6}$$

If $q(\theta)$ has the same distribution with $p(\theta)$, then $d(\theta) = \frac{1}{2}$ for all θ. In this case, KL = 0, theoretically. Note that smaller KL divergence means better precision, i.e. closer to the true distribution.

To have a feeling of how this method of estimating KL divergence works, we use a toy example, i.e. two normal distributions with the same variance and different mean, to illustrate. Let $X_0 \sim N(0,1)$ denote the reference distribution and $X \sim N(\mu,0)$ denote the test distribution, where we use μ = 0, 0.1, 0.5, 1, 2. Table 1 shows the result and note that the estimate of KL divergence does not equal to 0 when $\mu = 0$, i.e., the two distributions are the same. Thus, noise exists in this method.

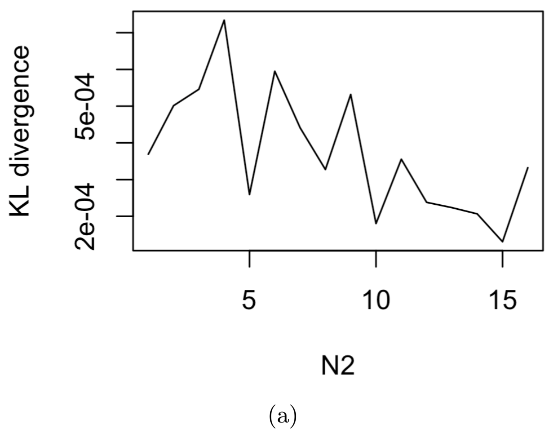
7.3 Results

After setting up the standards to evaluate MCMC algorithms, we now show the comparison results. We vary the choice of parameters and see how efficiency and precision change with different parameter values. To generate the plots, we produce 10 synthetic datasets, and take the average values.

μ	KL divergence
0	4.92×10^{-5}
0.1	5.95×10^{-3}
0.5	0.12
1.0	0.49
2.0	2.02

Table 1: KL divergence estimate with different μ in a toy example.

In nested MCMC, we test with N_2 ranging from 1 to 16 with an increase interval of 1, i.e. $N_2 = (1, 2, 3, \dots, 16)$. KL divergence and energy distance decreases as N_2 increases with some oscillations possibly caused by noise (Fig. 7). As N_2 increases, the computational efficiency decreases in general (see Fig. 8). Fig. 9 is the scatter plot combining precision and efficiency of nested MCMC with varying N_2 values. The exact data is given in Table 2. We can see that there is a trade-off between precision and efficiency, i.e. as precision increases, efficiency decreases.



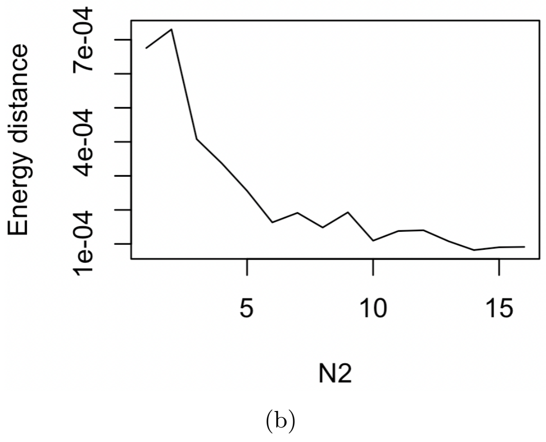


Figure 7: Precision of nested MCMC with N2 ranging from 1 to 16: (a) KL divergence; (b) Energy distance.

In tempering MCMC, we test with ntemper ranging from 1 to 16 with an increase interval of 1, i.e. ntemper $= (1, 2, 3, \dots, 16)$, the same value as N_2 in nested MCMC . KL divergence and energy distance both show a decreasing trend as ntemper increases (Fig. 10). The computational efficiency decreases as ntemper increases and slightly increases at small temper values (see Fig. 11). Fig. 12 is the scatter plot combining precision and efficiency of tempering MCMC with varying ntemper values. The exact data is given in Table 3. The general feature is there is a trade-off between precision and efficiency, i.e. as precision increases, efficiency decreases.

Now, we look at the performance of nested MCMC and tempering MCMC when $N_2 =$ ntemper. From Fig. 13 we can see that when $N2 =$ ntemper, tempering MCMC has small divergence, i.e better precision. In terms of computational efficiency, nested MCMC is slightly more competitive (see Fig. 14).

Fig. 15 is the scatter plot of both algorithm results. Nested MCMC and tempering MCMC have similar trade-off trend and performace. As precision increases, the computational efficiency generally decreases.

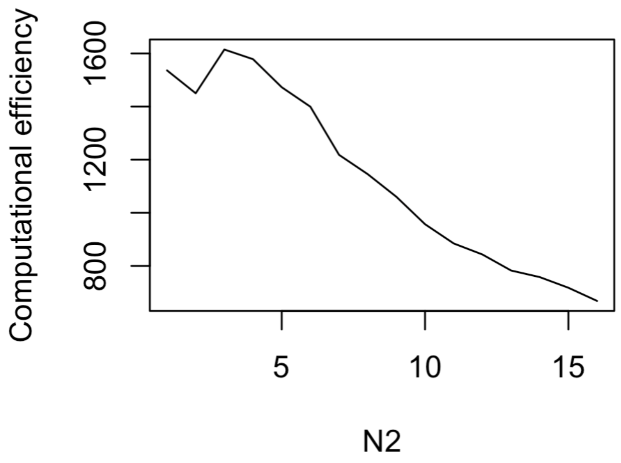
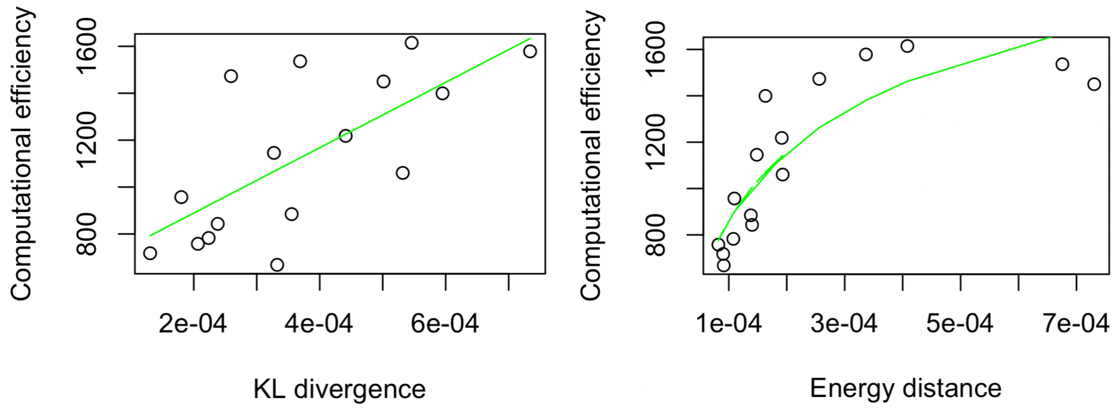


Figure 8: Computational efficiency of nested MCMC with N2 ranging from 1 to 16.

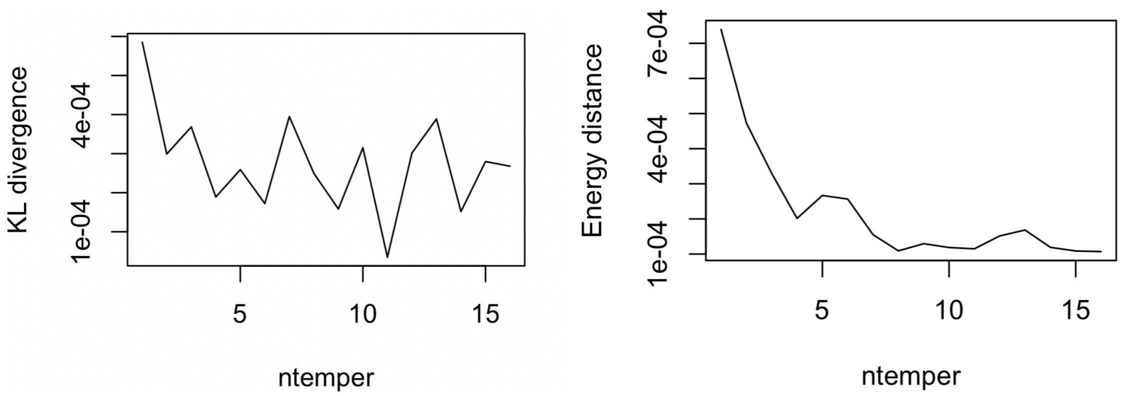
Algorithm type	**Computational efficiency ρ**	**KL divergence**	**Energy distance**
$N_2 = 1$	1536	3.69×10^{-4}	6.76×10^{-4}
$N_2 = 2$	1450	5.01×10^{-4}	7.31×10^{-4}
$N_2 = 3$	1615	5.46×10^{-4}	4.08×10^{-4}
$N_2 = 4$	1578	7.34×10^{-4}	3.37×10^{-4}
$N_2 = 5$	1473	2.59×10^{-4}	2.56×10^{-4}
$N_2 = 6$	1400	5.95×10^{-4}	1.63×10^{-4}
$N_2 = 7$	1218	4.41×10^{-4}	1.91×10^{-4}
$N_2 = 8$	1146	3.27×10^{-4}	1.48×10^{-4}
$N_2 = 9$	1060	5.32×10^{-4}	1.93×10^{-4}
$N_2 = 10$	957	1.80×10^{-4}	1.10×10^{-4}
$N_2 = 11$	885	3.55×10^{-4}	1.38×10^{-4}
$N_2 = 12$	843	2.34×10^{-4}	1.41×10^{-4}
$N_2 = 13$	783	2.24×10^{-4}	1.08×10^{-4}
$N_2 = 14$	758	2.07×10^{-4}	8.19×10^{-5}
$N_2 = 15$	718	1.31×10^{-4}	9.03×10^{-5}
$N_2 = 16$	668	3.32×10^{-4}	9.13×10^{-5}

Table 2: Efficiency and precision data of nested MCMC with N_2 ranging from 1 to 16.



(a) (b)

Figure 9: Comparison of nested MCMC with N_2 ranging from 1 to 16: (a) KL divergence; (b) Energy distance. The green line is the regression fit line.



(a) (b)

Figure 10: Precision of tempering MCMC with ntemper ranging from 1 to 16: (a) KL divergence; (b) Energy distance.

8 References

Carmona, C. U., Nicholls, G. K. (2020). Semi-Modular Inference: enhanced learning in multi-modular models by tempering the influence of components. 108, 4226-4235.

Gutmann, M. U., Hyvärinen, A. (2012). Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. Journal of machine learning research, 13(2).

Jacob, P. E., Murray, L. M., Holmes, C. C., and Robert, C. P. (2017). Better together? Statistical learning in models made of modules.

Liu, F., Bayarri, M. J., and Berger, J. O. (2009). Modularization in Bayesian analysis, with emphasis on analysis of computer models. Bayesian Analysis, 4(1):119–150.

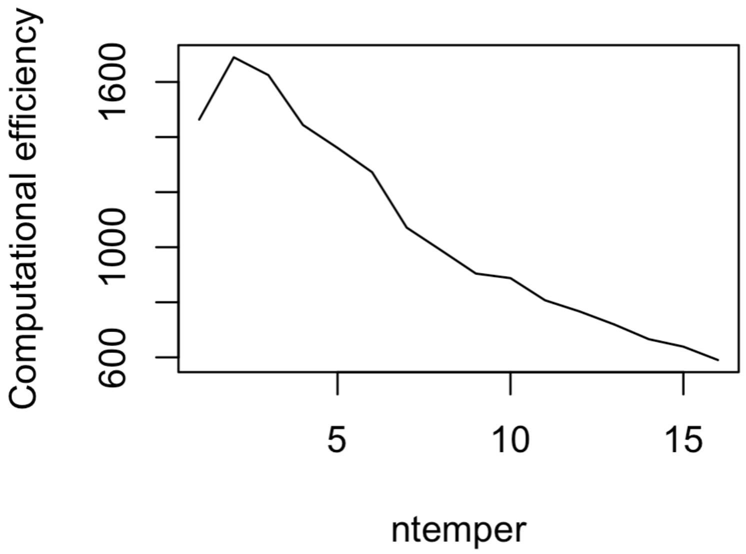
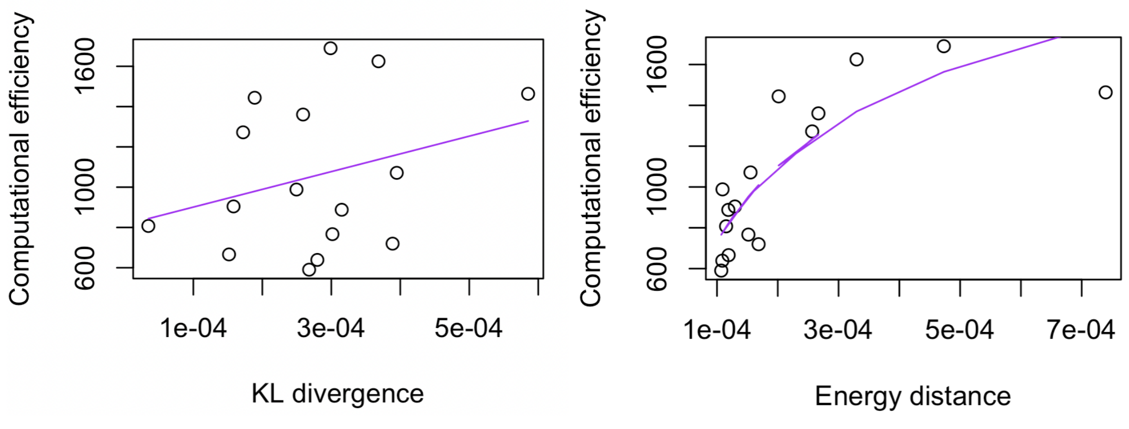


Figure 11: Computational efficiency of tempering MCMC with ntemper ranging from 1 to 16.

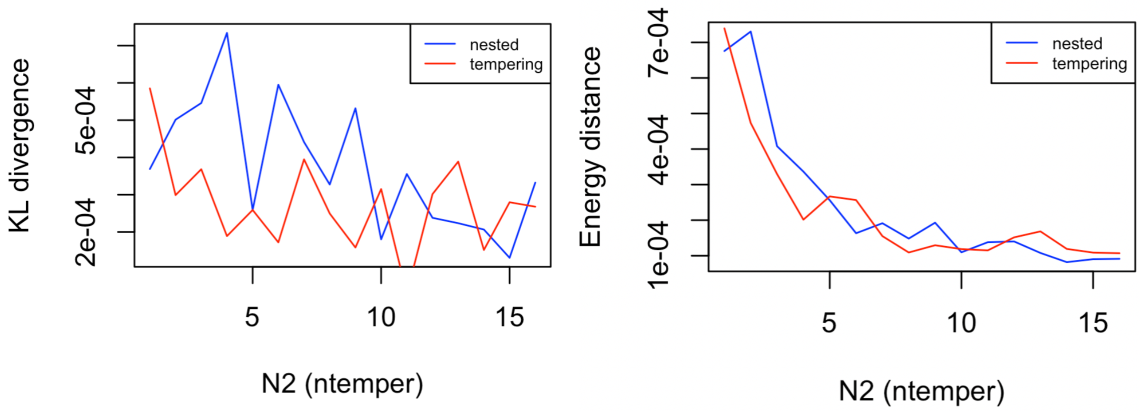
Algorithm type	**Computational efficiency ρ**	**KL divergence**	**Energy distance**
ntemper = 1	1464	5.85×10^{-4}	7.40×10^{-4}
ntemper = 2	1690	2.99×10^{-4}	4.73×10^{-4}
ntemper = 3	1625	3.68×10^{-4}	3.30×10^{-4}
ntemper = 4	1444	1.89×10^{-4}	2.01×10^{-4}
ntemper = 5	1361	2.59×10^{-4}	2.67×10^{-4}
ntemper = 6	1272	1.72×10^{-4}	2.57×10^{-4}
ntemper = 7	1071	3.95×10^{-4}	1.55×10^{-4}
ntemper = 8	988	2.50×10^{-4}	1.09×10^{-4}
ntemper = 9	904	1.58×10^{-4}	1.29×10^{-4}
ntemper = 10	888	3.15×10^{-4}	1.19×10^{-4}
ntemper = 11	807	3.48×10^{-5}	1.15×10^{-4}
ntemper = 12	767	3.02×10^{-4}	1.52×10^{-4}
ntemper = 13	720	3.89×10^{-4}	1.68×10^{-4}
ntemper = 14	666	1.52×10^{-4}	1.19×10^{-4}
ntemper = 15	639	2.80×10^{-4}	1.09×10^{-4}
ntemper = 16	590	2.68×10^{-4}	1.07×10^{-4}

Table 3: Efficiency and precision data of tempering MCMC with ntemper ranging from 1 to 16.



(a) (b)

Figure 12: Comparison of tempering MCMC with ntemper ranging from 1 to 16: (a) KL divergence; (b) Energy distance. The purple line is the regression fit line.



(a) (b)

Figure 13: Precision of nested MCMC and tempering MCMC when N_2 = ntemper: (a) KL divergence; (b) Energy distance.

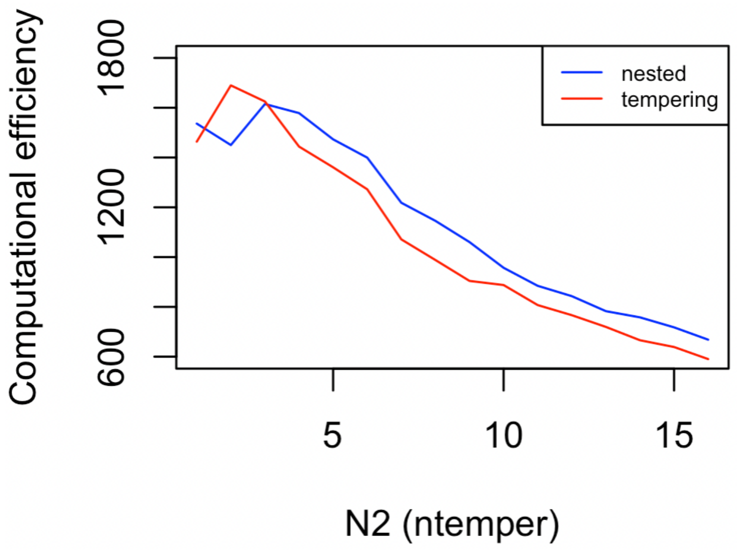


Figure 14: Computational efficiency of ested MCMC and tempering MCMC when N_2 = ntemper.

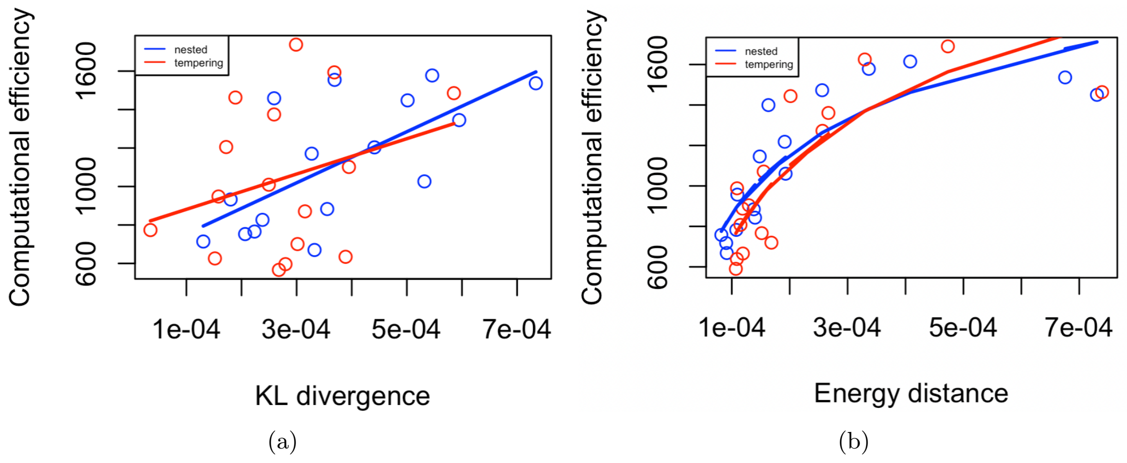


Figure 15: Precision and computational efficiency of nested MCMC and tempering MCMC when N_2 = ntemper: (a) KL divergence; (b) Energy distance.