Monte Carlo Approximation of Bayes Factors via Mixing with Surrogate Distributions

Chenguang Dai and Jun S. Liu*

Department of Statistics, Harvard University

September 16, 2019

Abstract

By mixing the posterior distribution with a surrogate distribution, of which the normalizing constant is tractable, we describe a new method to estimate the normalizing constant using the Wang-Landau algorithm. We then introduce an accelerated version of the proposed method using the momentum technique. In addition, several extensions are discussed, including (1) a parallel variant, which inserts a sequence of intermediate distributions between the posterior distribution and the surrogate distribution, to further improve the efficiency of the proposed method; (2) the use of the surrogate distribution to help detect potential multimodality of the posterior distribution, upon which a better sampler can be designed utilizing mode jumping algorithms; (3) a new jumping mechanism for general reversible jump Markov chain Monte Carlo algorithms that combines the Multiple-try Metropolis and the directional sampling algorithm, which can be used to estimate the normalizing constant when a surrogate distribution is difficult to come by. We illustrate the proposed methods on several statistical models, including the Log-Gaussian Cox process, the Bayesian Lasso, the logistic regression, the Gaussian mixture model, and the g-prior Bayesian variable selection.

1 Introduction

Given data $\boldsymbol{y}$, we consider a finite sequence of competing models $\{\mathcal{M}_k\}$ associated with parameters $\{\boldsymbol{\theta}_k\}$. The marginal likelihood of data under model $\mathcal{M}_k$, also referred to as the normalizing constant, is defined as

$$p(\boldsymbol{y} \mid \mathcal{M}_k) = \int \gamma(\boldsymbol{\theta}_k \mid \boldsymbol{y}, \mathcal{M}_k) d\boldsymbol{\theta}_k = \int p(\boldsymbol{\theta}_k \mid \mathcal{M}_k) p(\boldsymbol{y} \mid \boldsymbol{\theta}_k, \mathcal{M}_k) d\boldsymbol{\theta}_k,$$

in which $p(\boldsymbol{\theta}_k|\mathcal{M}_k)$ is the prior and $\gamma(\boldsymbol{\theta}_k|y, \mathcal{M}_k)$ is the unnormalized posterior distribution. To compare different models, Bayesian methods typically compute the Bayes factor, which is defined as the ratio of the normalizing constants under different models, that is, $B_{i,j} = p(\boldsymbol{y} \mid \mathcal{M}_i)/p(\boldsymbol{y} \mid \mathcal{M}_j)$. With the uniform prior on model $\mathcal{M}_i$ and $\mathcal{M}_j$, $B_{i,j} > 1$ indicates that model $\mathcal{M}_i$ is more favorable than model $\mathcal{M}_j$ given the current data $\boldsymbol{y}$.

We can approximate the Bayes factor by estimating the normalizing constant of each model. For simplicity, we will drop the dependency on $\boldsymbol{y}$ and the model index k in $\gamma(\boldsymbol{\theta}_k \mid \boldsymbol{y}, \mathcal{M}_k)$ when the context is clear, and use Z_γ to denote the normalizing constant of $\gamma(\boldsymbol{\theta})$. Let $\gamma^\star(\boldsymbol{\theta}) = \gamma(\boldsymbol{\theta})/Z_\gamma$ be the corresponding normalized distribution. Computing the normalizing constant Z_γ is essentially a task of calculating an integral. However, in many interesting cases, the complex form of the unnormalized density $\gamma(\boldsymbol{\theta})$, sometimes with high dimensionality, prohibits us from obtaining neither analytic solutions nor easy numerical approximations. Various Monte Carlo strategies have been developed to estimate the normalizing constant, such as Chib's method (Chib, 1995), inverse logistic regression (Geyer, 1994), importance sampling (Gelfand and Smith, 1990), bridge sampling (Meng and Schilling, 1996; Meng and Wong, 1996), path sampling (Gelman and Meng, 1994; Ogata, 1989), sequential importance sampling (Hammersley and Morton, 1954; Rosenbluth and Rosenbluth, 1955; Kong et al., 1994), and sequential Monte Carlo (SMC) (Liu and Chen, 1998; Doucet et al., 2000; Del Moral et al., 2006).

*The authors are gratefully supported by NSF DMS-1613035, DMS-1712714 and NIGMS R01GM122080.

In this article, we present a mixture approach for estimating normalizing constant using the Wang-Landau (WL) algorithm (Wang and Landau, 2001). Our main idea is to construct a matching surrogate distribution $q^*(\boldsymbol{\theta})$ with its normalizing constant Z_q known, and combine $\gamma^\star(\boldsymbol{\theta})$ and $q^\star(\boldsymbol{\theta})$ to form a mixture distribution with an adjustable mixing parameter tuned through the WL algorithm. The ratio $r = Z_\gamma/Z_q$ is then an easy function of the mixing parameter. For notational convenience, we also use $q(\boldsymbol{\theta})$ to denote the unnormalized surrogate distribution. Many of the aforementioned methods also use a surrogate distribution $q^\star(\boldsymbol{\theta})$, and the idea of using the WL algorithm to estimate the ratio $r = Z_\gamma/Z_q$ also appears in Liang (2005) and Atchadé and Liu (2010) in more restricted settings.

The proposed WL mixture approach is different from existing methods in the literature in the following perspectives. First, when we apply the WL algorithm in our setting, there is a natural partition of the target space indicated by the two (or more if needed) mixture components. Second, unlike the method in Liang (2005), we do not require $\gamma^\star(\boldsymbol{\theta})$ and $q^\star(\boldsymbol{\theta})$ to be well-separated. In fact, we recommend to mix $\gamma^\star(\boldsymbol{\theta})$ and $q^\star(\boldsymbol{\theta})$ together so that global jumps between $\gamma^\star(\boldsymbol{\theta})$ and $q^\star(\boldsymbol{\theta})$ can be potentially avoided, and the data augmentation strategy (Diebolt and Robert, 1994) can be used to improve the sampling efficiency. Third, the WL mixture method does not require $\gamma^\star(\boldsymbol{\theta})$ and $q^\star(\boldsymbol{\theta})$ to have any overlap. With the help of mode jumping algorithms such as the Multiple-try Metropolis (MTM) (Liu et al., 2000), it tends to be much more robust than importance sampling based methods such as bridge sampling, which crucially rely on the amount of overlaps between $\gamma^\star(\boldsymbol{\theta})$ and $q^\star(\boldsymbol{\theta})$.

Following Dai and Liu (2019), we introduce an accelerated version of the proposed method using the momentum technique. The main idea is to formulate the WL algorithm as a stochastic gradient descent algorithm minimizing a convex and smooth function, of which the gradient is estimated using Markov chain Monte Carlo (MCMC) iterations. Under this optimization framework, some acceleration strategies can be employed to speed up the convergence of the WL algorithm. Empirically, we find that the simple momentum technique helps improve the efficiency of our algorithm, and we demonstrate it on two statistical models, the Log-Gaussian Cox process and the Bayesian Lasso.

Several extensions of the WL mixture method will be discussed. First, we propose a parallel Wang-Landau (PWL) algorithm built on top of the proposed mixing strategy. Analogous to the ideas used by bridge sampling and SMC, the PWL algorithm introduces a sequence of (unnormalized) intermediate distributions $\{\eta_t(\boldsymbol{\theta})\}_{t=0}^T$ between the surrogate distribution $q^\star(\boldsymbol{\theta})$ and the target distribution $\gamma^\star(\boldsymbol{\theta})$, where $\eta_0(\boldsymbol{\theta}) = q(\boldsymbol{\theta})$ and $\eta_T(\boldsymbol{\theta}) = \gamma(\boldsymbol{\theta})$. For each neighboring pair $\eta_{t-1}(\boldsymbol{\theta})$ and $\eta_t(\boldsymbol{\theta})$, we use the WL mixture method to estimate the ratio of their normalizing constants, and then multiply all the ratios together. We note that this step can be implemented completely in parallel, which makes the PWL algorithm a computationally efficient choice if users have parallel computing resources.

The second extension is utilizing the proposed mixing framework to enable mode finding and jumping for complex target distributions. Many attempts have been made in the literature to handle the complex landscape of the target state space, including simulated tempering (Marinari and Parisi, 1992), parallel tempering, and population MCMC (Geyer and Thompson, 1995; Liang and Wong, 2001). Our method is most similar to parallel tempering, which relies on a sequence of auxiliary distributions so that global jumps are possible by "transporting" samples back and forth from the target distribution to auxiliary distributions. In our method, the surrogate distribution $q^\star(\boldsymbol{\theta})$ serves as an analog of the auxiliary distribution used in parallel tempering, and the WL weight adjustment ensures that the transportation between the target distribution and the auxiliary distribution is sufficiently frequent. The advantage is that whenever we sample from the surrogate distribution $q^\star(\boldsymbol{\theta})$, it provides us with a potential chance to jump towards a different local mode of the target distribution $\gamma^\star(\boldsymbol{\theta})$. Furthermore, based on the identified local modes, we can design a better sampler for the target distribution $\gamma^\star(\boldsymbol{\theta})$ using mode jumping algorithms (Tjelmeland and Hegstad, 2001; Liu et al., 2000).

The third extension concerns with the general Bayesian model selection. Specifically, we can include the model index $\mathcal{M}_k$ as a parameter in the full posterior distribution specified as:

$$p(\boldsymbol{\theta}_k, \mathcal{M}_k \mid \boldsymbol{y}) \propto p\left(\boldsymbol{y} \mid \boldsymbol{\theta}_k, \mathcal{M}_k\right) p\left(\boldsymbol{\theta}_k \mid \mathcal{M}_k\right) p\left(\mathcal{M}_k\right), \tag{1}$$

and use MCMC to traverse the joint model and parameter space. The ratio between the proportions of time that the Markov chain spends in model $\mathcal{M}_i$ and model $\mathcal{M}_j$, adjusted by the prior $p(\mathcal{M}_k)$, serves as a consistent estimator to the Bayes factor $B_{i,j}$. A reversible jump MCMC (RJMCMC) (Green, 1995) is often required to sample across different dimensional spaces. However, it is well-known that constructing an efficient trans-dimensional proposal is challenging (Brooks et al., 2003). To enable efficient RJMCMC, we propose to combine MTM and the directional sampling (Liu et al., 2000) algorithm. The proposed method will be most

effective if $p(\boldsymbol{\theta}_k \mid \boldsymbol{y}, \mathcal{M}_k)$ is uni-modal for each model $\mathcal{M}_k$, and the mode $\widehat{\boldsymbol{\theta}}_k$ can be located reasonably well beforehand. We note that the proposed method is different from the MTM version of RJMCMC algorithm proposed in Pandolfi et al. (2014). Their method mainly focuses on using a computationally favourable weight function in MTM to avoid evaluating the target density, which can be expensive in complex statistical models. Our method is perhaps most similar to the mode jumping algorithm proposed in Tjelmeland and Hegstad (2001). While they design a mixture of Metropolis-Hastings proposals guided by deterministic local optimization to enable large step-size jumps, we utilize the more flexible MTM. The proposed method can serve as an alternate when it is challenging to propose an appropriate surrogate distribution.

The rest of the article is organized as follows. Section 2.1 reviews the general WL algorithm; Section 2.2 proposes our mixture formulation, and explains how we adapt the WL algorithm in the mixture setting to estimate the normalizing constant; Section 2.3 introduces an accelerated version of the WL mixture method; Section 2.4 explains how to use MTM to jump between the two mixture components if $q^\star(\boldsymbol{\theta})$ and $\gamma^\star(\boldsymbol{\theta})$ are relatively separated; Section 2.5 describes a principled way of using the variational approximation to construct the surrogate distribution. Section 3.1 discusses the possible advantages of the WL mixture method compared to importance sampling, bridge sampling, and Chib's method. Three extensions of the WL mixture method are introduced in Section 4, including (1) a parallel variant of the proposed method (Section 4.1); (2) the use of the surrogate distribution to handle multimodality of the target distribution (Section 4.2); (3) an efficient MTM-RJMCMC algorithm to sample the model space (Section 4.3). Section 5 illustrates the utility of the proposed methods with several numerical examples including a Bayesian evaluation of the Log-Gaussian Cox process fitting, a hyper-parameter selection problem for Bayesian Lasso regression, estimating the marginal likelihood for a logistic regression model, determining the number of components of a mixture model fitting, and Bayesian variable selection for linear models under the spike-and-slab g-prior. Section 6 concludes with some final remarks.

2 Normalizing Constant Estimation

2.1 The Wang-Landau algorithm

In order to improve the convenience and efficiency of the multicanonical sampling (Berg and Neuhaus, 1992), Wang and Landau (2001) proposed a simple stochastic adaptive updating algorithm, which quickly becomes a very popular Monte Carlo method for sampling complex physical systems. Given a target distribution $p(\boldsymbol{\theta})$ and a user-specified partition of the target space $\Theta = \cup_{i=1}^{s} \Theta_i$, where s is the total number of subregions, we can use the WL algorithm to estimate the probability mass of the target distribution within each subregion, that is, $\psi(i) = \int_{\Theta_i} p(\boldsymbol{\theta}) d\boldsymbol{\theta}$. The main steps of the WL algorithm are outlined in Algorithm 1. Here K_t is a

Algorithm 1 Wang-Landau algorithm (Wang and Landau, 2001)

1. Sample $\boldsymbol{\theta}_t$ from $K_{t-1}(\boldsymbol{\theta}_{t-1}, \cdot)$;
2. Update $\psi_t(i) = \psi_{t-1}(i)\left[1 + \eta_t \mathbb{1}(\boldsymbol{\theta}_t \in \Theta_i)\right]$ for $i \in [1:s]$;
3. Normalize $\{\psi_t(i)\}_{i=1}^{s}$ to sum 1.

Markov kernel invariant to the adaptive target distribution $p_t^\dagger(\boldsymbol{\theta})$ defined as

$$p_t^\dagger(\boldsymbol{\theta}) \propto \sum_{i=1}^{s} \frac{p(\boldsymbol{\theta})}{\psi_t(i)} \mathbb{1}(\boldsymbol{\theta} \in \Theta_i), \tag{2}$$

and we can simply initialize $\psi_0(i)$ as $1/s$. Parameter η_t is the learning rate, and typically we should decrease it at the rate of $1/t$ so as to guarantee the convergence of the algorithm. The WL algorithm essentially updates the reweighting factors $\{\psi_t(i)\}$ based on the *flat histogram* criterion, that is, after $\{\psi_t(i)\}$ has converged, the chain should spend equal amount of time within each subregion. The convergence of $\psi_t(i)/\psi_t(j)$ to $\psi(i)/\psi(j)$ for $i, j \in [1:s]$ has been established in Atchadé and Liu (2010) under proper conditions.

2.2 The surrogate mixture method

Suppose we have a (unnormalized) surrogate distribution $q(\boldsymbol{\theta})$ satisfying the following two conditions: (i) preferably, $q(\boldsymbol{\theta}) > 0$ whenever $\gamma(\boldsymbol{\theta}) > 0$, and vice versa; (ii) its normalizing constant Z_q is known. In addition, we assume that we have two effective Markov kernels K_γ and K_q in hand so that we can sample from $\gamma^\star(\boldsymbol{\theta})$ and $q^\star(\boldsymbol{\theta})$ sufficiently well. For a large number of statistical problems, efficient MCMC algorithms have been developed for sampling from the target posterior distribution $\gamma^\star(\boldsymbol{\theta})$. Furthermore, since we control the construction of the surrogate distribution $q^\star(\boldsymbol{\theta})$, we can typically make it easy to sample from without using MCMC. More details on constructing the surrogate distribution $q^\star(\boldsymbol{\theta})$ are deferred to Section 2.5. Condition (i) is not essential. In fact, by utilizing the MTM mode jumping strategies, our method can still work well in cases where $q^\star(\boldsymbol{\theta})$ and $\gamma^\star(\boldsymbol{\theta})$ are well separated. We postpone the discussion to Section 2.4.

Let $\pi(\boldsymbol{\theta}) = \gamma(\boldsymbol{\theta}) + q(\boldsymbol{\theta})$. The key to the WL mixture method is to recognize that the normalizing constants Z_γ and Z_q are proportional to the relative probability masses of the two components, $\gamma^\star(\boldsymbol{\theta})$ and $q^\star(\boldsymbol{\theta})$, in the mixture distribution $\pi(\boldsymbol{\theta})$. Therefore, we can directly apply the WL algorithm to estimate the ratio Z_γ/Z_q. To avoid numerical issues, we recommend to work on the logarithmic scale of the normalizing constants. The details are summarized in Algorithm 2. The γ and q in the brackets serve as indexes (the same role as i in Algorithm 1), and should not be misinterpreted as function arguments.

Algorithm 2 Normalizing constant estimation using the Wang-Landau algorithm

1. Algorithmic setup. Choose a decreasing positive sequence $\{\eta_t\}$ as the sequence of learning rate. Set $a_0 = 1$, $c \in (0, 1)$, $\xi_0(\gamma) = \xi_0(q) = 0$, and $\psi_0(\gamma) = \psi_0(q) = 1/2$. Set the total number of iterations to be S, and we exclude the results from the first b iterations in estimation.

2. At time $t = 0$: initialize $\boldsymbol{\theta}_0$ from some initial distribution, and sample a binary indicator I_0 with probability $\mathbb{P}(I_0 = 1) \propto \gamma(\boldsymbol{\theta}_0)$ and $\mathbb{P}(I_0 = 0) \propto q(\boldsymbol{\theta}_0)$.

3. For time $t \in [1 : S]$: given $(\boldsymbol{\theta}_{t-1}, I_{t-1})$, iterate between the following steps.

   (a) Sample $\boldsymbol{\theta}_t$ from $K_{t-1}(\boldsymbol{\theta}_{t-1}, \cdot)$, which is invariant to the the adaptive mixture distribution $\pi^\dagger_{t-1}(\boldsymbol{\theta})$ defined by $\{\psi_{t-1}(\gamma), \psi_{t-1}(q)\}$:

   $$\pi^\dagger_{t-1}(\boldsymbol{\theta}) \propto \frac{\gamma(\boldsymbol{\theta})}{\psi_{t-1}(\gamma)} + \frac{q(\boldsymbol{\theta})}{\psi_{t-1}(q)}. \tag{3}$$

   (b) Sample a binary indicator I_t with probability $\mathbb{P}(I_t = 1|\boldsymbol{\theta}_t) \propto \gamma(\boldsymbol{\theta}_t)/\psi_{t-1}(\gamma)$ and $\mathbb{P}(I_t = 0|\boldsymbol{\theta}_t) \propto q(\boldsymbol{\theta}_t)/\psi_{t-1}(q)$.

   (c) Update $\{\xi_t(\gamma), \xi_t(q)\}$ and $\{\psi_t(\gamma), \psi_t(q)\}$ as follows:

   $$\begin{aligned} \xi_t(\gamma) &\leftarrow \xi_{t-1}(\gamma) + \mathbb{1}(I_t = 1), \quad \psi_t(\gamma) \leftarrow \psi_{t-1}(\gamma)\left[1 + \eta_{a_t}\mathbb{1}(I_t = 1)\right], \\ \xi_t(q) &\leftarrow \xi_{t-1}(q) + \mathbb{1}(I_t = 0), \quad \psi_t(q) \leftarrow \psi_{t-1}(q)\left[1 + \eta_{a_t}\mathbb{1}(I_t = 0)\right]. \end{aligned} \tag{4}$$

   (d) Normalize $\{\psi_t(\gamma), \psi_t(q)\}$ to sum 1.

   (e) If the following condition is satisfied:

   $$\frac{\max\{\xi_t(\gamma),\ \xi_t(q)\}}{|\xi_t(\gamma) + \xi_t(q) - 1/2|} \le c/2,$$

   update $a_{t+1} = a_t + 1$ and reset $\xi_t(\gamma) = \xi_t(q) = 0$. Otherwise set $a_{t+1} = a_t$.

4. Output the estimators $\log \widehat{Z}_\gamma = \log \widehat{r} + \log Z_q$, where $\log \widehat{r} = \frac{1}{S-b}\sum_{t=b+1}^{S}[\log \xi_t(\gamma) - \log \xi_t(q)]$.

Successful application of the proposed method relies on an efficient strategy to sample from the adaptive mixture distribution $\pi^\dagger_t(\boldsymbol{\theta})$ (Step 3(a) in Algorithm 2). This point will become more explicit after we formulate the WL algorithm as a stochastic gradient descent algorithm following Dai and Liu (2019) (see Section 2.3), where the MCMC sampling in Step 3(a) is essentially estimating the gradient. In practice, it can be difficult to construct an effective Markov kernel K_t invariant to $\pi^\dagger_t(\boldsymbol{\theta})$. However, in the case where $q^\star(\boldsymbol{\theta})$ and $\gamma^\star(\boldsymbol{\theta})$ are

well mixed, we can avoid using K_t by performing a Gibbs sampling step (Diebolt and Robert, 1994) using K_γ and K_q, which are presumably two efficient Markov kernels leaving $\gamma^\star(\boldsymbol{\theta})$ and $q^\star(\boldsymbol{\theta})$ invariant, respectively. To be specific, if $I_{t-1} = 1$, we sample $\boldsymbol{\theta}_t$ from $K_\gamma(\boldsymbol{\theta}_{t-1}, \cdot)$; otherwise we sample $\boldsymbol{\theta}_t$ from $K_q(\boldsymbol{\theta}_{t-1}, \cdot)$.

Tunable parameter sequences $\{\xi_t(\gamma),\ \xi_t(q)\}$ are introduced to help check the flat histogram criterion, that is, whether the Markov chain has spent equal amount of time in each of the two components $\gamma^\star(\boldsymbol{\theta})$ and $q^\star(\boldsymbol{\theta})$. If this is approximately satisfied to the extent controlled by a threshold $c \in (0, 1)$ (see the condition in Step 3(e) in Algorithm 2), we decrease the learning rate and refresh $\xi_t(\gamma) = \xi_t(q) = 0$ so that it can start to monitor the next stage of the algorithm. Throughout all the numerical examples in the article, we set $c = 0.2$.

The WL mixture method naturally adapts to the missing data framework. The marginal likelihood of the observed-data can be formulated as follows:

$$L(\boldsymbol{y}_{\text{obs}}) = \int p(\boldsymbol{y}_{\text{obs}} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} = \int \int p(\boldsymbol{y}_{\text{obs}}, \boldsymbol{y}_{\text{mis}} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{y}_{\text{mis}} d\boldsymbol{\theta},$$

where $p(\boldsymbol{y}_{\text{obs}}, \boldsymbol{y}_{\text{mis}} \mid \boldsymbol{\theta})$ is the complete-data distribution, and $p(\boldsymbol{\theta})$ is the prior. In the simplest case where the integral $\int p(\boldsymbol{y}_{\text{obs}}, \boldsymbol{y}_{\text{mis}} \mid \boldsymbol{\theta}) d\boldsymbol{y}_{\text{mis}}$ can be analytically calculated, such as the finite mixture model in which $\boldsymbol{y}_{\text{mis}}$ are discrete, we can directly apply the WL mixture method to estimate the normalizing constant $L(\boldsymbol{y}_{\text{obs}})$. More generally, we can treat the missing data $\boldsymbol{y}_{\text{mis}}$ as parameters, and apply the WL mixture method to estimate the normalizing constant of the (unnormalized) complete-data posterior distribution $\gamma(\boldsymbol{\theta}, \boldsymbol{y}_{\text{mis}} \mid \boldsymbol{y}_{\text{obs}}) = p(\boldsymbol{y}_{\text{obs}}, \boldsymbol{y}_{\text{mis}} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})$.

2.3 Acceleration of the Wang-Landau mixture method

The efficiency of the WL mixture method can be further improved using the acceleration idea discussed in Dai and Liu (2019). Note that the update of the reweighting factor $\psi_t(\gamma)$ (Step 3(c) in Algorithm 2) can be approximated as follows:

$$\begin{aligned} \log \psi_t(\gamma) &= \log \psi_{t-1}(\gamma) + \log \left[1 + \eta_{a_t} \mathbb{1}\left(I_t = 1\right)\right] \\ &\approx \log \psi_{t-1}(\gamma) + \eta_{a_t} \mathbb{1}\left(I_t = 1\right) \\ &\approx \log \psi_{t-1}(\gamma) + \eta_{a_t} \mathbb{P}\left(I_t = 1\right), \end{aligned} \tag{5}$$

and the update of $\psi_t(q)$ can be approximated in the same way. In the last line of the above approximation, we replace $\mathbb{1}\left(I_t = 1\right)$ by its expectation taken with respect to the adaptive mixture distribution defined in (3).

Denote $u_t(\gamma) = \log \psi_t(\gamma)$ and $u_t(q) = \log \psi_t(q)$, then the approximation in (5) leads to the following updates, which are in the form of gradient descent:

$$\begin{aligned} u_t(\gamma) &= u_{t-1}(\gamma) + \eta_{a_t} \left(\mathbb{P}\left(I_t = 1\right) - 1/2\right), \\ u_t(q) &= u_{t-1}(q) + \eta_{a_t} \left(\mathbb{P}\left(I_t = 0\right) - 1/2\right). \end{aligned} \tag{6}$$

We add $-1/2$ because the (negative) gradient $\mathbb{P}\left(I_t = 1\right) - 1/2$ and $\mathbb{P}\left(I_t = 0\right) - 1/2$ should vanish after the algorithm has converged, and $\lim_{t\to\infty} \mathbb{P}\left(I_t = 1\right) = \lim_{t\to\infty} \mathbb{P}\left(I_t = 0\right) = 1/2$ because the chain is expected to spend equal amount of time in the two mixture components $\gamma^\star(\boldsymbol{\theta})$ and $q^\star(\boldsymbol{\theta})$ after the WL algorithm has converged. We note that adding a constant to $u_t(\gamma)$ and $u_t(q)$, or equivalently multiplying a constant to $\psi_t(\gamma)$ and $\psi_t(q)$, does not affect the WL algorithm because the multiplicative constant will be canceled out in Step 3(a) in Algorithm 2.

The objective function $f(u(\gamma), u(q))$ corresponding to the gradient descent updates in (6) can be derived as follows. By (3), $\mathbb{P}\left(I_t = 1\right) \propto Z_\gamma / \psi_{t-1}(\gamma)$, and $\mathbb{P}\left(I_t = 0\right) \propto Z_q / \psi_{t-1}(q)$, thus the gradient of $f(u(\gamma), u(q))$ is:

$$\begin{cases} \dfrac{\partial f}{\partial u(\gamma)} = -\dfrac{Z_\gamma \exp(-u(\gamma))}{Z_\gamma \exp(-u(\gamma)) + Z_q \exp(-u(q))} + \dfrac{1}{2}, \\ \dfrac{\partial f}{\partial u(q)} = -\dfrac{Z_q \exp(-u(q))}{Z_\gamma \exp(-u(\gamma)) + Z_q \exp(-u(q))} + \dfrac{1}{2}. \end{cases} \tag{7}$$

Solving this simple partial gradient system leads to the form of the objective function:

$$f(u(\gamma), u(q)) = \log \left[Z_\gamma \exp\left(-u(\gamma)\right) + Z_q \exp\left(-u(q)\right)\right] + \frac{1}{2}\left[u(\gamma) + u(q)\right]. \tag{8}$$

The WL algorithm is essentially a stochastic gradient descent algorithm minimizing the objective function $f(u(\gamma), u(q))$, of which the gradient involves the unknown quantity Z_γ/Z_q, and is estimated using a MCMC step as Step 3(a) and 3(b) in Algorithm 2. It is not difficult to see that $f(u(\gamma), u(q))$ is smooth and convex, and has a unique solution $(u^\star(\gamma), u^\star(q))$, up to an additive constant, satisfying $u^\star(\gamma) - u^\star(q) = \log Z_\gamma/Z_q$.

Once we have the optimization perspective, various acceleration tools can be employed to improve the efficiency of the WL mixture method. One simple tool we find useful is the momentum method, which exponentially accumulates a momentum vector to amplify the persistent gradient across iterations, thus reducing the oscillation caused by the noise in the gradient estimate. To apply the acceleration method, we only need to modify Step 3(c) in Algorithm 2 as follows:

3 (c′) (Momentum accelerated WL updates)

(i) Update the momentum vector:

$$m_t(\gamma) \leftarrow \beta m_{t-1}(\gamma) - \eta_{a_t} \mathbb{1}(I_t = 1), \qquad m_t(q) \leftarrow \beta m_{t-1}(q) - \eta_{a_t} \mathbb{1}(I_t = 0).$$

(ii) Update the reweighting vector:

$$\log \psi_t(\gamma) \leftarrow \log \psi_{t-1}(\gamma) - m_t(\gamma), \qquad \log \psi_t(q) \leftarrow \log \psi_{t-1}(q) - m_t(q).$$

(iii) Update $\{\xi_t(\gamma), \xi_t(q)\}$ as in Step 3(c) in Algorithm 2.

The momentum vector can be simply initialized as $m_0(\gamma) = m_0(q) = 0$, and β is commonly set to be 0.9 or higher, which calibrates the fraction of the accumulated past gradients that we want to incorporate into the current update. Numerical illustrations of the accelerated WL mixture method is given in Figure 2 on two statistical models, the Log-Gaussian Cox process and the Bayesian Lasso.

2.4 Global jump via Multiple-try Metropolis

One way to handle the possible separation between the target distribution and the surrogate distribution is using the Multiple-try Metropolis (MTM) (Liu et al., 2000). Given a target distribution $\pi(\boldsymbol{x})$ defined on $\mathbb{R}^d$ and a proposal transition function $T(\boldsymbol{x}, \boldsymbol{y})$, MTM aims at *biasing* the local sampling with a proper weight function $w(\boldsymbol{x}, \boldsymbol{y})$:

$$w(\boldsymbol{x}, \boldsymbol{y}) = \pi(\boldsymbol{x}) T(\boldsymbol{x}, \boldsymbol{y}) \lambda(\boldsymbol{x}, \boldsymbol{y}), \tag{9}$$

where $\lambda(\boldsymbol{x}, \boldsymbol{y})$ is a user-chosen nonnegative symmetric function. Assuming the current state of the Markov chain is $\boldsymbol{x}_t$, one step of the algorithm is detailed in Algorithm 3.

Algorithm 3 Multiple-try Metropolis (Liu et al., 2000)

1. Sample $\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(m)}$ i.i.d from $T(\boldsymbol{x}_t, \cdot)$, and compute the weight function $w(\boldsymbol{y}^{(j)}, \boldsymbol{x})$.
2. Sample $\boldsymbol{y}$ from $\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(m)}$ with probability proportional to $w(\boldsymbol{y}^{(j)}, \boldsymbol{x})$.
3. Given $\boldsymbol{y}$, sample $\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(m-1)}$ i.i.d from $T(\boldsymbol{y}, \cdot)$, and set $\boldsymbol{x}^{(m)} = \boldsymbol{x}_t$.
4. Accept $\boldsymbol{y}$ with probability:

$$\alpha = \min\left\{1, \quad \frac{w(\boldsymbol{y}^{(1)}, \boldsymbol{x}) + \cdots + w(\boldsymbol{y}^{(m)}, \boldsymbol{x})}{w(\boldsymbol{x}^{(1)}, \boldsymbol{y}) + \cdots + w(\boldsymbol{x}^{(m)}, \boldsymbol{y})}\right\}. \tag{10}$$

A special choice of λ is $\lambda(\boldsymbol{x}, \boldsymbol{y}) = [T(\boldsymbol{x}, \boldsymbol{y}) + T(\boldsymbol{y}, \boldsymbol{x})]^{-1}$. In the case where $T(\boldsymbol{x}, \boldsymbol{y})$ is a symmetric proposal, the corresponding acceptance probability simplifies to:

$$\alpha = \min\left\{1, \quad \frac{\pi(\boldsymbol{y}^{(1)}) + \cdots + \pi(\boldsymbol{y}^{(m)})}{\pi(\boldsymbol{x}^{(1)}) + \cdots + \pi(\boldsymbol{x}^{(m)})}\right\}. \tag{11}$$

This special case is referred to as MTM (II) in Liu et al. (2000). MTM is particularly useful when it is combined with the directional sampling algorithm. For instance, if we know a desirable jumping direction, we can use MTM to explore a wide range along it. Let $\boldsymbol{e}$ denote the jumping direction. For the simple case

where $\boldsymbol{e}$ is fixed throughout the algorithm and independent of the current state $\boldsymbol{x}_t$, we outline the main steps in Algorithm 4. More generally, we can choose the jumping direction $\boldsymbol{e}$ based on the current state $\boldsymbol{x}_t$. Some detailed discussion on a special form of this adaptive strategy can be found in Section 4.3.

Algorithm 4 Multiple-try Metropolis combined with the directional sampling algorithm

1. Sample $r^{(1)}, \cdots, r^{(m)}$ from some user chosen distribution $p(r)$, and let $\boldsymbol{y}^{(j)} = \boldsymbol{x}_t + r^{(j)} \cdot \boldsymbol{e}$. Compute the target density $\pi(\boldsymbol{y}^{(j)})$.
2. Sample $\boldsymbol{y}$ from $\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(m)}$ with probability proportional to $\pi(\boldsymbol{y}^{(j)})$. Set $\boldsymbol{x}^{(j)} = \boldsymbol{y} - r^{(j)} \cdot \boldsymbol{e}$.
3. Accept $\boldsymbol{x}_{t+1} = \boldsymbol{y}$ with probability:

$$\alpha = \min\left\{1, \quad \frac{\pi(\boldsymbol{y}^{(1)}) + \cdots + \pi(\boldsymbol{y}^{(m)})}{\pi(\boldsymbol{x}^{(1)}) + \cdots + \pi(\boldsymbol{x}^{(m)})}\right\}. \tag{12}$$

Assuming that we are equipped with an efficient kernel K_γ, some pre-MCMC runs should help us pin down informative jumping directions, such as the directions connecting the modes of the target distribution and the surrogate distribution. Thus, we can replace Step 3(a) in Algorithm 2 by randomly alternating between MTM global jumps following Algorithm 4, and local moves around each mixture component using K_γ and K_q.

In practice, if possible, we recommend the users to place the surrogate distribution close to the target distribution to potentially avoid the implementation of MTM. On the other hand, using MTM, the WL mixture method can handle the case where the target distribution and the surrogate distribution has little overlaps, as long as we have an effective jumping strategy to sample from $\pi_t^\dagger(\boldsymbol{\theta})$ defined in (3). Indeed, the standard WL algorithm, which partitions the sample space along the energy function, is able to handle the non-overlapping case since the subregions defined by the partition are mutually exclusive. More detailed discussion can be found in Section 3.

2.5 Constructing the surrogate distribution

In principle, any posterior approximation with a known normalizing constant, such as the Laplace approximation or the variational approximation, can be used as the surrogate distribution. We can also use MCMC methods to obtain posterior samples, and fit some parametric distributions on them. In this section, we describe how to construct a surrogate distribution $q(\boldsymbol{\theta})$ using the variational approximation (Jordan et al., 1999; Blei et al., 2017). The variational approach enjoys two main advantages. First, it is computationally efficient and does not require MCMC sampling to explore $\gamma^\star(\boldsymbol{\theta})$. Second, it provides a reasonable approximation to $\gamma^\star(\boldsymbol{\theta})$ in a wide class of statistical models (Wainwright and Jordan, 2008).

The variational approximation aims at finding the closest distribution $q^\star(\boldsymbol{\theta})$ to $\gamma^\star(\boldsymbol{\theta})$ in the KL divergence within a particular class of distributions Q, that is,

$$q^\star(\boldsymbol{\theta}) = \underset{p(\boldsymbol{\theta}) \in Q}{\arg\min}\, \mathrm{KL}\left(p(\boldsymbol{\theta}) || \gamma^\star(\boldsymbol{\theta})\right). \tag{13}$$

$\mathrm{KL}\left(p(\boldsymbol{\theta})||\gamma^\star(\boldsymbol{\theta})\right)$ is not computable due to the unknown normalizing constant of Z_γ. However, the optimization problem (13) can be equivalently formulated as follows:

$$q^\star(\boldsymbol{\theta}) = \underset{p(\boldsymbol{\theta}) \in Q}{\arg\max}\, \mathrm{ELBO}(p) = \underset{p(\boldsymbol{\theta}) \in Q}{\arg\max} \left\{\mathbb{E}_p\left[\log \gamma(\boldsymbol{\theta})\right] - \mathbb{E}_p\left[\log p(\boldsymbol{\theta})\right]\right\}. \tag{14}$$

ELBO refers to the *evidence lower bound* of $\log Z_\gamma$ since:

$$\log Z_\gamma = \mathrm{KL}\left(q^\star(\boldsymbol{\theta})||\gamma^\star(\boldsymbol{\theta})\right) + \mathrm{ELBO}(q^\star) \geq \mathrm{ELBO}(q^\star). \tag{15}$$

We note that the EM algorithm (Dempster et al., 1977) can also be formulated as a two-step iterative algorithm that maximizes the ELBO with respect to the density $p(\boldsymbol{\theta})$ and the relevant model parameters (Tzikas et al., 2008). Before solving the optimization problem (14), we need to specify the variational family Q. A commonly considered class of distributions Q is the mean-field variational family, which assumes that

$q^\star(\boldsymbol{\theta})$ is a product of univariate distributions, that is, $q^\star(\boldsymbol{\theta}) = \prod_{j=1}^d q_j^\star(\theta_j)$. We assume that $q_j^\star(\theta_j)$ belongs to some parametric family Q_j whose probability density function can be evaluated exactly.

To solve the optimization problem (14), we can use the *coordinate ascent variational inference* (CAVI) algorithm (Bishop, 2006). CAVI, detailed in Algorithm 5, iteratively maximizes the ELBO in a coordinate-wise fashion. We note that optimizing (17) can be further simplified in some conjugate cases since for each $j \in [1:d]$, conditioning on all the other components $q_i^\star(\theta_i),\ i \neq j$, ELBO$(q^\star)$ can be rewritten as:

$$\text{ELBO}(q^\star) = -\text{KL}\left(q_j^\star(\theta_j) || q_j^{\text{opt}}(\theta_j)\right) + \text{constant}, \tag{16}$$

where $q_j^{\text{opt}}(\theta_j) \propto \exp\left[\mathbb{E}_{-j}\left(\log \gamma(\theta_j, \boldsymbol{\theta}_{-j})\right)\right]$. If $q_j^{\text{opt}}(\theta_j) \in Q_j$ (conjugacy), we know that the optimal $q_j^\star(\theta_j)$ that maximizes ELBO$(q^\star)$ in (17) is $q_j^{\text{opt}}(\theta_j)$ since the KL divergence is non-negative.

Algorithm 5 Coordinate ascent variational inference (CAVI) (Blei et al., 2017)

1. Initialize each $q_j^\star(\theta_j) \in Q_j$ for $j \in [1:d]$.
2. For each $j \in [1:d]$, fix all the other components $q_i^\star(\theta_i),\ i \neq j$, update $q_j^\star(\theta_j)$ with the following optimal univariate distribution that maximizes

$$\text{ELBO}(q^\star) = \mathbb{E}_j\left[\mathbb{E}_{-j}\left(\log \gamma(\theta_j, \boldsymbol{\theta}_{-j})\right)\right] - \mathbb{E}_j\left[\log q_j^\star(\theta_j)\right] + \text{constant}. \tag{17}$$

3. Calculate ELBO$(q^\star)$ where $q^\star(\boldsymbol{\theta}) = \prod_{j=1}^d q_j^\star(\theta_j)$. If ELBO hasn't converged, go back to step 2. Otherwise output $q^\star(\boldsymbol{\theta})$.

3 Comparison to Other Methods

3.1 Importance sampling and sequential Monte Carlo

It is known that the performance of importance sampling is determined by how closely the proposal distribution tracks the target distribution. In a good importance sampler, high probability regions of the proposal and target distributions typically overlap substantially, and the proposal distribution typically has a heavier tail than the target one. Otherwise, the variance of the importance sampling estimator can be unacceptably large and the resulting estimation can be misleading. If the dimension of the problem is high, it is generally hard to construct an appropriate proposal distribution.

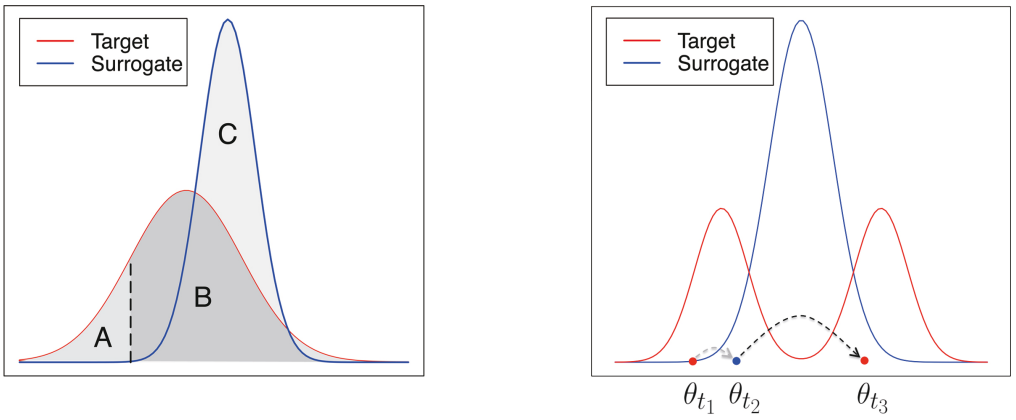
Figure 1a provides a cartoon illustration of the setting that the surrogate distribution has a smaller domain (and thinner tail) compared to the target one. Importance sampling only estimates the normalizing constant of the target distribution restricted on the region B, thus produces an underestimated normalizing constant. This phenomenon is illustrated on two realistic examples in Section 5, the Log-Gaussian Cox process and the Bayesian Lasso. In contrast, the WL mixture method has approximately equal chance to explore the whole high-density regions of the surrogate distribution and the target distribution, respectively, assuming that K_γ and K_q both mix well. Therefore, we expect the WL mixture method to produce more accurate, rather than underestimated, normalizing constant estimates.

Sequential Monte Carlo (SMC), built based on importance sampling, resolves the aforementioned difficulty in handling a multidimensional target distribution by sequentially sampling one variable conditioning on previously sampled variables (Liu et al., 2001; Liu, 2008). To apply SMC in our setting, we first need a decomposition, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$, so as to construct a sequence of (unnormalized) auxiliary distributions $\eta_1(\theta_1), \eta_2(\theta_1, \theta_2), \ldots, \eta_p(\boldsymbol{\theta}) = \gamma(\boldsymbol{\theta})$. We then initialize by drawing n samples from a (normalized) proposal distribution $q_1(\theta_1)$ for the first component: $\theta_1^{(1)}, \cdots, \theta_1^{(n)}$, and attach to each with weight $w_1^{(i)} = \eta_1(\theta_1^{(i)})/q_1(\theta_1^{(i)})$. We record $W_1 = \frac{1}{n}\sum_{i=1}^n w_1^{(i)}$, which serves as an estimate of the normalizing constant of $\eta_1(\theta_1)$. In the next step, we can either resample the obtained "particles" $\{(\theta_1^{(i)}, w_1^{(i)}), i = 1, \ldots, n\}$ with probability proportional to, say $(w_1^{(i)})^\alpha$ with $\alpha \in [0,1]$, and modify the new weights to $(w_1^{(i)})^{1-\alpha}$, or proceed directly to sample $\theta_2^{(i)}$ from a (normalized) user-chosen sampling distribution $q_2(\theta_2 \mid \theta_1^{(i)})$, and compute the updated weights

$$w_2^{(i)} = w_1^{(i)} \times \frac{\eta_2(\theta_1^{(i)}, \theta_2^{(i)})}{\eta_1(\theta_1^{(i)}) q_2(\theta_2^{(i)} \mid \theta_1^{(i)})}.$$

Similarly, $W_2 = \frac{1}{n}\sum_{i=1}^{n} w_2^{(i)}$ is an estimate of the normalizing constant of $\eta_2(\theta_1, \theta_2)$. This sequential update is carried out up to step p, and the final average weight W_p is an estimate of the normalizing constant of $\eta_p(\boldsymbol{\theta}) = \gamma(\boldsymbol{\theta})$. It is easy to see that W_p is unbiased, if no resampling is involved, and is always consistent for estimating Z_γ (Del Moral, 2004). We can generalize the above SMC construct to cases where no dimensional changes are involved, that is, $\eta_1, \ldots, \eta_p$ are all defined on the full space of $\boldsymbol{\theta}$. In this case SMC looks very similar to bridge sampling and path sampling (Gelman and Meng, 1994).

It is generally nontrivial to design a good sequence of auxiliary and sampling distributions. In Bayesian inference problems, a common choice of the auxiliary distributions is the posterior distribution with partial data, and a common choice for the sampling distributions is some form of prior/posterior predictive distributions. But even in those cases, the efficiency of SMC can still be problematic (Doucet et al., 2000). The WL mixture method is generally much easier to implement and requires less tuning. In the example Log-Gaussian Cox process (Section 5.1), instead of comparing the WL mixture method to this somewhat too generic SMC, we compare it to a generalized form of SMC detailed in Algorithm 7.



(a) Demonstration of the proposed WL method.

(b) Demonstration of mode jumping.

Figure 1

3.2 Bridge sampling

Bridge sampling provides an efficient way of utilizing importance samples (Meng and Wong, 1996). Given the (unnormalized) target distribution $\gamma(\boldsymbol{\theta})$ and proposal distribution $q(\boldsymbol{\theta})$, bridge sampling inserts a bridge $\gamma_{1/2}(\boldsymbol{\theta})$ between $\gamma(\boldsymbol{\theta})$ and $q(\boldsymbol{\theta})$, and estimates the ratio Z_γ/Z_q based on the following identity:

$$r = \frac{Z_\gamma}{Z_q} = \frac{\mathbb{E}_q\left[\gamma_{1/2}(\boldsymbol{\theta})/q(\boldsymbol{\theta})\right]}{\mathbb{E}_\gamma\left[\gamma_{1/2}(\boldsymbol{\theta})/\gamma(\boldsymbol{\theta})\right]}. \tag{18}$$

The corresponding bridge sampling estimator is

$$\widehat{r} = \frac{(1/n_q)\sum_{i=0}^{n_q}\gamma_{1/2}(\boldsymbol{\theta}_{qi})/q(\boldsymbol{\theta}_{qi})}{(1/n_\gamma)\sum_{i=0}^{n_\gamma}\gamma_{1/2}(\boldsymbol{\theta}_{\gamma i})/\gamma(\boldsymbol{\theta}_{\gamma i})}, \tag{19}$$

where $\boldsymbol{\theta}_{q1}, \cdots, \boldsymbol{\theta}_{qn_q}$ are n_q samples from the proposal distribution $q^\star(\boldsymbol{\theta})$, and $\boldsymbol{\theta}_{\gamma 1}, \cdots, \boldsymbol{\theta}_{\gamma n_\gamma}$ are n_γ samples from the target distribution $\gamma^\star(\boldsymbol{\theta})$.

The goal of introducing the bridge distribution $\gamma^\star_{1/2}(\boldsymbol{\theta})$ is to get more connections between $q^\star(\boldsymbol{\theta})$ and $\gamma^\star(\boldsymbol{\theta})$ through the bridge $\gamma^\star_{1/2}(\boldsymbol{\theta})$. In addition, since bridge sampling also utilizes samples from the target distribution $\gamma^\star(\boldsymbol{\theta})$, it helps resolve the issue of underestimating the normalizing constant illustrated in Figure 1a and discussed in Section 3.1. However, the efficiency of bridge sampling is still sensitive to the "distance" between $q^\star(\boldsymbol{\theta})$ and $\gamma^\star(\boldsymbol{\theta})$. For simplicity, let us assume $n_q = n_\gamma = n$, and consider the optimal bridge $\gamma_{\text{opt}}(\boldsymbol{\theta}) = (n_q q^\star(\boldsymbol{\theta})^{-1} + n_\gamma \gamma^\star(\boldsymbol{\theta})^{-1})^{-1}$ that minimizes the asymptotic variance of $\log \widehat{r}$ under the assumption

that all the samples are independent draws. The corresponding optimal asymptotic variance is:

$$V_{\text{opt}} = \frac{2}{n}\left[\left(\int \frac{2q^\star(\boldsymbol{\theta})\gamma^\star(\boldsymbol{\theta})}{q^\star(\boldsymbol{\theta})+\gamma^\star(\boldsymbol{\theta})}d\boldsymbol{\theta}\right)^{-1} - 1\right] \geq \frac{2}{n}\left[\left(\int 2\min\{q^\star(\boldsymbol{\theta}), \gamma^\star(\boldsymbol{\theta})\}d\boldsymbol{\theta}\right)^{-1} - 1\right]. \tag{20}$$

We see that the lower bound increases if we push the proposal distribution $q^\star(\boldsymbol{\theta})$ and the target distribution $\gamma^\star(\boldsymbol{\theta})$ further apart. The separation can be overcome by creating multiple bridges between the proposal distribution and the target distribution.

In contrast, with the help of the MTM directional sampling discussed in Section 2.4, the WL mixture method can potentially overcome this separation issue. We empirically illustrate this on a 20-dimensional multivariate normal distribution. The target distribution is $N(\mathbf{0}, I_{20})$, and the proposal (surrogate) distribution is $N(\mu \times \mathbf{1}_{20}, I_{20})$ with $\mu = 1, 2, 3, 4, 5$. The target distribution has been normalized so that the true log normalizing constant is 0. The fixed jumping direction is $\boldsymbol{e} = \pm(\mu \times \mathbf{1}_{20})$, and we use 8 tries in each multiple-try iteration. For simplicity, we substitute the local MCMC moves around the two mixture components by directly sampling from either the target distribution or the surrogate distribution. We set $n_\gamma = n_q = 5,000$, and run 5,000 iterations for the WL mixture method. The results are summarized in Table 1.

Method	$\mu = 1$	$\mu = 2$	$\mu = 3$	$\mu = 4$	$\mu = 5$
WL	0.000 (0.047)	0.005 (0.035)	0.004 (0.040)	-0.001 (0.041)	0.013 (0.049)
BS	-0.004 (0.109)	0.209 (3.152)	-0.348 (5.426)	-1.273 (6.932)	1.595 (7.902)

Table 1: Comparisons of the WL mixture method with bridge sampling for estimating the logarithm of the integral of the multivariate normal density, $\log Z_{\text{mvn}}$, which is exactly 0 in all cases. The reported values are empirical means and standard deviations (in the bracket) of estimates based on 10 independent runs.

We see that the WL mixture method has robust performances for different μ's, whereas bridge sampling performs worse as the target and the surrogate/proposal distributions become more and more separated. We note that the comparison is not entirely fair because we pre-locate the mode of the target distribution for MTM. Our point is that the WL mixture method should be classified as a MCMC-based method and behaves very differently from bridge sampling and other importance sampling based methods. The performance of the WL mixture method crucially relies on an efficient strategy to sample from the adaptive mixture distribution $\pi_t^\dagger(\boldsymbol{\theta})$ defined in (3), rather than the amount of overlaps between the target and the surrogate distributions.

3.3 Chib's method

The method proposed by Chib (1995) is very effective for estimating normalizing constants for a class of Bayesian models and has been widely adopted. For any $\boldsymbol{\theta}^\star$ such that $p(\boldsymbol{\theta}^* \mid \boldsymbol{y}) > 0$, we have

$$\log Z_\gamma = \log p(\boldsymbol{y} \mid \boldsymbol{\theta}^\star) + \log p(\boldsymbol{\theta}^\star) - \log \gamma^\star(\boldsymbol{\theta}^\star \mid \boldsymbol{y}), \tag{21}$$

where $p(\boldsymbol{y} \mid \boldsymbol{\theta}^\star)$ and $p(\boldsymbol{\theta}^\star)$ are the normalized model and prior densities at $\boldsymbol{\theta}^\star$, respectively. Consequently, if we can estimate well the normalized posterior density at $\boldsymbol{\theta}^\star$, that is, $\gamma^\star(\boldsymbol{\theta}^\star \mid \boldsymbol{y})$, we have an estimate of the normalizing constant Z_γ.

Chib (1995) shows that this is feasible using Gibbs outputs. For example, suppose $\boldsymbol{\theta}$ can be decomposed into two blocks, $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$, and we are equipped with an efficient Gibbs sampler targeting the posterior distribution $\gamma(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \mid \boldsymbol{y})$, which iteratively samples from the two conditional distributions $\gamma^\star(\boldsymbol{\theta}_1 \mid \boldsymbol{\theta}_2, \boldsymbol{y})$ and $\gamma^\star(\boldsymbol{\theta}_2 \mid \boldsymbol{\theta}_1, \boldsymbol{y})$. We further assume that we can evaluate the two conditional distributions exactly. With the Gibbs outputs $\{(\boldsymbol{\theta}_1^{(1)}, \boldsymbol{\theta}_2^{(1)}), \cdots, (\boldsymbol{\theta}_1^{(n)}, \boldsymbol{\theta}_2^{(n)})\}$, we can estimate the normalized posterior density at $\boldsymbol{\theta}^\star$ as below:

$$\widehat{\gamma}^\star(\boldsymbol{\theta}^\star \mid \boldsymbol{y}) = \widehat{\gamma}^\star(\boldsymbol{\theta}_1^\star \mid \boldsymbol{y})\gamma^\star(\boldsymbol{\theta}_2^\star \mid \boldsymbol{\theta}_1^\star, \boldsymbol{y}) = \left[\frac{1}{n}\sum_{i=1}^{n}\gamma^\star(\boldsymbol{\theta}_1^\star \mid \boldsymbol{\theta}_2^{(i)}, \boldsymbol{y})\right]\gamma^\star(\boldsymbol{\theta}_2^\star \mid \boldsymbol{\theta}_1^\star, \boldsymbol{y}), \tag{22}$$

which utilizes the fact that $\gamma^\star(\boldsymbol{\theta}_1 \mid \boldsymbol{y}) = \int \gamma^\star(\boldsymbol{\theta}_1 \mid \boldsymbol{\theta}_2, \boldsymbol{y})\gamma^\star(\boldsymbol{\theta}_2 \mid \boldsymbol{y})d\boldsymbol{\theta}_2$. For a better statistical efficiency, it is recommended to select $\boldsymbol{\theta}^\star$ close to the posterior mode. The above scheme can be generalized to cases in which $\boldsymbol{\theta}$ is decomposed into an arbitrary number of blocks, and also cases with missing data (Chib, 1995).

We see that Chib's method is particularly useful and easy to implement when we have an efficient Gibbs sampler with all the conditional distributions being tractable. In Section 5.2, we compare Chib's method

and the WL mixture method on the Bayesian Lasso example, in which we indeed have a closed form Gibbs sampler. The performances of the two methods are comparable. Chib and Jeliazkov (2001) extended the method to the setting with intractable full conditional densities, but its applicability can still be limited if we encounter other types of MCMC algorithms such as Hamiltonian Monte Carlo (HMC), Metropolis-adjusted Langevin algorithms (MALA), etc. In contrast, a major advantage of the WL mixture method is that it can be built on any type of MCMC samplers, and is reasonably easy to implement.

4 Extensions

4.1 Parallel Wang-Landau (PWL) algorithm

The main idea of PWL is to introduce a sequence of intermediate distributions between the target distribution $\gamma^\star(\boldsymbol{\theta})$ and the surrogate distribution $q^\star(\boldsymbol{\theta})$, and to estimate many ratios of normalizing constants in parallel. This method is particularly useful when a good surrogate distribution $q^\star(\boldsymbol{\theta})$ is not easy to construct. The idea of constructing auxiliary distributions has been used in bridge sampling (Gelman and Meng, 1994), annealed importance sampling (Neal, 2001), sequential importance sampling (Hammersley and Morton, 1954; Rosenbluth and Rosenbluth, 1955; Kong et al., 1994), and the general SMC algorithms (Liu and Chen, 1998; Chopin, 2002). However, these earlier methods typically cannot be easily parallelized.

We denote the sequence of (unnormalized) intermediate distributions as $\{\eta_t(\boldsymbol{\theta})\}_{t=0}^T$ where $\eta_0(\boldsymbol{\theta}) = q(\boldsymbol{\theta})$ and $\eta_T(\boldsymbol{\theta}) = \gamma(\boldsymbol{\theta})$. The corresponding normalizing constant of $\eta_t(\boldsymbol{\theta})$ is denoted as Z_t. For example, suppose we have n data points $\boldsymbol{y} = (y_1, \cdots, y_n)$ with the likelihood function and the prior specified as $L(\boldsymbol{\theta} \mid \boldsymbol{y}) = p(\boldsymbol{y} \mid \boldsymbol{\theta})$ and $p(\boldsymbol{\theta})$, respectively. We can choose the surrogate distribution $q^\star(\boldsymbol{\theta})$ as the prior distribution $p(\boldsymbol{\theta})$, which is often very far from the target posterior distribution, and set the intermediate distributions as

$$\eta_t(\boldsymbol{\theta}) = L(\boldsymbol{\theta} \mid y_1, \cdots, y_{n_t}) p(\boldsymbol{\theta}), \tag{23}$$

where $0 = n_0 < n_1 < \cdots < n_T = n$. Assuming that we have a good sampler to sample from the full posterior distribution $\gamma^\star(\boldsymbol{\theta})$, it should be easy to construct an efficient sampler to sample from the intermediate distributions $\eta_t(\boldsymbol{\theta})$. Another possibility of the intermediate distributions can be the power-function sequences (as in parallel tempering of Geyer (1991)) such as $\eta_t(\boldsymbol{\theta}) = [q(\boldsymbol{\theta})]^{1-\alpha_t}[\gamma(\boldsymbol{\theta})]^{\alpha_t}$, where $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_T = 1$.

For each pair of neighboring distributions $\eta_t(\boldsymbol{\theta})$ and $\eta_{t-1}(\boldsymbol{\theta})$, we estimate $\log Z_t/Z_{t-1}$ using the WL mixture method. Denoting the estimates as $\log \widehat{r}_t$ for $t \in [1:T]$, we then estimate $\log Z_\gamma$ as:

$$\log \widehat{Z}_\gamma = \sum_{t=1}^{T} \log \widehat{r}_t + \log Z_q. \tag{24}$$

Since the estimation of $\log Z_t/Z_{t-1}$ for $t = 1, \ldots, T$ can be carried out in parallel, the PWL method can be computationally favorable if users have parallel computing resources. We note that the PWL is different from the parallel Wang-Landau algorithm considered in Bornn et al. (2013), which aims at accelerating the convergence of the WL algorithm.

4.2 Handling multimodality

The proposed method can be modified to detect the potential multimodality of the target distribution. As shown in Figure 1b, suppose the current state of the Markov chain is $\boldsymbol{\theta}_{t_1}$. In Step 3(b) in Algorithm 2, $\boldsymbol{\theta}_{t_1}$ is considered to be more likely coming from $\gamma^\star(\boldsymbol{\theta})$ instead of $q^\star(\boldsymbol{\theta})$ (thus the color of the point is red). Consequently, the WL mixture method will keep downweighting the mixture component $\gamma^\star(\boldsymbol{\theta})$ by increasing its reweighting factor $\psi_{t_1}(\gamma)$. In the meantime, we move $\boldsymbol{\theta}_{t_1}$ using the Markov kernel K_γ, thus the chain still stays around the same local mode. At some point t_2, the mixture component $\gamma^\star(\boldsymbol{\theta})$ has been downweighted enough, that is, the reweighting factor $\psi_{t_2}(\gamma)$ is large enough, so that $\boldsymbol{\theta}_{t_2}$ is considered to be more likely coming from $q^\star(\boldsymbol{\theta})$ instead of $\gamma^\star(\boldsymbol{\theta})$ (thus the color of the point becomes blue). Then in the next step we will move $\boldsymbol{\theta}_{t_2}$ according to the Markov kernel K_q invariant to $q^\star(\boldsymbol{\theta})$. Since in most cases we can directly sample from the surrogate distribution $q^\star(\boldsymbol{\theta})$, this step is potentially a global move compared to the previous local move using the Markov kernel K_γ. In Figure 1b, $\boldsymbol{\theta}_{t_2}$ jumps to $\boldsymbol{\theta}_{t_3}$, which is close to a different local mode of $\gamma^\star(\boldsymbol{\theta})$.

To estimate $\mathbb{E}_{\gamma^\star}[h]$ for some function h using the outputs of the WL mixture method, we need to weight each sample differently. At iteration t, $\boldsymbol{\theta}_t$ is approximately a sample from the adaptive mixture distribution $\pi^\dagger_{t-1}(\boldsymbol{\theta})$ defined in (3). Thus, we need to assign to $\boldsymbol{\theta}_t$ a weight $w_t \propto \gamma(\boldsymbol{\theta}_t)/\pi^\dagger_{t-1}(\boldsymbol{\theta}_t)$. Then, we have the estimate $\sum_{t=0}^S w_t h(\boldsymbol{\theta}_t)/\sum_{t=0}^S w_t$, where S is the total number of iterations we run.

4.3 Multiple-try reversible jump MCMC

Instead of estimating the normalizing constant of each model, we can also perform model comparison for a sequence of models $\{\mathcal{M}_k\}$ by incorporating the model index into the full posterior distribution as in (1), and sampling the model indicator and model parameters jointly. Since the posterior distribution $p(\boldsymbol{\theta}_k, \mathcal{M}_k \mid \boldsymbol{y})$ is potentially trans-dimensional, we need a reversible jump MCMC algorithm (RJMCMC) (Green, 1995). We here propose a MTM-based reversible jump algorithm (MTM-RJMCMC), which is most useful if $p(\boldsymbol{\theta}_k \mid \boldsymbol{y}, \mathcal{M}_k)$ is approximately unimodal for each model $\mathcal{M}_k$, and we can estimate the mode $\widehat{\boldsymbol{\theta}}_k$ well before running the algorithm.

For $i \neq j$, suppose we want to move from model $\mathcal{M}_i$ to model $\mathcal{M}_j$. We wish to directly jump towards the mode of $p(\boldsymbol{\theta}_j \mid \boldsymbol{y}, \mathcal{M}_j)$ using the MTM directional sampling, so that the acceptance probability can be much higher than other generic jumping mechanisms. Since $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ are potentially in different dimensions, we first match the dimensionality of $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ by introducing auxiliary parameters $\boldsymbol{u} = (u_1, \cdots, u_{d_j}) \in \mathbb{R}^{d_j}$ and $\boldsymbol{v} = (v_1, \cdots, v_{d_i}) \in \mathbb{R}^{d_i}$ so that the dimension and domain of $(\boldsymbol{\theta}_i, \boldsymbol{u})$ matches those of $(\boldsymbol{v}, \boldsymbol{\theta}_j)$. We note that this is just one principled way to match the parameter spaces. For specific problems, more efficient designs may exist and should be considered.

We define the augmented posterior distributions as:

$$p_i(\boldsymbol{\theta}_i, \boldsymbol{u}, \mathcal{M}_i \mid \boldsymbol{y}) = p(\boldsymbol{\theta}_i \mid \boldsymbol{y}, \mathcal{M}_i) q_i(\boldsymbol{u}) p(\mathcal{M}_i), \quad p_j(\boldsymbol{v}, \boldsymbol{\theta}_j, \mathcal{M}_j \mid \boldsymbol{y}) = p(\boldsymbol{\theta}_j \mid \boldsymbol{y}, \mathcal{M}_j) q_j(\boldsymbol{v}) p(\mathcal{M}_j), \tag{25}$$

where $q_i(\boldsymbol{u})$ and $q_j(\boldsymbol{v})$ are user-chosen unimodal distributions with modes denoted as $\widehat{\boldsymbol{u}}$ and $\widehat{\boldsymbol{v}}$. The above construction implies that $\boldsymbol{\theta}_i \perp \boldsymbol{u}$ and $\boldsymbol{\theta}_j \perp \boldsymbol{v}$. In general, users can consider introducing dependence structures between $\boldsymbol{\theta}_i, \boldsymbol{u}$ and $\boldsymbol{\theta}_j, \boldsymbol{v}$.

The multiple-try trans-dimensional move from model $\mathcal{M}_i$ to model $\mathcal{M}_j$ is summarized in Algorithm 6 and briefly explained here. Given the current state $\boldsymbol{\theta}_i$, we first sample $\boldsymbol{u}$ from $q_i(\boldsymbol{u})$, then propose multiple tries $(\boldsymbol{v}^{(k)}, \boldsymbol{\theta}_j^{(k)}) = (\boldsymbol{\theta}_i, \boldsymbol{u}) + r^{(k)} \cdot \boldsymbol{e}$ for $k \in [1:m]$. Two types of directional jumping mechanisms can be applied to our setting: (1) fixed-directional jump; (2) adaptive-directional jump. For the fixed-directional jump, the jumping direction is defined by the two pre-located modes of the augmented posterior distributions $p_i(\boldsymbol{\theta}_i, \boldsymbol{u}, \mathcal{M}_i \mid \boldsymbol{y})$ and $p_j(\boldsymbol{v}, \boldsymbol{\theta}_j, \mathcal{M}_j \mid \boldsymbol{y})$, and is fixed throughout the algorithm. For the adaptive-directional jump, the jumping direction is defined by the current state of the chain $(\boldsymbol{\theta}_i, \boldsymbol{u})$ and the mode of the augmented posterior distribution $p_j(\boldsymbol{v}, \boldsymbol{\theta}_j, \mathcal{M}_j \mid \boldsymbol{y})$.

There are subtle differences in the implementation of the two jumping mechanisms. First, when we set the adaptive-directional jump as *jumping towards* a mode, that is, $\boldsymbol{e} = (\widehat{\boldsymbol{v}} - \boldsymbol{\theta}_i, \widehat{\boldsymbol{\theta}}_j - \boldsymbol{u})/||(\widehat{\boldsymbol{v}} - \boldsymbol{\theta}_i, \widehat{\boldsymbol{\theta}}_j - \boldsymbol{u})||$, we need to sample the jumping distance r from a centered symmetric distribution $p(r)$ in order to use the same simple form of the acceptance ratio as in (27). A more general $p(r)$ can be allowed if we use a generalized form of MTM in Liu et al. (2000), which results in a form more complicated than (27). In contrast, for the fixed-directional jump, we can simply set the jumping direction as $\boldsymbol{e} = (\widehat{\boldsymbol{v}} - \widehat{\boldsymbol{\theta}}_i, \widehat{\boldsymbol{\theta}}_j - \widehat{\boldsymbol{u}})$ without standardization, and sample the jumping distance r from an arbitrary distribution, not necessarily being symmetric and centered at 0. In fact, to push the chain directly jump into the mode $(\widehat{\boldsymbol{v}}, \widehat{\boldsymbol{\theta}}_j)$, we recommend the users to center $p(r)$ around 1. Second, when calculating the acceptance probability, the adaptive-directional jump involves an additional Jacobian (27). However, for the fixed-directional jump, the Jacobian is simply 1 and thus can be omitted.

Each of the two jumping mechanisms has its own advantages depending on the scenarios. For instance, if the local variations around two modes $(\widehat{\boldsymbol{\theta}}_i, \widehat{\boldsymbol{u}})$ and $(\widehat{\boldsymbol{v}}, \widehat{\boldsymbol{\theta}}_j)$ differ significantly, the adaptive-directional jump is more favorable, as the fixed jumping direction can be misleading when the chain jumps from the relatively wider mode to the narrower one. On the other hand, since the sampling distribution of the jumping distance $p(r)$ for the fixed-directional jump can be centered at 1, when the jumping direction $\boldsymbol{e} = (\widehat{\boldsymbol{v}} - \widehat{\boldsymbol{\theta}}_i, \widehat{\boldsymbol{\theta}}_j - \widehat{\boldsymbol{u}})$ is indeed informative, it tends to be more efficient than the adaptive-directional jump in which $p(r)$ centers at 0.

We then sample $(\boldsymbol{v}, \boldsymbol{\theta}_j)$ from the multiple tries $\{(\boldsymbol{v}^{(k)}, \boldsymbol{\theta}_j^{(k)})\}_{k=1}^m$, with probability proportional to the augmented posterior density $p_j(\boldsymbol{v}^{(k)}, \boldsymbol{\theta}_j^{(k)}, \mathcal{M}_j \mid \boldsymbol{y})$. After obtaining $(\boldsymbol{v}, \boldsymbol{\theta}_j)$, we set $(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{u}^{(k)}) = (\boldsymbol{v}, \boldsymbol{\theta}_j) - r^{(k)} \cdot \boldsymbol{e}$

for $k \in [1:m]$. We accept the trans-dimensional proposal $(\boldsymbol{v}, \boldsymbol{\theta}_j)$ with probability α given in (26) and (27), depending on which jumping mechanism we employ. We note that the proposed trans-dimensional move

Algorithm 6 Multiple-Try reversible jump MCMC

For $i \neq j$, given that the current posterior draw $\boldsymbol{\theta}_i$ is from model $\mathcal{M}_i$, the trans-dimensional move to model $\mathcal{M}_j$ is accomplished as follows:

1. Sample the auxiliary variable $\boldsymbol{u}$ from $q_i(\boldsymbol{u})$, which matches the dimension of $\boldsymbol{\theta}_j$ in model $\mathcal{M}_j$.
2. Set the jumping direction and sample the jumping distances.
   (a) (Fixed-directional jump) Set the jumping direction as $\boldsymbol{e} = (\widehat{\boldsymbol{v}} - \widehat{\boldsymbol{\theta}}_i, \widehat{\boldsymbol{\theta}}_j - \widehat{\boldsymbol{u}})$. Sample the jumping distances $r^{(1)}, \cdots, r^{(m)}$ from an arbitrary distribution $p(r)$ (recommend to center $p(r)$ at 1).
   (b) (Adaptive-directional jump) Set the jumping direction as $\boldsymbol{e} = (\widehat{\boldsymbol{v}} - \boldsymbol{\theta}_i, \widehat{\boldsymbol{\theta}}_j - \boldsymbol{u})/||(\widehat{\boldsymbol{v}} - \boldsymbol{\theta}_i, \widehat{\boldsymbol{\theta}}_j - \boldsymbol{u})||$. Sample the jumping distances $r^{(1)}, \cdots, r^{(m)}$ from a *symmetric* distribution $p(r)$ centered at 0.
3. Propose multiple tries: Set $(\boldsymbol{v}^{(k)}, \boldsymbol{\theta}_j^{(k)}) = (\boldsymbol{\theta}_i, \boldsymbol{u}) + r^{(k)} \cdot \boldsymbol{e}$ for $k \in [1:m]$.
4. Sample $(\boldsymbol{v}, \boldsymbol{\theta}_j)$ from $\{(\boldsymbol{v}^{(k)}, \boldsymbol{\theta}_j^{(k)})\}_{k=1}^m$ with probability proportional to $p_j(\boldsymbol{v}^{(k)}, \boldsymbol{\theta}_j^{(k)}, \mathcal{M}_j \mid \boldsymbol{y})$.
5. Given $(\boldsymbol{v}, \boldsymbol{\theta}_j)$, set $(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{u}^{(k)}) = (\boldsymbol{v}, \boldsymbol{\theta}_j) - r^{(k)} \cdot \boldsymbol{e}$ for $k \in [1:m]$.
6. Accept $(\boldsymbol{v}, \boldsymbol{\theta}_j)$ with probability α specified as below:
   (a) (Fixed-directional jump)

$$\alpha = \min\left\{1, \quad \frac{\sum_{k=1}^m p_j\left(\boldsymbol{v}^{(k)}, \boldsymbol{\theta}_j^{(k)}, \mathcal{M}_j \mid \boldsymbol{y}\right)}{\sum_{k=1}^m p_i\left(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{u}^{(k)}, \mathcal{M}_i \mid \boldsymbol{y}\right)}\right\}. \tag{26}$$

   (b) (Adaptive-directional jump)

$$\alpha = \min\left\{1, \quad \frac{\sum_{k=1}^m p_j\left(\boldsymbol{v}^{(k)}, \boldsymbol{\theta}_j^{(k)}, \mathcal{M}_j \mid \boldsymbol{y}\right)}{\sum_{k=1}^m p_i\left(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{u}^{(k)}, \mathcal{M}_i \mid \boldsymbol{y}\right)} \times \left|1 - \frac{r}{||(\widehat{\boldsymbol{v}} - \boldsymbol{\theta}_i, \widehat{\boldsymbol{\theta}}_j - \boldsymbol{u})||}\right|^{d_i + d_j - 1}\right\}. \tag{27}$$

   r is the jumping distance corresponding to the selected try $(\boldsymbol{v}, \boldsymbol{\theta}_j)$.

should be combined with local MCMC moves within each model $\mathcal{M}_k$. Since the current setting is slightly different from that of a typical MTM, we provide here a theoretical validation in the following proposition. The proof of Proposition 1 can be found in Section 7.1.

Proposition 1: The proposed trans-dimensional move, equipped with either the fixed-directional jumping mechanism or the adaptive-directional jumping mechanism, leaves the posterior distribution $p(\boldsymbol{\theta}_k, \mathcal{M}_k \mid \boldsymbol{y})$ invariant.

5 Illustrations

5.1 Log-Gaussian Cox process

We consider estimating the normalizing constant of a Log-Gaussian Cox process model on the pine forest data set studied in Penttinen et al. (1992) and Stoyan and Stoyan (1994). The data contains the locations of 126 Scots pine saplings in a 10×10 m^2 square (see Figure 10(a) in Møller et al. (1998)). We first standardize the locations into unit square and then discretize the unit square into a $M \times M$ regular grid. Let $\boldsymbol{y} = (y_m)_{m \in [1:M]^2}$ denote the number of pine saplings in each grid cell, and let $\boldsymbol{\lambda} = (\lambda_m)_{m \in [1:M]^2}$ denote the latent intensity

process. We assume the following model:

$$[y_m \mid \lambda_m] \sim \text{Poisson}\left(a\lambda_m\right),$$

where $a = M^{-2}$ is the area of each grid cell. The dimension of $\boldsymbol{\lambda}$ is M^2, and in this example, we test out $M = 10, 20, 30$ thus the dimension of the problem is $100, 400, 900$, respectively. We transform $\boldsymbol{\theta} = \log \boldsymbol{\lambda}$ so that all the parameters are defined on $\mathbb{R}$. We specify a Gaussian process prior given as below with constant mean μ_0 and exponential covariance function on $\boldsymbol{\theta} = (\theta_m)_{m\in[1:M]^2}$,

$$\Sigma_0\left(m, n\right) = \sigma^2 \exp\left(-\frac{1}{M\beta}\left|m - n\right|\right), \qquad m, n \in \left[1 : M\right]^2,$$

where we follow the same parameters setting in Møller et al. (1998): $\sigma^2 = 1.91$, $\beta = 1/33$ and $\mu_0 = \log(126) - \sigma^2/2$. The Poisson likelihood is as follows:

$$L\left(\boldsymbol{\theta} \mid \boldsymbol{y}\right) = \prod_{m\in[1:M]^2} \exp\left(\theta_m y_m - a\exp\left(\theta_m\right)\right),$$

thus the unnormalized posterior distribution is $\gamma\left(\boldsymbol{\theta} \mid \boldsymbol{y}\right) = N\left(\boldsymbol{\theta}; \mu_0, \Sigma_0\right) L\left(\boldsymbol{\theta} \mid \boldsymbol{y}\right)$. An approximate mode $\widehat{\boldsymbol{\theta}}$ of $\gamma\left(\boldsymbol{\theta} \mid \boldsymbol{y}\right)$ is obtained using the Newton-Raphson method.

With this example, we compare the performances of three reasonably well-tuned competitors: the WL mixture method, importance sampling with the surrogate distribution as the proposal distribution, and a generalized SMC algorithm detailed in Algorithm 7 in the appendix. The algorithmic settings of the three algorithms are described below. For the WL mixture method, we use $N(\boldsymbol{\mu}_q, \sigma_q^2 I)$ as the surrogate distribution, where $\boldsymbol{\mu}_q = \widehat{\boldsymbol{\theta}}$ and $\sigma_q = 1.0, 1.2, 1.3$ for $M = 10, 20, 30$, respectively. We use HMC local moves around the mixture component γ. The gradient of the log likelihood is

$$\nabla \log L\left(\boldsymbol{\theta} \mid \boldsymbol{y}\right) = \boldsymbol{y} - a\exp(\boldsymbol{\theta}),$$

and the HMC kernel contains 10 leapfrog steps with step size 0.25. We run in total $S = 5 \times 10^4$ iterations for $M = 10$, $S = 10^5$ iterations for $M = 20, 30$, and set $b = S/2$. For importance sampling, we use 10^6 samples so that the computation time for the three methods are comparable (see Table 2). For SMC, we use 2^{10} particles, and run 10 HMC rejuvenation steps (Step 2(f) in Algorithm 7) for each intermediate distribution to diversify the particles. The effective sample size adaptation criterion is set to be $\kappa = 0.5$, and on average there are 14, 16 and 17 intermediate steps for $M = 10, 20, 30$, respectively.

The results are summarized in Table 2. We find that the WL mixture method and SMC produced similar and stable estimates of the targeted log normalizing constant under all settings, whereas importance sampling underestimated the target for both $M = 20, 30$, consistent with our discussion in Section 3.1. Although the WL mixture method had slightly larger standard deviations, its implementation is much easier compared to SMC. Importance sampling is also easy to implement, but is least accurate among the three. For this and the Bayesian Lasso example in the next section, we also compare the convergence speeds of the standard and the accelerated WL algorithms (see Section 2.3). Figure 2 shows that the accelerated algorithm converges much faster than the standard one.

5.2 Hyper-parameter determination for Bayesian Lasso

We consider the Bayesian Lasso method proposed in Park and Casella (2008), which assumes a hierarchical prior on the linear regression coefficients so that the posterior mode corresponds to the Lasso estimator (Tibshirani, 1996). Given a centered and standardized $n \times p$ design matrix X, the response vector $\boldsymbol{y}$ follows $N(X\boldsymbol{\beta}, \sigma^2 I_n)$. Following Park and Casella (2008), we put prior $N(\mathbf{0}_p, \sigma^2 D_\tau)$ on $\boldsymbol{\beta}$, where D_τ is a diagonal matrix $\text{diag}(\tau_1^2, \cdots, \tau_p^2)$. Besides, we put independent hyper-prior $\text{Exp}(\lambda^2/2)$ on τ_j^2 for $j \in [1 : p]$, and put improper prior $p(\sigma^2) \propto 1/\sigma^2$ on σ^2. This completes the full model specification and the unnormalized posterior distribution is given by

$$\gamma(\boldsymbol{\beta}, \boldsymbol{\tau}, \sigma^2 \mid X, \boldsymbol{y}) = \frac{1}{\sigma^2} N(\boldsymbol{y}; X\boldsymbol{\beta}, \sigma^2 I_n) \prod_{j=1}^{p} \text{Exp}\left(\tau_j^2 \mid \lambda^2/2\right).$$

Log normalizing constant estimates			
Dimension	100	400	900
SMC	474.22 (0.16)	490.71 (0.18)	497.60 (0.21)
WL	474.39 (0.10)	490.59 (0.27)	496.94 (0.34)
IS	474.14 (0.24)	487.11 (1.02)	475.98 (1.34)
Computation time (second)			
Dimension	100	400	900
SMC	17.75 (0.74)	178.87 (15.99)	893.74 (66.18)
WL	15.47 (0.46)	182.80 (15.42)	815.42 (56.32)
IS	22.18 (0.84)	186.88 (16.68)	989.81 (92.15)

Table 2: Results summary of the Log-Gaussian-Cox process. WL and IS refer to the WL mixture method and importance sampling, respectively. The reported values are empirical means and standard deviations (in the bracket) of log normalizing constant estimates based on 10 independent runs.

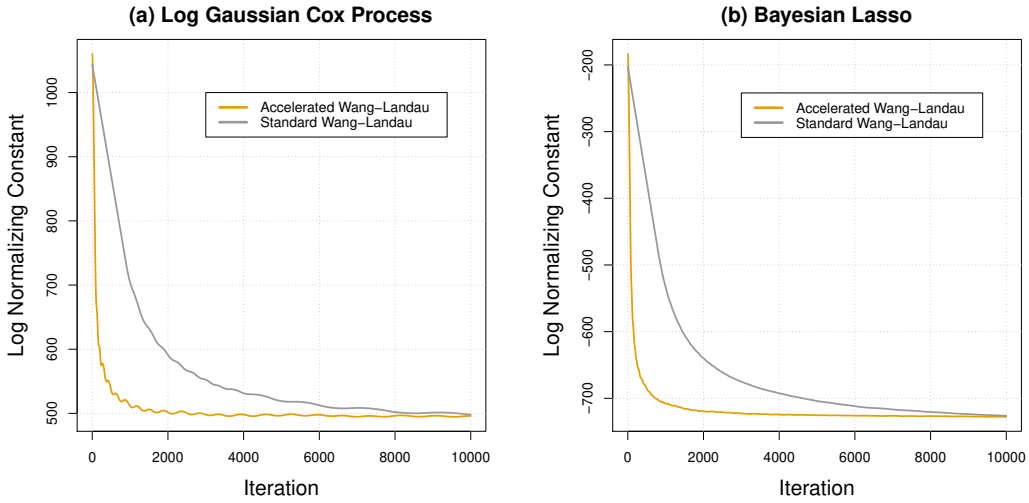


Figure 2: Demonstration of the accelerated Wang-Landau algorithm. (a) The Log-Gaussian Cox process in Section 5.1 with $M = 30$. (b) The Bayesian Lasso example in Section 5.2 with SNR $= 0.1$ and $\lambda = 20$.

We transform the parameters $\eta_j = \log \tau_j^2$ for $j \in [1:p]$ and $\xi = \log \sigma^2$ so that all the parameters are defined on $\mathbb{R}$. In this example, we simulate the data set as in Yang et al. (2016). We define

$$\boldsymbol{\beta}^\star = \mathrm{SNR}\sqrt{\sigma_0^2 \frac{\log p}{n}}(2, -3, 2, 2, -3, 3, -2, 3, -2, 3, 0, \cdots, 0)^\intercal \in \mathbb{R}^p$$

with $p = 100$, $n = 500$, SNR $\in \{0.1, 1, 3\}$ (signal-to-noise ratio), $\sigma_0^2 = 1$. The dimension of the posterior distribution is $2 \times p + 1 = 201$. The design matrix X is generated from a centered multivariate normal distribution with covariance matrix $\Sigma_{ij} = \exp(-|i-j|)$. The response variable $\boldsymbol{y}$ is generated from $N(X\boldsymbol{\beta}^\star, \sigma_0^2 I_n)$. The task is to estimate the marginal likelihood of data for a set of regularization parameters $\lambda \in \{5, 10, 15, 20\}$ under different SNR.

We compare the WL mixture method, Chib's method and importance sampling in this example. The surrogate distribution used in the WL mixture method, which is also the proposal distribution used in importance sampling, is constructed using the variational approximation discussed in Section 2.5. We consider the Normal mean-field variational family where $q(\beta_j)$ is $N(m_j, s_j^2)$, $q(\eta_j)$ is $N(\phi_j, \zeta_j^2)$, and $q(\xi)$ is $N(u, v^2)$. The CAVI updates are summarized in Section 7.3 in the appendix. For the WL mxiture method, the Gibbs move proposed in Park and Casella (2008) is used for the local move around the mixture component γ. We

Signal-to-noise ratio: SNR = 3				
λ	5	10	15	20
WL	-504.03 (0.57)	-491.98 (0.65)	-501.35 (0.32)	-518.13 (0.61)
Chib	-504.63 (0.73)	-492.35 (1.03)	-501.60 (0.43)	-518.40 (0.92)
IS	-508.16 (0.49)	-496.02 (0.77)	-506.83 (0.76)	-524.43 (0.84)
Signal-to-noise ratio: SNR = 1				
λ	5	10	15	20
WL	-742.92 (0.32)	-706.80 (0.48)	-694.93 (0.42)	-689.33 (0.36)
Chib	-743.59 (0.46)	-707.40 (0.47)	-695.28 (0.63)	-690.22 (0.46)
IS	-747.18 (1.29)	-711.67 (1.45)	-700.61 (1.38)	-697.72 (1.38)
Signal-to-noise ratio: SNR = 0.1				
λ	5	10	15	20
WL	-802.28 (0.51)	-758.01 (0.48)	-738.56 (0.41)	-727.31 (0.42)
Chib	-802.92 (0.58)	-758.91 (0.79)	-739.01 (0.77)	-728.10 (1.04)
IS	-806.13 (1.51)	-763.26 (0.90)	-746.09 (1.22)	-735.96 (0.75)

Table 3: Results summary of Bayesian Lasso. WL, Chib and IS refer to the WL mixture method, Chib's method and importance sampling, respectively. The reported values are empirical means and standard deviations (in the bracket) of log normalizing constant estimates based on 10 independent runs. The computation time for the WL mixture method, Chib's method, and importance sampling are 32.64 ($\pm$1.77) seconds, 43.50 ($\pm$1.96) seconds and 51.15 ($\pm$2.64) seconds, respectively.

detail the Gibbs move in the following. The conjugate conditional posterior distributions are

$$\begin{aligned}
[\boldsymbol{\beta} \mid \text{rest}] &\sim N(C_\tau^{-1} X^\intercal \boldsymbol{y}, \sigma^2 C_\tau^{-1}), \quad C_\tau = X^\intercal X + D_\tau^{-1}, \\
[\tau_j^{-2} \mid \text{rest}] &\sim \text{Inverse-Gaussian}(\lambda\sigma/|\beta_j|, \lambda^2), \\
[\sigma^2 \mid \text{rest}] &\sim \text{Inv-}\chi^2 \left(n+p, \frac{||\boldsymbol{y} - X\boldsymbol{\beta}||_2^2 + \boldsymbol{\beta}^\intercal D_\tau^{-1} \boldsymbol{\beta}}{n+p} \right).
\end{aligned}$$

We run a total of $S = 10^4$ iterations, and set $b = 2,000$. In order that the total computation time for the three methods are comparable (see the caption in Table 3), we use 5×10^5 samples for importance sampling. For Chib's method, the parameters are cut into three blocks, $\boldsymbol{\beta}$, $\boldsymbol{\tau}^2$ and σ^2, and we run the Gibbs sampler for 5,000 iterations and burn-in the first 10% samples.

The results are summarized in Table 3. We see that under all settings, the WL mixture method and Chib's method produced similar and stable estimates of the log normalizing constant, whereas importance sampling underestimated the log normalizing constant. The regularization parameter that maximizes the marginal likelihood of data are $\lambda = 10, 20, 20$ for SNR $= 3, 1, 0.1$, respectively, which corresponds to our intuition that it requires more regularization for estimating the regression coefficients when there exists larger noises in the data.

5.3 Logistic regression

We consider a Bayesian logistic regression on the classic German credit dataset (available from the UCI repository (Frank and Asuncion, 2011)). There are in total $n = 1,000$ personal records in the dataset. For each records, there are 24 associated attributes including sex, age, and credit amount. The binary response variable $\boldsymbol{y}$ indicates good or bad credit risks. Let $X_{n\times p}$ be the design matrix after we standardize all the predictors. In particular, we include an intercept and all pairwise interactions. The dimension of the problem is $p = 24 + 24 \times 23/2 + 1 = 301$. We consider the following logistic regression model:

$$\mathbb{P}(y_i = 1 \mid \boldsymbol{x}_i) = \frac{\exp(\alpha + \boldsymbol{\beta}^\intercal \boldsymbol{x}_i)}{1 + \exp(\alpha + \boldsymbol{\beta}^\intercal \boldsymbol{x}_i)},$$

where $y_i \in \{0, 1\}$, $\boldsymbol{x}_i \in \mathbb{R}^{300}$, $\alpha \in \mathbb{R}$, $\boldsymbol{\beta} \in \mathbb{R}^{300}$, $i \in [1 : n]$. All the observations are assumed to be independent. We set up similar priors on the parameters as in Heng and Jacob (2019),

$$[\alpha \mid s^2] \sim N(0, s^2), \quad [\boldsymbol{\beta} \mid s^2] \sim N(\mathbf{0}_{300}, s^2 I_{300}), \quad s^2 \sim \text{Exp}(\lambda),$$

with $\lambda \in \{0.01, 1.00\}$. This completes the full model specification, and the unnormalized posterior distribution is

$$\begin{aligned}\gamma(\alpha, \boldsymbol{\beta}, s^2 \mid \boldsymbol{y}, X) &= p(\alpha, \boldsymbol{\beta} \mid s^2) p(s^2) \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i) \\ &= \lambda e^{-\lambda s^2} N(\alpha; 0, s^2) \prod_{j=1}^{300} N(\boldsymbol{\beta}_j; 0, s^2) \prod_{i=1}^{n} \frac{\exp(\alpha + \boldsymbol{\beta}^\intercal \boldsymbol{x}_i)}{1 + \exp(\alpha + \boldsymbol{\beta}^\intercal \boldsymbol{x}_i)}.\end{aligned}$$

We transform s^2 to the logarithmic scale $\log s^2$ so that all the parameters are defined on $\mathbb{R}$. The task is to estimate the log normalizing constant $\log Z_\gamma$.

We compare the WL mixture method and bridging sampling (BS). We use the same surrogate (proposal) distribution detailed in the following for both methods. We note that the variational approximation is not straightforward as there is a term $\log[1+\exp(\alpha + \boldsymbol{\beta}^\intercal \boldsymbol{x}_i)]$ involved in $\log \gamma(\alpha, \boldsymbol{\beta}, \log s^2 \mid \boldsymbol{y}, X)$. Instead, we use Laplace approximation. We first run a HMC algorithm to obtain posterior samples from $\gamma(\alpha, \boldsymbol{\beta}, \log s^2 \mid \boldsymbol{y}, X)$. Then we fit a multivariate normal distribution on the posterior samples, and choose it as the surrogate distribution. Each HMC step contains 10 leapfrog steps with step size adjusted to be 0.03. For bridge sampling, we use the R package *bridgesampling* (Gronau et al., 2017), and obtain n samples from the target distribution using RStan (Stan Development Team, 2019). Correspondingly, we run $2 \times n$ iterations for the WL mixture method so that approximately we also use n samples from the target distribution.

For this example, we tested out $n = 1000, 1500, 2000, 2500$ for $\lambda \in \{0.01, 1.00\}$. The results are summarized in Figure 3. We see that the WL mixture method has a much better estimation efficiency compared to bridge sampling. Bridge sampling approaches to the vicinity of the correct estimate only after 2,500 iterations/samples for both cases $\lambda = 0.01$ and $\lambda = 1.00$.

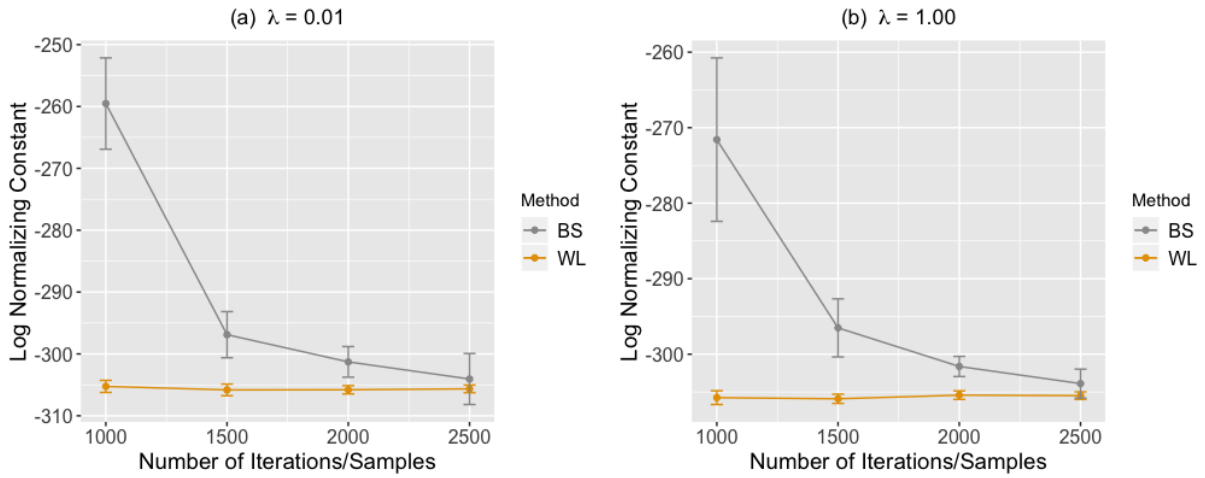


Figure 3: Comparison between the WL mixture method and bridge sampling (BS). For bridge sampling, the x-axis represents the number of samples we draw from the target distribution and the proposal distribution. For the WL mixture method, the x-axis represents half of the total number of iterations we run (see the second to last paragraph in this section). The error bars represent the standard deviations of the log normalizing constant estimates based on 20 independent runs.

5.4 Mixture model

We consider the univariate Gaussian mixture model. The model assumes that $y_1, \cdots, y_n$ are i.i.d. samples from the following mixture distribution:

$$[\boldsymbol{y} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2] \sim \sum_{j=1}^{K} \pi_j N(\mu_j, \sigma_j^2),$$

where n denotes the number of observations, K denotes the number of mixture components, $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_K)$ is defined on the probability simplex, that is, $\pi_1 + \cdots + \pi_K = 1$, $\pi_j > 0$. The prior $p(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ follows the

Galaxy	$K=3$	$K=4$	$K=5$	$K=6$	$K=7$	$K=8$
RJMCMC	0.061	0.128	0.182	0.199	0.160	0.109
PWL	0.058	0.153	0.183	0.200	0.151	0.089
Galaxy‘	$K=9$	$K=10$	$K=11$	$K=12$	$K=13$	$K\geq 14$
RJMCMC	0.071	0.040	0.023	0.013	0.006	0.008
PWL	0.067	0.051	0.019	0.013	0.006	0.010
Acidity	$K=2$	$K=3$	$K=4$	$K=5$	$K=6$	$K=7$
RJMCMC	0.082	0.244	0.236	0.172	0.118	0.069
PWL	0.086	0.245	0.250	0.179	0.095	0.064
Acidity	$K=8$	$K=9$	$K=10$	$K=11$	$K\geq 12$	–
RJMCMC	0.037	0.020	0.011	0.006	0.005	-
PWL	0.042	0.019	0.010	0.005	0.005	-
Enzyme	$K=2$	$K=3$	$K=4$	$K=5$	$K=6$	$K=7$
RJMCMC	0.024	0.290	0.317	0.206	0.095	0.041
PWL	0.036	0.284	0.290	0.191	0.118	0.046
Enzyme	$K=8$	$K=9$	$K\geq 10$	–	–	–
RJMCMC	0.017	0.007	0.003	–	–	–
PWL	0.023	0.008	0.003	–	–	–

Table 4: Posterior distribution of the number of mixture components K for the three data sets: galaxy, acidity, and enzyme. The number of observations for each data set are 82, 155, 245 respectively.

random β model specified in Richardson and Green (1997):

$$\mu_j \sim N(\xi, \kappa^{-1}), \quad \sigma_j^2 \sim \text{Inv-Gamma}(\alpha, \beta), \quad \boldsymbol{\pi} \sim \text{Dirichlet}(\delta, \cdots, \delta), \quad \beta \sim \text{Gamma}(g, h),$$

where $R = \max y_i - \min y_i$, $\xi = [\max y_i + \min y_i]/2$, $\kappa = 1/R^2$, $\alpha = 2$, $g = 0.2$, $h = 10/R^2$, and $\delta = 1$. We set a uniform prior on $[2, K_{\max}]$ for the number of components K, where $K_{\max}$ is a pre-specified upper bound. This completes the full model specification. We are interested in the following two tasks for this example: (1) estimating the normalizing constant for each $K \in [2, K_{\max}]$ so that we can approximate the marginal posterior distribution of K; (2) demonstrating the ability of the WL mixture method in handling multimodality.

Since the posterior distribution for the mixture model is known to be multimodal, it is challenging to construct an appropriate surrogate distribution. Thus, we use the PWL method proposed in Section 4.1. In particular, the surrogate distribution (also η_0) is set to be the prior $p(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, and the sequence of intermediate distributions $\{\eta_t\}$ for $t \in [1:n]$ is chosen following (23), that is, η_t is the (unnormalized) partial posterior distribution $\gamma(\boldsymbol{\mu}, \boldsymbol{\sigma}^2, \boldsymbol{\pi} \mid y_1, \cdots, y_t)$. We use the same Gibbs sampler (detailed in Section 7.4 in the appendix) as in Richardson and Green (1997) to move around the mixture component η_t. We run in total $S = 10^4$ iterations and set $b = S/2$. We applied the PWL method in three datasets: galaxy, acidity and enzyme, which can be found in the R package *multimode*. The numbers of observations in each dataset are 82, 155, 245, respectively. We compare the estimated posterior distribution of K to the results in Richardson and Green (1997). Throughout this example, the specific reversible jump MCMC algorithm proposed in Richardson and Green (1997) is referred to as RJMCMC. Table 4 shows that PWL and RJMCMC produced similar estimates for all three datasets.

To demonstrate the ability of the WL mixture method in handling multimodality, we simulated $n = 100$ data points from a Gaussian mixture distribution with $K = 4$, $\boldsymbol{\pi} = (0.25, 0.25, 0.25, 0.25)$, $\boldsymbol{\mu} = (-3, 0, 3, 6)$, and $\boldsymbol{\sigma} = (0.5, 0.5, 0.5, 0.5)$. The posterior distribution $\gamma(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mid \boldsymbol{y})$ should have 12 symmetric modes in the subspace of (μ_1, μ_2). We choose the prior $p(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ to be the surrogate distribution. We run the WL mixture method for 5×10^5 iterations equipped with the same Gibbs kernel mentioned before. We employ an importance sampling step as discussed in Section 4.2 to properly weight the samples so that they target the posterior distribution $\gamma^\star(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mid \boldsymbol{y})$.

Figure 4 shows that the WL mixture method successfully identified all the 12 modes, whereas the Gibbs sampler, running for the same number of iterations, was still stuck in a local mode. Table 5 compares the estimates of $\mathbb{E}[\mu_j|\boldsymbol{y}]$, whose true value is approximately 1.5, and shows that the WL mixture method significantly outperformed the Gibbs sampler. Based on the identified local modes, Table 5 also shows that

Method	$\mathbb{E}[\mu_1 \mid \boldsymbol{y}]$	$\mathbb{E}[\mu_2 \mid \boldsymbol{y}]$	$\mathbb{E}[\mu_3 \mid \boldsymbol{y}]$	$\mathbb{E}[\mu_4 \mid \boldsymbol{y}]$
WL	1.61 (0.27)	1.38 (0.46)	1.41 (0.36)	1.60 (0.28)
MTM	1.50 (0.11)	1.54 (0.08)	1.51 (0.13)	1.50 (0.08)
Gibbs	-2.92 (0.27)	0.05 (0.52)	3.00 (0.75)	5.89 (0.54)

Table 5: Estimation results obtained using the Gibbs sampler, the WL mixture method and MTM. The truth is approximately 1.50.

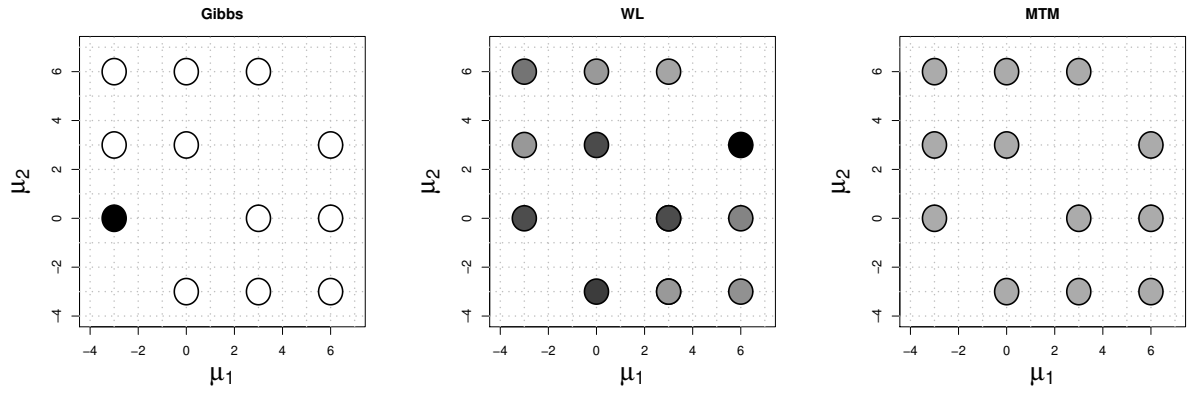


Figure 4: Proportion of time the Markov chain spent in each mode. Darker color corresponds to longer time.

the MTM mode jumping appoach can further improve the estimation accuracy. Specifically, at each iteration, we flip a coin to decide whether we move the current state of the Markov chain using the Gibbs kernel or the MTM kernel. If we choose a MTM directional sampling move, we randomly select a local mode and employ the fixed-directional jump detailed in Algorithm 4. For the tuning parameters, we set $m = 10$ and choose $p(r)$ to be $N(1, 0.1^2)$. We run $1/m$ of the total number of iterations that we run for the WL mixture method and the Gibbs sampler to account for the additional computational effort in the MTM move. Figure 4 shows that the chain obtained using MTM spent approximately equal amount of time in all 12 symmetric modes.

5.5 g-Prior variable selection

We compare the performance of the MTM-RJMCMC proposed in Section 4.3 and that of a standard birth-and-death RJMCMC (BD-RJMCMC, detailed below) in the setting of Bayesian variable selection for the pollution data (McDonald and Schwing, 1973). The response variable $\boldsymbol{y}$ is the age-adjusted mortality rate obtained for the years 1959-1961 in 201 standard metropolitan statistical areas. There are in total $n = 60$ observations. The design matrix X contains $p = 15$ predictors including the average annual precipitation, the average temperature in January and July, and the population per household. We consider the standard linear model assuming that $[\boldsymbol{y} \mid X, \boldsymbol{\beta}, \sigma^2]$ follows $N\left(X\boldsymbol{\beta}, \sigma^2 I\right)$. We center the response variable $\boldsymbol{y}$ so that there is no intercept in the model, and standardize each predictor in the design matrix X.

Let $\boldsymbol{\gamma} \in \{0,1\}^p$ be the binary indicator such that $\gamma_j = 1$ represents that the predictor X_j is selected into the model. We employ the g-prior on parameters $\boldsymbol{\beta}$:

$$\left[\boldsymbol{\beta}_{\boldsymbol{\gamma}} \mid \boldsymbol{\gamma}, \sigma^2\right] \sim N\left(\mathbf{0}_{\boldsymbol{\gamma}}, g\sigma^2 \left(X_{\boldsymbol{\gamma}}^{\intercal} X_{\boldsymbol{\gamma}}\right)^{-1}\right).$$

The g-prior enables us to integrate out $\boldsymbol{\beta}$ so that we can obtain the marginal distribution of $\boldsymbol{\gamma}$:

$$p(\boldsymbol{\gamma} \mid \boldsymbol{y}, X) \propto (g+1)^{-q_{\boldsymbol{\gamma}}/2} \left[\boldsymbol{y}^{\intercal}\boldsymbol{y} - \frac{g}{g+1}\boldsymbol{y}^{\intercal} X_{\boldsymbol{\gamma}} \left(X_{\boldsymbol{\gamma}}^{\intercal} X_{\boldsymbol{\gamma}}\right)^{-1} X_{\boldsymbol{\gamma}} \boldsymbol{y}\right]^{-n/2}, \tag{28}$$

where $q_{\boldsymbol{\gamma}}$ denotes the number of selected predictors. We see that g controls the sparsity of the model, and a larger g induces a sparser model. For σ^2, we use a noninformative prior $p\left(\sigma^2\right) \propto 1/\sigma^2$. This completes the full model specification. The task is to estimate the marginal probability of each predictor being selected. The ground truth is obtained by enumerating all 32,768 possible $\boldsymbol{\gamma}$ and calculating the marginal probability using (28). To compare the MTM-RJMCMC and the BD-RJMCMC, we pretend that we do not have the

privilege to integrate out $\boldsymbol{\beta}$, thus we will sample from the trans-dimensional joint posterior distribution $p\left(\boldsymbol{\beta}_{\boldsymbol{\gamma}}, \boldsymbol{\gamma}, \sigma^2 \mid \boldsymbol{y}, X\right)$.

We use the Gibbs sampler to iterate between the following conditional distributions:

$$\left[\boldsymbol{\beta}_{\boldsymbol{\gamma}}, \boldsymbol{\gamma} \mid \sigma^2, \boldsymbol{y}, X\right] \sim \left(2\pi g\sigma^2\right)^{-\frac{q_{\boldsymbol{\gamma}}}{2}} \left|X_{\boldsymbol{\gamma}}^{\intercal} X_{\boldsymbol{\gamma}}\right|^{\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2}\left[\frac{g+1}{g}\left|\left|X_{\boldsymbol{\gamma}}\boldsymbol{\beta}_{\boldsymbol{\gamma}}\right|\right|^2 - 2\boldsymbol{\beta}_{\boldsymbol{\gamma}}^{\intercal} X_{\boldsymbol{\gamma}}^{\intercal}\boldsymbol{y}\right]\right),$$

$$\left[\sigma^2 \mid \boldsymbol{y}, X, \boldsymbol{\beta}_{\boldsymbol{\gamma}}, \boldsymbol{\gamma}\right] \sim \text{Inv-Gamma}\left(\frac{n+q_{\boldsymbol{\gamma}}}{2}, \frac{1}{2}\left[\frac{1}{g}\left|\left|X_{\boldsymbol{\gamma}}\boldsymbol{\beta}_{\boldsymbol{\gamma}}\right|\right|^2 + \left|\left|\boldsymbol{y} - X_{\boldsymbol{\gamma}}\boldsymbol{\beta}_{\boldsymbol{\gamma}}\right|\right|^2\right]\right).$$

Given $\boldsymbol{\gamma}_t$, the jumping rule for $\boldsymbol{\gamma}_{t+1}$ as described below is the same for both algorithms. We first flip a coin to decide whether we stay in the current model ($\boldsymbol{\gamma}_{t+1} = \boldsymbol{\gamma}_t$) or move to a different model ($\boldsymbol{\gamma}_{t+1} \neq \boldsymbol{\gamma}_t$). If we choose to leave the current model (a trans-dimensional move), we randomly move into a higher dimension (add a predictor) or move into a lower dimension (exclude a predictor) with equal probability 0.5. When the chain is at the boundary ($q_{\boldsymbol{\gamma}}$ is 1 or 15), the proposal going out of the range is automatically rejected. Given $\boldsymbol{\gamma}_{t+1}$, the within-dimensional move (when $\boldsymbol{\gamma}_{t+1} = \boldsymbol{\gamma}_t$) for $\boldsymbol{\beta}_{t+1}$ is conjugate:

$$\left[\boldsymbol{\beta}_{\boldsymbol{\gamma}} \mid \boldsymbol{\gamma}, \sigma^2, \boldsymbol{y}, X\right] \sim N\left(\frac{g}{g+1}\left(X_{\boldsymbol{\gamma}}^{\intercal} X_{\boldsymbol{\gamma}}\right)^{-1} X_{\boldsymbol{\gamma}}^{\intercal}\boldsymbol{y},\ \frac{g\sigma^2}{g+1}\left(X_{\boldsymbol{\gamma}}^{\intercal} X_{\boldsymbol{\gamma}}\right)^{-1}\right).$$

For the trans-dimensional move, the MTM-RJMCMC and the BD-RJMCMC have different proposals for $\boldsymbol{\beta}_{t+1}$. For the MTM-RJMCMC, we follow the fixed-directional jumping mechanism detailed in Algorithm 6. Since we only add or remove one predictor in each trans-dimensional move, the algorithm requires only one auxiliary variable. We choose the auxiliary distribution to be $N(0,1)$. We sample the jumping distance r from $N(1,1)$, and set the number of tries to be $m = 5$. For the BD-RJMCMC, if we choose to add a predictor, we propose it from $N\left(0, 0.5^2\right)$. We run 5×10^4 iterations for the MTM-RJMCMC and 1.5×10^5 iterations for the BD-RJMCMC so that the computation for the two algorithms are comparable (see the caption in Table 6). For both algorithms, we burn-in the first 10% samples.

The estimation results for $g = \exp(10)$ and $g = \exp(15)$, respectively, are summarized in Table 6. We see that the MTM-RJMCMC produced more accurate estimation results than the BD-RJMCMC. In particular, we see that the BD-RJMCMC might have been stuck in a local mode thus mistakenly selected two wrong predictors X_{12} and X_{13}. Intuitively, we see that the directional jumping in the MTM-RJMCMC is much more informative than the blind proposal in the BD-RJMCMC, thus preventing the algorithm from getting stuck in local modes.

Sparsity	$g = \exp(10)$			$g = \exp(15)$		
Predictors	Truth	MTM-RJMCMC	BD-RJMCMC	Truth	MTM-RJMCMC	BD-RJMCMC
X_1	0.118	0.118 (0.016)	0.068 (0.042)	0.036	0.025 (0.013)	0.007 (0.006)
X_2	0.177	0.170 (0.008)	0.115 (0.052)	0.118	0.100 (0.010)	0.043 (0.041)
X_3	0.009	0.009 (0.002)	0.010 (0.003)	0.001	0.001 (0.001)	0.001 (0.001)
X_4	0.020	0.019 (0.003)	0.016 (0.004)	0.012	0.009 (0.002)	0.004 (0.004)
X_5	0.010	0.008 (0.002)	0.005 (0.002)	0.001	0.000 (0.001)	0.001 (0.001)
X_6	0.143	0.148 (0.017)	0.139 (0.035)	0.270	0.278 (0.015)	0.126 (0.118)
X_7	0.005	0.005 (0.001)	0.003 (0.001)	0.001	0.001 (0.002)	0.001 (0.001)
X_8	0.013	0.014 (0.002)	0.013 (0.001)	0.005	0.003 (0.001)	0.001 (0.001)
X_9	0.289	0.300 (0.008)	0.302 (0.027)	0.468	0.527 (0.013)	0.433 (0.112)
X_{10}	0.008	0.008 (0.002)	0.007 (0.001)	0.004	0.004 (0.001)	0.002 (0.001)
X_{11}	0.010	0.009 (0.002)	0.009 (0.003)	0.004	0.003 (0.001)	0.001 (0.001)
X_{12}	0.011	0.013 (0.007)	0.108 (0.110)	0.003	0.001 (0.001)	0.183 (0.149)
X_{13}	0.010	0.011 (0.008)	0.107 (0.111)	0.002	0.000 (0.001)	0.182 (0.148)
X_{14}	0.168	0.163 (0.013)	0.094 (0.065)	0.070	0.050 (0.013)	0.016 (0.016)
X_{15}	0.003	0.003 (0.001)	0.002 (0.001)	0.001	0.000 (0.001)	0.001 (0.001)

Table 6: Estimates of the marginal probability of each variable being selected. The reported values are empirical means and standard deviations (in the bracket) based on 10 independent runs. The computation time for the MTM-RJMCMC and the BD-RJMCMC are 39.2 ($\pm$1.01) seconds and 59.8 ($\pm$1.24) seconds, respectively.

6 Concluding Remarks

We have described a general strategy to construct a mixture of the unnormalized target posterior distribution and a surrogate distribution with a known normalizing constant for estimating the Bayes factor. Such a mixture formulation allows us to use the generalized WL algorithm and the MTM machinery for fast MCMC mixing and accurate estimation of the normalizing constant. We have also designed acceleration and parallelization schemes to further improve its performance.

By efficiently jumping back and forth between the target distribution and the surrogate distribution, possibly with the help of mode jumping algorithms such as MTM, the performance of the WL mixture method is less sensitive to the potential separation between the target distribution and the surrogate distribution compared to importance sampling based methods. The WL mixture method also has more general applicability compared to Chib's method, when the sampler of the target distribution involves more sophisticated MCMC steps beyond the closed-form Gibbs sampler (all conditional distributions are easy to sample from) or standard Metropolis-Hasting algorithms. In addition, the WL mixture method requires less effort in delicate tuning in its implementation compared to other advanced methods such as path sampling, reversible-jump MCMC, and sequential Monte Carlo methods.

There are several future directions that we would like to follow. First, although we have shown the power of the WL mixture method, a rigorous theoretical framework is required to better understand the nature of the method. Second, instead of mixing the target distribution with a single surrogate distribution, a multiple-component mixture formulation can be considered. Third, although the intuitive idea of first using some deterministic algorithm to find modes and then conducting MCMC to do mode jumping has been around, an efficient way of achieving the intended goal has not been formulated precisely. Our proposed MTM-enhanced jumping strategy, together with the WL adjustment, can help achieve the goal. It is particularly useful to identify some specific classes of models where this general methodology is straightforward and effective to apply.

References

Atchadé, Y. F. and J. S. Liu (2010). The Wang-Landau algorithm in general state spaces: applications and convergence analysis. *Statistica Sinica*, 209–233.

Berg, B. A. and T. Neuhaus (1992). Multicanonical ensemble: a new approach to simulate first-order phase transitions. *Physical Review Letters 68*(1), 9.

Bishop, C. M. (2006). Pattern recognition and machine learning (information science and statistics) springer-verlag new york. *Inc. Secaucus, NJ, USA*.

Blei, D. M., A. Kucukelbir, and J. D. McAuliffe (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association 112*(518), 859–877.

Bornn, L., P. E. Jacob, P. Del Moral, and A. Doucet (2013). An adaptive interacting Wang-Landau algorithm for automatic density exploration. *Journal of Computational and Graphical Statistics 22*(3), 749–773.

Brooks, S. P., P. Giudici, and G. O. Roberts (2003). Efficient construction of reversible jump Markov chain Monte Carlo proposal distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 65*(1), 3–39.

Chib, S. (1995). Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association 90*(432), 1313–1321.

Chib, S. and I. Jeliazkov (2001). Marginal likelihood from the Metropolis-Hastings output. *Journal of the American Statistical Association 96*(453), 270–281.

Chopin, N. (2002). A sequential particle filter method for static models. *Biometrika 89*(3), 539–552.

Dai, C. and J. S. Liu (2019). The Wang-Landau algorithm as stochastic optimization and its acceleration. *arXiv:1907.11985*.

Del Moral, P. (2004). Feynman-kac formulae: Genealogical and interacting particle systems with applications.

Del Moral, P., A. Doucet, and A. Jasra (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 68*(3), 411–436.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological) 39*(1), 1–22.

Diebolt, J. and C. P. Robert (1994). Estimation of finite mixture distributions through Bayesian sampling. *Journal of the Royal Statistical Society: Series B (Methodological) 56*(2), 363–375.

Doucet, A., S. Godsill, and C. Andrieu (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing 10*(3), 197–208.

Frank, A. and A. Asuncion (2011). Uci machine learning repository, 2010. *URL http://archive. ics. uci. edu/ml 15*, 22.

Gelfand, A. E. and A. F. M. Smith (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association 85*(410), 398–409.

Gelman, A. and X. Meng (1994). Path sampling for computing normalizing constants: identities and theory. *University of Chicago Department of Statistics Technical Report* (377).

Geyer, C. (1991). Markov chain Monte Carlo maximum likelihood. In E. Keramigas (Ed.), *Computing Science and Statistics: he 23rd symposium on the interface*, Fairfax, pp. 156–163. Interface Foundation.

Geyer, C. J. (1994). Estimating normalizing constants and reweighting mixtures.

Geyer, C. J. and E. A. Thompson (1995). Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association 90*(431), 909–920.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika 82*(4), 711–732.

Gronau, Q. F., H. Singmann, and E. Wagenmakers (2017). Bridgesampling: an R package for estimating normalizing constants. *arXiv:1710.08162*.

Hammersley, J. M. and K. W. Morton (1954). Poor man's Monte Carlo. *Journal of the Royal Statistical Society: Series B (Methodological) 16*(1), 23–38.

Heng, J. and P. E. Jacob (2019). Unbiased Hamiltonian Monte Carlo with couplings. *Biometrika 106*(2), 287–302.

Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, and L. K. Saul (1999). An introduction to variational methods for graphical models. *Machine Learning 37*(2), 183–233.

Kong, A., J. S. Liu, and W. H. Wong (1994). Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association 89*(425), 278–288.

Liang, F. (2005). A generalized Wang-Landau algorithm for Monte Carlo computation. *Journal of the American Statistical Association 100*(472), 1311–1327.

Liang, F. and W. H. Wong (2001). Real-parameter evolutionary Monte Carlo with applications to Bayesian mixture models. *Journal of the American Statistical Association 96*(454), 653–666.

Liu, J. S. (2008). *Monte Carlo strategies in scientific computing*. Springer Science & Business Media.

Liu, J. S. and R. Chen (1998). Sequential Monte Carlo methods for dynamic systems. *Journal of the American statistical association 93*(443), 1032–1044.

Liu, J. S., R. Chen, and T. Logvinenko (2001). A theoretical framework for sequential importance sampling with resampling. In *Sequential Monte Carlo methods in practice*, pp. 225–246. Springer.

Liu, J. S., F. Liang, and W. H. Wong (2000). The multiple-try method and local optimization in Metropolis sampling. *Journal of the American Statistical Association 95*(449), 121–134.

Marinari, E. and G. Parisi (1992). Simulated tempering: a new Monte Carlo scheme. *EPL (Europhysics Letters) 19*(6), 451.

McDonald, G. C. and R. C. Schwing (1973). Instabilities of regression estimates relating air pollution to mortality. *Technometrics 15*(3), 463–481.

Meng, X. and S. Schilling (1996). Fitting full-information item factor models and an empirical investigation of bridge sampling. *Journal of the American Statistical Association 91*(435), 1254–1267.

Meng, X. and W. H. Wong (1996). Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Statistica Sinica*, 831–860.

Møller, J., A. R. Syversveen, and R. P. Waagepetersen (1998). Log Gaussian Cox processes. *Scandinavian Journal of Statistics 25*(3), 451–482.

Neal, R. M. (2001). Annealed importance sampling. *Statistics and Computing 11*(2), 125–139.

Ogata, Y. (1989). A Monte Carlo method for high dimensional integration. *Numerische Mathematik 55*(2), 137–157.

Pandolfi, S., F. Bartolucci, and N. Friel (2014). A generalized multiple-try version of the reversible jump algorithm. *Computational Statistics & Data Analysis 72*, 298–314.

Park, T. and G. Casella (2008). The Bayesian Lasso. *Journal of the American Statistical Association 103*(482), 681–686.

Penttinen, A., D. Stoyan, and H. M. Henttonen (1992). Marked point processes in forest statistics. *Forest Science 38*(4), 806–824.

Richardson, S. and P. J. Green (1997). On Bayesian analysis of mixtures with an unknown number of components (with discussion). *Journal of the Royal Statistical Society: series B (statistical methodology) 59*(4), 731–792.

Rosenbluth, M. N. and A. W. Rosenbluth (1955). Monte Carlo calculation of the average extension of molecular chains. *The Journal of Chemical Physics 23*(2), 356–359.

Stan Development Team (2019). RStan: the R interface to Stan. R package version 2.19.2.

Stoyan, D. and H. Stoyan (1994). *Fractals, random shapes, and point fields: methods of geometrical statistics*, Volume 302. John Wiley & Sons Inc.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological) 58*(1), 267–288.

Tjelmeland, H. and B. K. Hegstad (2001). Mode jumping proposals in MCMC. *Scandinavian Journal of Statistics 28*(1), 205–223.

Tzikas, D. G., A. C. Likas, and N. P. Galatsanos (2008). The variational approximation for Bayesian inference. *IEEE Signal Processing Magazine 25*(6), 131–146.

Wainwright, M. J. and M. I. Jordan (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning 1*(1–2), 1–305.

Wang, F. and D. P. Landau (2001). Efficient, multiple-range random walk algorithm to calculate the density of states. *Physical Review Letters 86*(10), 2050.

Yang, Y., M. J. Wainwright, and M. I. Jordan (2016). On the computational complexity of high-dimensional Bayesian variable selection. *The Annals of Statistics 44*(6), 2497–2532.

7 Appendix

7.1 Proof of Proposition 1

Proof: We first consider the fixed-directional jump. In the following, we calculate the transition probability $\mathbb{P}\left((\boldsymbol{\theta}_j, \mathcal{M}_j) \,|\, (\boldsymbol{\theta}_i, \mathcal{M}_i)\right)$. Suppose the transition from $(\boldsymbol{\theta}_i, \mathcal{M}_i)$ to $(\boldsymbol{\theta}_j, \mathcal{M}_j)$ is achieved by some jumping distance r , then we have:

$$\boldsymbol{\theta}_j = \boldsymbol{u} + r(\widehat{\boldsymbol{\theta}}_j - \widehat{\boldsymbol{u}}) \quad \text{and} \quad \boldsymbol{v} = \boldsymbol{\theta}_i + r(\widehat{\boldsymbol{v}} - \widehat{\boldsymbol{\theta}}_i).$$

It follows that

$$\begin{aligned}
\mathbb{P}\left((\boldsymbol{\theta}_j, \mathcal{M}_j) \,|\, (\boldsymbol{\theta}_i, \mathcal{M}_i)\right) &= \int q_i(\boldsymbol{u}) \times \frac{p_j\left(\boldsymbol{v}, \boldsymbol{\theta}_j, \mathcal{M}_j \mid \boldsymbol{y}\right)}{\sum_{k=1}^m p_j(\boldsymbol{v}^{(k)}, \boldsymbol{\theta}_j^{(k)}, \mathcal{M}_j \mid \boldsymbol{y})} \times \min\left\{1, \frac{\sum_{k=1}^m p_j\left(\boldsymbol{v}^{(k)}, \boldsymbol{\theta}_j^{(k)}, \mathcal{M}_j \mid \boldsymbol{y}\right)}{\sum_{k=1}^m p_i\left(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{u}^{(k)}, \mathcal{M}_i \mid \boldsymbol{y}\right)}\right\} \\
&\quad \times \prod_{k=1}^{m-1} p(r^{(k)}) p(r) dr^{(1)} \cdots dr^{(m-1)} \\
&= \int \min\left\{ \left[\sum_{k=1}^m p_j\left(\boldsymbol{v}^{(k)}, \boldsymbol{\theta}_j^{(k)}, \mathcal{M}_j \mid \boldsymbol{y}\right)\right]^{-1}, \left[\sum_{k=1}^m p_i\left(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{u}^{(k)}, \mathcal{M}_i \mid \boldsymbol{y}\right)\right]^{-1} \right\} \\
&\quad \times \prod_{k=1}^{m-1} p(r^{(k)}) dr^{(1)} \cdots dr^{(m-1)} \times p\left(\boldsymbol{\theta}_j, \mathcal{M}_j \mid \boldsymbol{y}\right) q_i(\boldsymbol{u}) q_j(\boldsymbol{v}) p(r),
\end{aligned}$$

where

$$\boldsymbol{\theta}_j^{(k)} = \boldsymbol{u} + r^{(k)}(\widehat{\boldsymbol{\theta}}_j - \widehat{\boldsymbol{u}}), \quad \boldsymbol{v}^{(k)} = \boldsymbol{\theta}_i + r^{(k)}(\widehat{\boldsymbol{v}} - \widehat{\boldsymbol{\theta}}_i), \quad \boldsymbol{\theta}_i^{(k)} = \boldsymbol{v} - r^{(k)}(\widehat{\boldsymbol{v}} - \widehat{\boldsymbol{\theta}}_i), \quad \boldsymbol{u}^{(k)} = \boldsymbol{\theta}_j - r^{(k)}(\widehat{\boldsymbol{\theta}}_j - \widehat{\boldsymbol{u}}).$$

We note that when we jump back from $(\boldsymbol{v}, \boldsymbol{\theta}_j)$ to $(\boldsymbol{\theta}_i, \boldsymbol{u})$, we flip the sign of the jumping direction, that is, changing the jumping direction to $(\widehat{\boldsymbol{u}} - \widehat{\boldsymbol{\theta}}_j, \widehat{\boldsymbol{\theta}}_i - \widehat{\boldsymbol{v}})$, so that we keep the same jumping distance r. This is the reason why we do not require the sampling distribution of the jumping distance $p(r)$ to be symmetric and centered at 0. Since the Jacobian between $(\boldsymbol{v}, \boldsymbol{\theta}_j)$ and $(\boldsymbol{\theta}_i, \boldsymbol{u})$ is simply 1, and the multiple integral is symmetric in the index i and j, the transition kernel satisfies the detailed balance condition thus leaves $p\left(\boldsymbol{\theta}_k, \mathcal{M}_k \mid \boldsymbol{y}\right)$ invariant.

For the adaptive-directional jump, to ensure the reversibility of the transition kernel, we first standardize the jumping direction, that is, setting $\boldsymbol{e} = (\widehat{\boldsymbol{v}} - \boldsymbol{\theta}_i, \widehat{\boldsymbol{\theta}}_j - \boldsymbol{u})/||(\widehat{\boldsymbol{v}} - \boldsymbol{\theta}_i, \widehat{\boldsymbol{\theta}}_j - \boldsymbol{u})||$, so that the jumping distance becomes independent to the current state of the chain $(\boldsymbol{\theta}_i, \boldsymbol{u})$. Besides, because the jumping direction always points to the modes of the augmented posterior distributions defined in (25), we should flip the sign of the jumping distance when we jump back from $(\boldsymbol{v}, \boldsymbol{\theta}_j)$ to $(\boldsymbol{\theta}_i, \boldsymbol{u})$ as we won't flip the sign of the jumping direction. Consequently, the sampling distribution of the jumping distance is required to be symmetric and centered at 0.

The proof of the reversibility of the transition kernel equipped with the adaptive-directional jump follows similarly (thus is omitted) as the case of the fixed-directional jump, but requires additional calculations of the Jacobian, which are detailed as follows. Suppose the transition from $(\boldsymbol{\theta}_i, \mathcal{M}_i)$ to $(\boldsymbol{\theta}_j, \mathcal{M}_j)$ is achieved by some jumping distance r, then we have:

$$\boldsymbol{\theta}_j = \boldsymbol{u} + r\frac{\widehat{\boldsymbol{\theta}}_j - \boldsymbol{u}}{||(\widehat{\boldsymbol{v}} - \boldsymbol{\theta}_i, \widehat{\boldsymbol{\theta}}_j - \boldsymbol{u})||} \quad \text{and} \quad \boldsymbol{v} = \boldsymbol{\theta}_i + r\frac{\widehat{\boldsymbol{v}} - \boldsymbol{\theta}_i}{||(\widehat{\boldsymbol{v}} - \boldsymbol{\theta}_i, \widehat{\boldsymbol{\theta}}_j - \boldsymbol{u})||}.$$

We define $\boldsymbol{x} \in \mathbb{R}^{d_i + d_j}$ as follows. For $1 \le k \le d_j$, let $x_k = (\widehat{\theta}_{jk} - u_k)/||(\widehat{\boldsymbol{v}} - \boldsymbol{\theta}_i, \widehat{\boldsymbol{\theta}}_j - \boldsymbol{u})||^{3/2}$. For $1 \le l \le d_i$, let $x_{d_j + l} = (\widehat{v}_l - \theta_{il})/||(\widehat{\boldsymbol{v}} - \boldsymbol{\theta}_i, \widehat{\boldsymbol{\theta}}_j - \boldsymbol{u})||^{3/2}$. Then we have:

$$\left|\frac{\partial(\boldsymbol{\theta}_j, \boldsymbol{v})}{\partial(\boldsymbol{u}, \boldsymbol{\theta}_i)}\right| = \det\left((1 - r||\boldsymbol{x}||^2) I_{d_j} + r\boldsymbol{x}\boldsymbol{x}^\intercal\right) = \left[1 - \frac{r}{||(\widehat{\boldsymbol{v}} - \boldsymbol{\theta}_i, \widehat{\boldsymbol{\theta}}_j - \boldsymbol{u})||}\right]^{d_i + d_j - 1}.$$

7.2 SMC algorithm

Algorithm 7 Sequential Monte Carlo (SMC)

Input: proposal distribution $q(\boldsymbol{\theta})$, Markov kernels $\{K_t\}$

1. Initialization:
   (a) sample $\boldsymbol{\theta}_0^{(i)}$ from $q(\boldsymbol{\theta})$ for $i \in [1:n]$ independently;
   (b) set $\lambda_0 = 0$ and $t = 0$.
2. While $\lambda_t < 1$ iterate the following steps:
   (a) set $t \leftarrow t + 1$;
   (b) for some pre-specified $\kappa \in (0,1)$, select λ_t based on λ_{t-1} and $\{\boldsymbol{\theta}_{t-1}^{(i)}\}_{i\in[1:n]}$ by solving the following equation

$$n^{-1}\mathrm{ESS}_t(\lambda) = \frac{\left(n^{-1}\sum_{i=1}^n (\gamma/q)(\boldsymbol{\theta}_{t-1}^{(i)})^{\lambda-\lambda_{t-1}}\right)^2}{n^{-1}\sum_{i=1}^n (\gamma/q)(\boldsymbol{\theta}_{t-1}^{(i)})^{2(\lambda-\lambda_{t-1})}} = \kappa;$$

   (c) if $\lambda_t = 1$, set $T = t$;
   (d) compute weights for $i \in [1:n]$

$$w_t^{(i)} = \frac{\gamma_t(\boldsymbol{\theta}_{t-1}^{(i)})}{\gamma_{t-1}(\boldsymbol{\theta}_{t-1}^{(i)})},$$

   where the (unnormalized) intermediate distribution $\gamma_t(\boldsymbol{\theta}) = \gamma(\boldsymbol{\theta})^{\lambda_t} q(\boldsymbol{\theta})^{1-\lambda_t}$.
   (e) resample particles $\{\boldsymbol{\theta}_{t-1}^{(i)}\}_{i\in[1:n]}$ proportional to weights $\{w_t^{(i)}\}_{i\in[1:n]}$ to obtain a new set of particles $(\hat{\boldsymbol{\theta}}_t^{(i)})_{i\in[1:n]}$;
   (f) sample $\boldsymbol{\theta}_t^{(i)} \sim K_t(\hat{\boldsymbol{\theta}}_t^{(i)}, \cdot)$ for $i \in [1:n]$;

Output: normalizing constant estimates $\widehat{Z}_\gamma = \prod_{t=1}^T \frac{1}{n}\sum_{i=1}^n w_t^{(i)}$.

7.3 Variational approximation for Bayesian Lasso

Algorithm 8 CAVI updates for Bayesian Lasso

1. Given $q(\beta_i)$ for $i \neq j$, $q(\boldsymbol{\eta})$ and $q(\xi)$, update $q(\beta_j)$.

$$(m_j, s_j) = \underset{(m,\ s>0)}{\arg\max} \left\{ e^{-u+\frac{v^2}{2}} \left[A_{-j} m - \frac{1}{2} \left((X^\intercal X)_{jj} + e^{-\phi_j + \frac{\zeta_j^2}{2}} \right) (m^2 + s^2) \right] + \frac{1}{2} \log s^2 \right\},$$

   where $A_{-j} = \sum_{k=1}^n X_{kj} y_k - \sum_{k \neq j} (X^\intercal X)_{kj} m_k$.

2. Given $q(\eta_i)$ for $i \neq j$, $q(\boldsymbol{\beta})$ and $q(\xi)$, update $q(\eta_j)$.

$$(\phi_j, \zeta_j) = \underset{(\phi,\ \zeta>0)}{\arg\max} \left\{ \phi - \lambda^2 e^{\phi + \frac{\zeta^2}{2}} - e^{-u+\frac{v^2}{2}} (m_j^2 + s_j^2) e^{-\phi + \frac{\zeta^2}{2}} + \log \zeta^2 \right\}.$$

3. Given $q(\boldsymbol{\beta})$, $q(\boldsymbol{\eta})$, update $q(\xi)$.

$$(u, v) = \underset{(u,\ v>0)}{\arg\max} \left\{ -(n+p)u - B e^{-u+\frac{v^2}{2}} + \log v^2 \right\},$$

   where $B = \boldsymbol{y}^\intercal \boldsymbol{y} - 2\boldsymbol{y}^\intercal X m + \operatorname{tr}\left[(m m^\intercal + \operatorname{diag}(s_k^2)) X^\intercal X \right] + \sum_{k=1}^p (m_k^2 + s_k^2) e^{-\phi_k + \frac{\zeta_k^2}{2}}$.

7.4 Gibbs sampler for mixture model

Algorithm 9 Gibbs sampler for mixture model

1. Given $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2$, sample $\boldsymbol{Z} = (Z_1, \cdots, Z_n)$ from the following distribution:

$$\mathbb{P}(Z_i = j \mid \boldsymbol{\mu}, \boldsymbol{\sigma}^2, \boldsymbol{y}) \propto \pi_j N(y_i; \mu_j, \sigma_j^2), \quad j \in [1:K].$$

2. Given $\boldsymbol{Z}$, sample $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_K)$ from Dirichlet$(\delta + n_1, \cdots, \delta + n_K)$. $n_j = \sum_{i=1}^n \mathbb{1}(Z_i = j)$ for $j \in [1:K]$.

3. Given $\boldsymbol{Z}, \boldsymbol{\sigma}^2$, sample $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_K)$ from:

$$[\mu_j \mid \sigma_j^2, \boldsymbol{Z}, \boldsymbol{y}] \sim N\left(\frac{\sum_{i:Z_i=j} y_i/\sigma_j^2 + \kappa\xi}{n_j/\sigma_j^2 + \kappa}, \frac{1}{n_j/\sigma_j^2 + \kappa} \right), \quad j \in [1:K].$$

4. Given $\boldsymbol{Z}, \boldsymbol{\mu}, \beta$, sample $\boldsymbol{\sigma}^2 = (\sigma_1^2, \cdots, \sigma_K^2)$ from:

$$[\sigma_j^2 \mid \boldsymbol{Z}, \mu_j, \beta, \boldsymbol{y}] \sim \text{Inv-Gamma}\left(\alpha + \frac{1}{2} n_j, \beta + \frac{1}{2} \sum_{i:Z_i=j} (y_i - \mu_j)^2 \right).$$

5. Given $\boldsymbol{\sigma}^2$, sample β from Gamma$(g + K\alpha, h + \sum_{j=1}^K 1/\sigma_j^2)$.